INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Monocular Human Motion Capture

And Other Works, 2000–2004

Collected Scientific Publications

in support of the degree of

Habilitation à Diriger des Recherches

presented and defended by

William Triggs

the 7th January 2005 at GRAVIR, Montbonnot, France

Jury

Prof. Roger Mohr (INP, Grenoble)	chairman
Prof. Philippe Cinquin (U. Joseph Fourier, Grenoble)	reporters
Prof. Jean Ponce (Beckman Institute & U. Illinois)	
Prof. Luc Van Gool (KU Leuven & ETH Zürich)	(apologies)
Prof. Olivier Faugeras (INRIA, Sophia-Antipolis)	examiners
Prof. Andrew Zisserman (U. Oxford)	

To my students and colleagues.

Contents

Preface

This is a supporting document for my application for the French post-doctoral diploma « Habilitation à Diriger des Recherches » (HDR). It contains the fullest existing versions of most of the articles that I have written or co-authored between January 2000 and August 2004. See my PhD thesis for a similar collection of papers written between 1995 and November 1999.

Organization

The papers are organized thematically rather than chronologically. Each chapter contains a short introduction and executive summary listing the main contributions of each paper, followed by the papers themselves. To avoid unnecessary duplication, only the most complete existing version of each work is included: the final or submitted journal version if one exists, otherwise the (possibly extended) conference paper. The paper summaries mention any previously published versions that have not been included. Barring a few extensions made at the time, the papers are reproduced essentially in the form in which they originally appeared. None are perfect, but I have resisted the temptation to update them.

The research presented here spans several different thematic areas. For convenience it is divided into two parts.

The first part contains 7 papers on monocular uninstrumented "motion capture" — the detection, tracking and reconstruction of human pose and motion from monocular images and image sequences — divided into 3 chapters, respectively on 3D motion capture using a model based approach, 2D detection and tracking, and 3D motion capture using a learning based approach.

The second part contains 10 papers on various other topics: low-level vision and image processing; vision geometry and scene reconstruction; and statistical modelling and object recognition.

Roughly speaking, the work on vision geometry dates from the period immediately after my thesis and is no longer my principal focus, while the work on machine learning and visual object recognition is a (slowly!) emerging theme. The work on human motion and on low-level vision, features & descriptors runs throughout the period and is likely to continue for the foreseeable future.

Part I

Monocular Human Motion Capture

Chapter 1

3D Model Based Motion Capture

This chapter contains four papers that adopt a 3D model based approach to recovering 3D human pose and motion from monocular image sequences. Each paper offers a different approach to the important problem of searching for nearby local minima of a complex cost function in a high-dimensional parameter space, and applies it to the human tracking problem. Only the last method is tied to the special structure of this problem — the other three methods are generic. All four papers were written jointly with my then PhD student Christian Sminchisescu, whose PhD thesis [Smi02] can be consulted for further details.

Summary of paper 1, "Estimating Articulated Human Motion With Covariance Scaled Sampling"

This paper appeared in a special issue on human motion published by the International Journal of Robotics Research (IJRR) [ST03a]. Earlier versions of the work were published as an INRIA technical report [ST01b] and in the 2001 International Conference on Computer Vision and Pattern Recognition (CVPR) [ST01a]. After giving details of our overall model and our multi-feature model-image matching cost function, the paper describes a quasi-local random search method called "Covariance Scaled Sampling". This is probably best seen as an effort to make Markov Chain Monte Carlo samplers such as particle filter trackers ('Condensation') feasible in high-dimensional multi-modal problems. The three key insights are:

1. **Shaped sampling:** For ill-conditioned problems in many dimensions, isotropic sampling is not enough. Samples should be distributed in a way that is sensitive to the local shape of the cost function, otherwise most are wasted in high cost regions. For example, for an isotropic sampler in a long narrow 'trough' or 'tube' in a high dimensional cost surface, most samples will fall on the steep sidewalls of the trough, and only a few will fall along the long axis. The workaround proposed is to use the local curvature of the cost function as a covariance model to scale the sampler.

2. **Long tailed sampling:** Adjacent local minima are often separated by tens or even hundreds of standard deviations, so shaped sampling is not enough. To reach the basins of attraction of other minima with reasonable probability, one needs to sample from a deliberately broadened ("long tailed") distribution.

3. **Optimized samples:** Sampling alone is not enough. In high dimensions, volume increases very rapidly with distance, so the low-cost 'cores' of minima are tiny compared to the volume occupied by their surrounding basins of attraction. Hence, even if a random sample falls in the basin of attraction, it is very likely to fall at a high-cost point and thus be rejected by the MCMC sampler. The work-around proposed is to perform several steps of optimization starting from the initial sample, before deciding whether to accept or reject the sample.

Given these three insights, it is easy to make a sampler with the required properties. In practice this method works quite well, but it still can not achieve very deep 'penetration' into the space of other possible minima within a reasonable number of samples.

Summary of paper 2, "Building Roadmaps of Minima and Transitions in Visual Models"

This paper has been accepted by the International Journal of Computer Vision and is scheduled for publication in early 2005. An earlier version of the work appeared in the 2002 European Conference on Computer Vision [ST02b]. It returns to the local minimum problem by noting that any path between two minima must necessarily have a maximum value somewhere. If this maximum is minimized, the path necessarily crosses a 'col' or 'mountain pass' between the two minima. There is a "transition state" — a stationary point with one negative curvature direction — at the top of the path, and if we can locate this, it is easy to slide downhill to find the next minima. Hence finding adjacent minima is reduced to climbing uphill while searching for nearby transition states. It turns out that there are several heuristic methods for doing this, originally developed by computational chemists for molecular configuration and surface adsorption calculations, but previously unknown in vision and even in optimization. In this paper we present generalizations of two of these methods, and apply them to the human pose problem. Both are variants on damped-Newton-based local descent minimization, modified to climb uphill towards a transition state saddle point. **Eigenvector tracking** explicitly flips the sign of one eigenvalue of the Hessian matrix, so that the damped optimization moves uphill in that direction while still trying to descend in all the others. This is simple and it works quite well, but it can also be rather erratic as there is no global progress criterion. **Hypersurface sweeping** sweeps a hypersurface through the space (a moving plane or expanding ellipsoid), tracking a local minimum within the surface. This gives better global guarantees, however it is 'blind' to transition states that are oriented transversally to the sweeping surface. Both methods work rather well in the human pose problem, in each case finding hundreds of local minima with a comparatively modest amount of computation.

Summary of paper 3, "Fast Mixing Hyperdynamic Sampling"

This paper has been accepted for a special issue of the Journal of Image and Vision Computing containing extended versions of selected ECCV 2002 papers, and should be published in late 2004 or early 2005. An earlier version appeared in the 2002 European Conference on Computer Vision [ST02a]. It applies another computational chemistry method, "hyperdynamics", to the problem of finding transition regions in human pose spaces. In contrast to the approaches studied in the previous paper, hyperdynamics is a randomized method, using Markov Chain Monte Carlo sampling in a modified potential designed to promote stochastic drift away from the original cost minimum towards regions containing codimension 1 saddle points. The eigendecomposition of the Hessian is again needed, to create this potential.

Summary of paper 4, "Kinematic Jump Processes For Monocular 3D Human Tracking"

This paper was presented at the 2003 International Conference on Computer Vision and Pattern Recognition [ST03b]. A journal version is currently under preparation. The main idea is to exploit the specific, known structure of the kinematic local minima in the human pose problem, exploring the space of kinematically possible pose solutions by explicitly generating forwards-backwards 'flips' of limbs to jump between linked minima. These "kinematic jump" methods turn out to be very efficient at exploring the kinematic ambiguities of the problem, significantly enhancing the practical tracking reliability. However a complementary search method such as Covariance Scaled Sampling is still needed to handle image matching ambiguities.

Estimating Articulated Human Motion With Covariance Scaled Sampling

Cristian Sminchisescu **Bill Triggs**
INRIA Rhône-Alpes, GRAVIR-CNRS, 655 avenue de l'Europe, 38330 Montbonnot, France
{*Cristian.Sminchisescu,Bill.Triggs*}*@inrialpes.fr, www.inrialpes.fr/movi/people/*{*Sminchisescu,Triggs*}

Abstract

We present a method for recovering 3D human body motion from monocular video sequences based on a robust image matching metric, incorporation of joint limits and non-self-intersection constraints, and a new sample-and-refine search strategy guided by rescaled cost-function covariances. Monocular 3D body tracking is challenging: besides the difficulty of matching an imperfect, highly flexible, self-occluding model to cluttered image features, realistic body models have at least 30 joint parameters subject to highly nonlinear physical constraints, and at least a third of these degrees of freedom are nearly unobservable in any given monocular image. For image matching we use a carefully designed robust cost metric combining robust optical flow, edge energy, and motion boundaries. The nonlinearities and matching ambiguities make the parameter-space cost surface multi-modal, ill-conditioned and highly nonlinear, so searching it is difficult. We discuss the limitations of CONDENSATION*-like samplers, and describe a novel hybrid search algorithm that combines inflated-covariance-scaled sampling and robust continuous optimization subject to physical constraints and model priors. Our experiments on challenging monocular sequences show that robust cost modeling, joint and self-intersection constraints, and informed sampling are all essential for reliable monocular 3D motion estimation.*

Keywords: *3D human body tracking, particle filtering, high-dimensional search, constrained optimization, robust matching.*

1 Introduction

Extracting 3D human motion from natural *monocular* video sequences poses difficult modeling and computation problems:

(*i*) Even a minimal human model is very complex, with at least 30 joint parameters and many more body shape ones, subject to highly nonlinear joint limits and non-self-intersection constraints.

(*ii*) Matching a complex, imperfectly known, self-occluding model to a cluttered scene is inherently difficult. Typical loose clothing only complicates matters.

(*iii*) In contrast to simplified 2D approaches (Cham and Rehg, 1999; Ju et al., 1996) and the multi-camera 3D case (Kakadiaris and Metaxas, 1996; Gavrila and Davis, 1996; Bregler and Malik, 1998; Delamarre and Faugeras, 1999; Plankers and Fua, 2001; Drummond and Cipolla, 2001), the estimation problem is extremely ill-conditioned, with at least 1/3 of the 30+ degrees of freedom remaining very nearly unobservable in any given monocular image. The most important non-observabilities are motions of major body segments in depth (*i.e*. towards or away from the camera — these account for 1/3 of the 3D d.o.f.), but others include rotations of near-cylindrical limbs about their axes, and internal motions of compound joints like the spine or shoulder that are difficult to observe even with 3D data.

(*iv*) In addition to being ill-conditioned, the monocular estimation problem is highly multi-modal. In particular, for any given set of image projections of the 3D joint centers, there are typically some thousands of possible inverse kinematics solutions for the 3D body configuration . Under any reasonable model-image matching cost metric, each kinematic solution produces a corresponding local minimum in configuration space, and correspondence ambiguities only compound this number of minima. In practice, choosing the wrong minimum rapidly leads to mistracking, so reliable tracking requires a powerful multiple hypothesis tracker capable of finding and following a significant number of minima. The development of such a tracker is one of the main contributions of this paper. Some more recent work, not reported here, further enhances tracking reliability by explicitly enumerating the possible kinematic minima (Sminchisescu and Triggs, 2003).

Also note that these four difficulties interact strongly in practice. For example, minor modeling or matching errors tend to lead to large compensatory biases in hard-to-estimate depth parameters, which in turn cause misprediction and tracking failure. Hence, we believe that a successful monocular 3D body tracking system must pay attention to all of them.

Organisation: §1.1 discusses several existing ap-

Published in *Int. J. Robotics Research* special issue on Visual Analysis of Human Movement, 22:371–391, June 2003.

For each body segment, for any given depth for its top (innermost) endpoint, the bottom endpoint can be aligned with its image projection either in a 'sloped forwards' configuration, or in a 'sloped backwards' one. A full body model contains at least 10 main body segments, and hence has at least $2^{10} = 1024$ possible inverse kinematics solutions (sets of forwards/backwards segment configurations). See Lee and Chen (1985) and the empirical confirmations in Sminchisescu and Triggs (2002a,b).

proaches to human articular tracking, explaining why we believe that they are not suitable for the difficult 3D-from-monocular case and informally motivating our new tracker. §2 briefly describes our 3D body model, which includes full 3D occlusion prediction, joint angle limits and body non-self-intersection constraints. §3 discusses our robust model-image matching framework, which combines robust optical flow, edge energy, and motion boundaries. §4 details our hybrid search / tracking scheme, which combines a mixture density propagation tracker with carefully shaped cost-sensitive sampling, with robust constraint-respecting local optimization. §5 briefly describes the local optimization schedule we use to find initial 3D body poses and internal body proportions from model/image joint correspondence input. §7 details some experiments on challenging monocular sequences. These illustrate the need for each of robust cost modeling, joint and self-intersection constraints, and well-controlled sampling plus local optimization. We end the paper with discussions of the effect of the sampling regime on search efficiency (§7) and approximation accuracy (§8), and ideas for future work.

1.1 High-Dimensional Tracking Strategies

Locating good poses in a high-dimensional body configuration space is intrinsically difficult. Three main classes of search strategies exist: **local descent** incrementally improves an existing estimate, *e.g.* using local Newton strategies to predict good search directions (Bregler and Malik, 1998; Rehg and Kanade, 1995; Kakadiaris and Metaxas, 1996; Wachter and Nagel, 1999); **regular sampling** evaluates the cost function at a predefined pattern of points in (a slice of) parameter space, *e.g.* a local rectangular grid (Gavrila and Davis, 1996); and **stochastic sampling** generates random sampling points according to some hypothesis distribution encoding "good places to look", *e.g.* (Deutscher et al., 2000; Sidenbladh et al., 2000). Densely sampling the entire parameter space would in principle guarantee a good solution, but it is infeasible in more than 2–3 dimensions. In 30 dimensions any feasible sample must be extremely sparse, and hence likely to miss significant cost minima. Local descent does at least find a local minimum, but with multimodality there is no guarantee that the globally most representative ones are found. Whichever method is used, effective focusing is the key to high-dimensional search. This is an active research area (Deutscher et al., 2000; Heap and Hogg, 1998; Cham and Rehg, 1999; Merwe et al., 2000), but no existing method can guarantee global minima.

During tracking the search method is applied time-recursively, the starting point(s) for the current search being obtained from the results at the previous time step, perhaps according to some noisy dynamical model. To the (often limited!) extent that the dynamics and the image matching cost are statistically realistic, Bayes-law propagation of a probability density for the true state is possible. For linearized unimodal dynamics and observation models under least squares / Gaussian noise, this leads to Extended Kalman Filtering. For likelihood-weighted random sampling under general multimodal dynamics and observation models, bootstrap filters (Gordon et al., 1993; Gordon and Salmond, 1995) or CONDENSATION (Isard and Blake, 1998) result. In either case various model parameters must be tuned and it sometimes happens that physically implausible settings are needed for acceptable performance. In particular, to control mistracking caused by correspondence errors, selection of slightly incorrect inverse kinematics solutions, and similar model identification errors, visual trackers often require exaggerated levels of dynamical noise. The problem is that even quite minor errors can pull the state estimate a substantial distance from its true value, especially if they persist over several time steps. Recovering from such an error requires a state space jump greater than any that a realistic random dynamics is likely to provide, whereas using an exaggeratedly noisy dynamics provides an easily controllable degree of local randomization that often allows the mistracked estimate to jump back onto the right track. Boosting the dynamical noise does have the side effect of reducing the information propagated from past observations, and hence increasing the local uncertainty associated with each mode. But this is a small penalty to pay for reliable tracking lock, and in any case the loss of accuracy is often minor in visual tracking, where weak dynamical models (*i.e.* short integration times: most of the state information comes from current observations and dynamical details are unimportant) are common.

In summary, in multi-modal problems, sample based Bayesian trackers often get trapped into following incorrect local minima, and some form of explicit local (but not *too* local!) search must be included to rescue them. For trackers operating in this "memoryless step and search" regime, the machinery of Bayes-law propagation is superfluous — the dynamical model is not correct in any case — and it is simpler to think in terms of sequential local search rather than tracking and noisy dynamics. It seems that many, if not most, existing Bayesian trackers in vision operate essentially in this regime, and the current paper is no exception. Hence, we will assume only weak zeroth order dynamical models and use the language of search rather than tracking. But this is largely a matter of terminology, and more elaborate dynamical models are trivial to incorporate if desired.

Many existing human trackers silently inflate the dynamical noise as a local search mechanism, *e.g.* (Cham and Rehg, 1999; Heap and Hogg, 1998; Deutscher et al., 2000). But in each of these papers, it is only one component of

the overall search strategy. The randomization provided by noise inflation is an effective search strategy only for relatively low-dimensional problems, where the samples can cover the surrounding neighborhood fairly densely. In high dimensions, volume increases very rapidly with radius, so any sample that is spread widely enough to reach nearby minima must necessarily be extremely sparse. Hence, the samples are most unlikely to hit the small core of low cost values surrounding another minimum: if they fall into its basin of attraction at all, they are much more likely do so at a high cost point, simply because high cost points are far more common. This is fatal for CONDENSATION-style weighted resampling: high cost points are very unlikely to be resampled, so the new minimum is almost certain to be missed even though an independent track started at the sample would eventually condense to the minimum. The moral is that in high dimensions, random sampling alone does not suffice: some form of local optimization of the samples, or at least a delayed decision about whether they are viable or not, is essential to prevent mistracking. Cham and Rehg (1999); Heap and Hogg (1998) and the current work use explicit descent-based local optimization for this, while (Deutscher et al., 2000) use a simulated annealing like process (which is usually less efficient, although better aligned with the point-based sample-and-evaluate philosophy of pure particle tracking).

The 3D-from-monocular problem has characteristic ill-conditioning associated with depth degrees of freedom, whereas transverse degrees of freedom are directly observable and hence relatively well conditioned. It also has large numbers of kinematic local minima related by motions in depth, in addition to the minima in transversal directions produced by correspondence ambiguities. Hence, we would like to ensure a thorough, perhaps even a preferential, search along the hard-to-estimate depth degrees of freedom. The problem is that the two sets of directions have very different properties and scales. Precisely because they have such similar image appearances, related kinematic minima may cause confusion even if they are separated by significant distances in parameter space, whereas false-correspondence minima only cause confusion if they are relatively nearby. In other words, the natural metric for tracker confusion — and hence for the sampling distribution of the randomized local search — is perceptual image distance, not parameter space distance. This holds notwithstanding the fact that large jumps in configuration (depth) are improbable under natural human dynamics: the tracker may have been gradually mislead over a period of time, and it is essential that it should be able to jump far enough to recover before tracking fails entirely.

This suggests that we need to inflate the dynamical noise preferentially along the depth directions. But these depend strongly on where the model is viewed from, so no constant (configuration or camera-position independent) noise inflation suffices here. The simplest way to adapt the noise to the configuration/camera-position is to estimate the covariance of the posterior likelihood and use this for noise scaling. (In fact, we advocate inflating the *prior* covariance — the previous posterior after dynamics with *physically realistic* noise levels — *i.e.* there should be both realistic dynamics and some degree of deliberate random search). Evaluating covariances might be burdensome in a conventional particle tracking framework where we only had point samples of likelihoods, but we have already seen that some form of local refinement of the samples is practically essential in high dimensions, and efficient local optimizers require (and in the case of quasi-Newton style methods, even provide) information equivalent to covariance estimates.

The emphasize how much difference covariance scaling can make, consider the 32 d.o.f. cost spectrum in fig. 5 on page 17, which has a 2000:1 range of principal standard deviations. For inflation large enough to double the sampling radius along the most uncertain direction (*e.g.*, for a modest search for local minima along this cost valley), a scaling based on uniform dynamical noise would produce a search volume 10^{54} times larger than that of our prior-based one, and an overwhelming fraction of these samples would have extremely high cost and images implausibly different from the source image (see also fig. 1 on page 12). Such wastage factors are clearly untenable. In practice, samplers based on inflating non-covariance-based dynamical noises simply can not sample deeply enough along the most uncertain (depth) directions to find the local minima there, and frequent mistracking is the result.

Finally, given that we are including a component of covariance-scaled but inflated noise expressly as a local search mechanism, what kinds of noise distributions will give the most efficient search? Basically, we need to keep a reasonably large proportion of the samples focused on the current track, while scattering the others fairly widely in the hope of finding other good tracks. Also, volume increases very rapidly with radius in high dimensions, so (even with local optimization) we can not hope to sample densely enough to provide effective search coverage at large inflation factors. It is preferable to choose a moderate inflation level, even though this only provides access to relatively nearby local minima.

In summary, owing to its high dimensionality and the ill-conditioning and multi-modality associated with unobservable depth degrees of freedom, we believe that reliable

Ideally, a subsequent smoothing process would push the corrective jump back in time to where the error first occurred (where the jump presumably becomes small). But whether or not this is done, the likelihood penalty for following an incorrect path arbitrarily far forwards in time is likely to be greater than that for any single corrective jump, bad as this may be.

3D-from-monocular human body tracking requires deliberate sampling (or some other form of local search) in a region shaped by, but significantly larger than, the local state covariance, followed by local optimization of the samples before any resampling step.

1.2 Previous Work

Below we will compare our method to several existing ones, which we briefly summarize here without attempting a full literature review. 3D body tracking from monocular sequences is significantly harder than 2D (Cham and Rehg, 1999; Ju et al., 1996) or multi-camera 3D (Kakadiaris and Metaxas, 1996; Gavrila and Davis, 1996; Bregler and Malik, 1998; Delamarre and Faugeras, 1999; Plankers and Fua, 2001; Drummond and Cipolla, 2001) tracking, and surprisingly few works have addressed it (Deutscher et al., 2000; Sidenbladh et al., 2000; Wachter and Nagel, 1999; Gonglaves et al., 1995; Howe et al., 1999; Brand, 1999).

Deutscher et al. (2000) uses a sophisticated 'annealed sampling' strategy and a cross-over operator (Deutscher et al., 2001) to speed up CONDENSATION. He reports very good results for unconstrained full-body motion, but for his main sequence he uses 3 cameras and a black background to limit the impact of the alternative minima produced by clutter and depth ambiguities. Sidenbladh et al. (2000) uses a similar importance sampling technique with a strong learned prior walking model or a database of motion snippets (Sidenbladh et al., 2002) to track a walking person in an outdoor monocular sequence. Subsequent work (Sidenbladh and Black, 2001) integrates flow, edge and ridge cues using Laplace like error distributions learned from training data, and shows improved upper body tracking for a subject performing planar motion in a cluttered scene, acquired with a moving camera. Our current method uses no motion model — we optimize static poses — but it is true that when they hold, prior motion models are very effective tracking stabilizers. It is possible, but expensive, to track using a bank of motion models (Blake et al., 1999). Partitioned sampling (MacCormick and Isard, 2000) is another notable sampling technique for articulated models, under certain labeling assumptions (MacCormick and Isard, 2000; Deutscher et al., 2000).

Several authors address the difficulty that the sampling-based searches of pure particle filtering converge rather slowly to modes (Pitt and Shephard, 1997; Heap and Hogg, 1998; Cham and Rehg, 1999; Merwe et al., 2000; Choo and Fleet, 2001), especially when the observation likelihood peaks deep in the tail of the prior. This is especially problematic in high dimensions, where prohibitively long sampling runs are often required for convergence. Heap and Hogg (1998); Cham and Rehg (1999); Merwe et al. (2000) all combine CONDENSATION-style sampling with either local optimization or Kalman filtering, while Pitt and Shephard (1997) samples discretely using the current observation likelihood (and not the transition prior). The visual trackers of Heap and Hogg (1998) and Cham and Rehg (1999) combine CONDENSATION-style sampling with least-squares optimization, but they only consider the simpler (and much better conditioned) case of 2D tracking. Cham & Rehg combine their heuristic 2D Scaled Prismatic Model (SPM) body representation with a first order motion model and a piecewise Gaussian resampling method for the CONDENSATION step. The Gaussian covariances are estimated from the Gauss-Newton approximation at the fitted optima, but the search region widths are controlled by the traditional method of adding a large dynamical noise ((Cham and Rehg, 1999) section 3.2).

Choo and Fleet (2001) use a stick model (without any shape model) for which 3D-2D joint to image correspondences from motion capture data are available and propose a (gradient-based) Hybrid Monte Carlo sampler that is more efficient than (point-based) CONDENSATION. The method provides more efficient local descent towards the minima, but it is still prone to trapping in sub-optimal local minima.

Wachter and Nagel (1999) use articulated kinematics and a shape model built from truncated cones, and estimate motion in a monocular sequence, using edge and intensity (optical flow) information using an extended Kalman filter. Anatomical joint limits are enforced at the level of the filter prediction, but not during the update step, where they could be violated. They show experiments in an unconstrained environment for a subject wearing normal clothing, tracking motion parallel with the image plane using articulated models with 10-15 d.o.f.

Both Brand (1999) and Howe et al. (1999) pose 3D estimation as a learning and inference problem, assuming that some form of 2D tracking (stick 2D model positions or silhouettes) is available over an entire time-series. Howe et al. (1999) learn Gaussian distributions over short "snippets" of observed human motion trajectories, then use these as priors in an EM-based Bayesian MAP framework to estimate new motions. Brand (1999) learns a HMM with piecewise linear states and solves for the MAP estimate using an entropy minimization framework. As presented, these methods are basically monomodal so they can not accommodate multiple trajectory interpretations, and they

The covariance estimates of nonlinear least-squares optimizers as used by (Heap and Hogg, 1998; Cham and Rehg, 1999) are not robust to model/image matching errors and incorrect (*i.e.* biased) for natural image statistics that have highly non-Gaussian shape with high kurtosis and long tails (Zhu and Mumford, 1997; Sidenbladh and Black, 2001). We use an observation likelihood and a robust local continuous optimizer based on heavy tail error distributions (see §3.1 and §4.1) to address these problems.

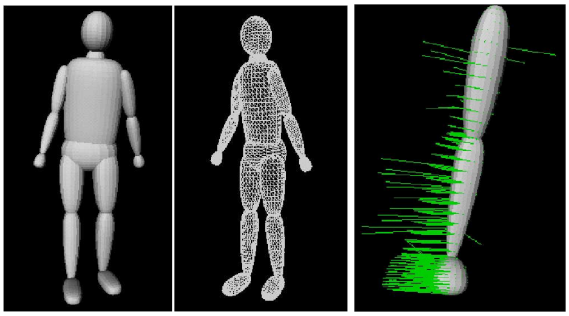


Figure 2: Different body models using for tracking (a,b,c). In (c) the prediction errors for a model configuration are also plotted (per node, for a contour and intensity cost function, see text).

also rely heavily on their learned-prior temporal models to stabilize the tracking. Nevertheless, they provide a powerful higher-level learning component that is complementary to the framework proposed in this paper.

2 Human Body Model

Our human body model (fig. 2a,b,c) consists of a kinematic 'skeleton' of articulated joints controlled by angular **joint parameters** $\mathbf{x}_a$, covered by 'flesh' built from superquadric ellipsoids with additional tapering and bending parameters (Barr, 1984). A typical model has around 30 joint parameters, plus 8 **internal proportion** parameters $\mathbf{x}_i$ encoding the positions of the hip, clavicle and skull tip joints, plus 9 **deformable shape** parameters for each body part, gathered into a vector $\mathbf{x}_d$. A complete model can be encoded as a single large parameter vector $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_d, \mathbf{x}_i)$. During tracking we usually estimate only joint parameters, but during initialization the most important internal proportions and shape parameters are also optimized, subject to a soft prior based on standard humanoid dimensions obtained from Group (2002) and updated using collected image evidence. This model is far from photorealistic, but it suffices for high-level interpretation and realistic occlusion prediction, offering a good trade-off between computational complexity and coverage.

The model is used as follows. Superquadric surfaces are discretized as meshes parameterized by angular coordinates in a 2D topological domain. Mesh nodes $\mathbf{u}_i$ are transformed into 3D points $\mathbf{p}_i = \mathbf{p}_i(\mathbf{x})$ and then into predicted image points $\mathbf{r}_i = \mathbf{r}_i(\mathbf{x})$ using composite nonlinear transformations:

$$\mathbf{r}_i(\mathbf{x}) \;=\; P(\mathbf{p}_i(\mathbf{x})) \;=\; P(A(\mathbf{x}_a, \mathbf{x}_i, D(\mathbf{x}_d, \mathbf{u}_i))) \quad (1)$$

where D represents a sequence of parametric deformations that construct the corresponding part in its own reference frame, A represents a chain of rigid transformations that map it through the kinematic chain to its 3D position, and P represents perspective image projection. During model estimation, robust prediction-to-image matching cost metrics are evaluated for each predicted image feature $\mathbf{r}_i$, and the results are summed over all features to produce the image contribution to the overall parameter space cost function. We use both direct image-based cost metrics such as robustified normalized edge energy, and extracted feature based ones. The latter associate the predictions $\mathbf{r}_i$ with one or more nearby image features $\bar{\mathbf{r}}_i$ (with additional subscripts if there are several matches). The cost is then a robust function of the prediction errors $\Delta\mathbf{r}_i(\mathbf{x}) \;=\; \bar{\mathbf{r}}_i - \mathbf{r}_i(\mathbf{x})$.

3 Problem Formulation

We aim towards a probabilistic interpretation and optimal estimates of the model parameters by maximizing the total probability according to Bayes rule:

$$p(\mathbf{x}|\bar{\mathbf{r}}) \;\propto\; p(\bar{\mathbf{r}}|\mathbf{x})\, p(\mathbf{x}) \;=\; \exp\left(-\textstyle\sum_i e(\bar{\mathbf{r}}_i|\mathbf{x})\right) p(\mathbf{x}) \quad (2)$$

where $e(\bar{\mathbf{r}}_i|\mathbf{x})$ is the cost density associated with the observation of node i and $p(\mathbf{x})$ is a prior on model parameters. In our MAP approach, we discretize the continuous problem and attempt to minimize the negative log-likelihood for the total posterior probability, expressed as the following cost function:

$$f(\mathbf{x}) \;=\; -\log(p(\bar{\mathbf{r}}|\mathbf{x})\, p(\mathbf{x})) \quad (3)$$

$$\;=\; -\log p(\bar{\mathbf{r}}|\mathbf{x}) - \log p(\mathbf{x}) \;=\; f_o(\mathbf{x}) + f_p(\mathbf{x}) \quad (4)$$

3.1 Observation Likelihood

Whether continuous or discrete, the search process depends critically on the observation likelihood component of the parameter space cost function. Besides smoothness properties, the likelihood should be designed to limit the number of spurious local minima in parameter space. Our method employs a combination of robust edge and intensity information on top of a multiple assignment strategy based on a weighting scheme that focuses attention towards motion boundaries. Our likelihood term is also based on robust (heavy-tailed) error distributions. Note that both robustly extracted image cues and robust parameter space estimation are used: the former provides "good features to track", while the latter directly addresses the model-image association problem.

Robust Error Distributions: Robust parameter estimation can be viewed as the choice of a realistic total likelihood model for the combined inlier and outlier distributions for the observation. We model the total likelihood in terms of robust radial terms ρ_i, where $\rho_i(s)$ can be any increasing function with $\rho_i(0) \;=\; 0$ and $\frac{d}{ds}\rho_i(0) \;=\; \frac{\nu}{\sigma^2}$. These

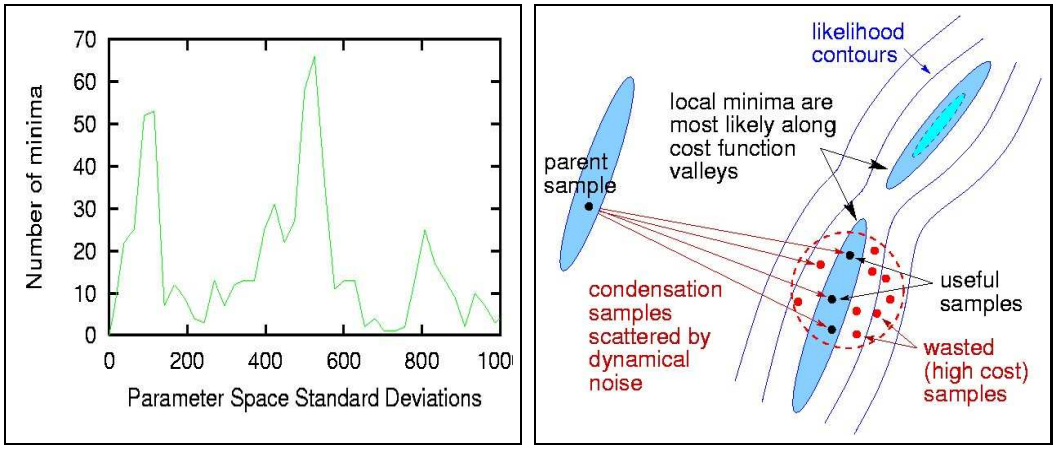


Figure 1: (a) Typical parameter space minima distribution measured with respect to an arbitrary minimum. Notice that the minima are far from each other in parameter space so wide sampling is necessary to find them. However, boosting the dynamics by sampling from the transition prior (as in particle filtering) leads to inefficiencies (b).

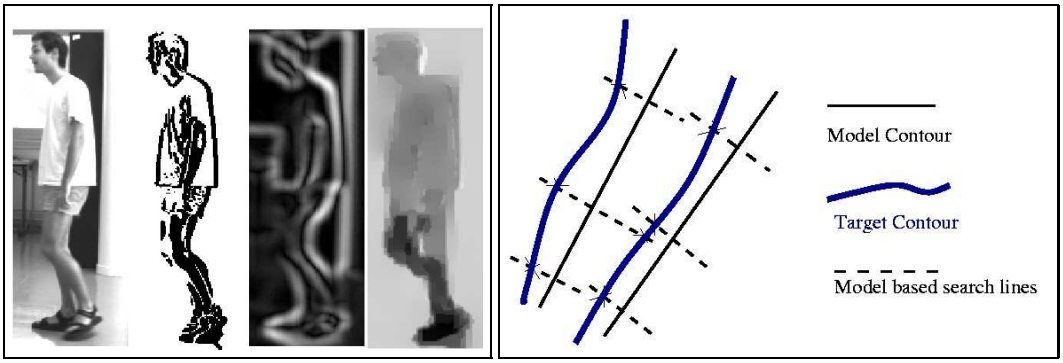


Figure 3: Examples of our robust low-level feature extraction: original image (a), motion boundaries (b), intensity-edge energy (c), robust horizontal flow field (d) and the model-based edge matching process (e). Multiple edge matches found along individual search lines (model projected contour normals) are fused using a probabilistic assignment strategy (see text).

model error distributions corresponding to a central peak with scale σ, and a widely spread background of outliers ν. Here we used the 'Lorentzian' $\rho_i(s,\sigma) = \nu \log(1 + \frac{s}{\sigma^2})$ and 'Leclerc' $\rho_i(s,\sigma) = \nu(1 - \exp(-\frac{s}{\sigma^2}))$ robust error potentials.

The cost for the observation i, expressed in terms of corresponding model prediction is $e(\bar{\mathbf{r}}_i|\mathbf{x}) = \frac{1}{N\nu} e_{ui}(\mathbf{x})$, where N is the total number of model nodes, $\mathbf{W}_i$ is a positive definite weighting matrix associated to the assignment i, and:

$$e_i(\mathbf{x}) = \begin{cases} \frac{1}{2}\rho_i(\mathbf{\Delta r}_i(\mathbf{x})\, \mathbf{W}_i\, \mathbf{\Delta r}_i(\mathbf{x})^\top) & \text{if } i \text{ is assigned} \\ \nu_{bf} = \nu & \text{if back-facing} \\ \nu_{occ} = k\nu, \;\; k > 1 & \text{if occluded} \end{cases} \quad (5)$$

The robust observation likelihood contribution is thus:

$$f_o(\mathbf{x}) = -\log p(\bar{\mathbf{r}}|\mathbf{x}) \quad (6)$$
$$= f_a(\mathbf{x}) + N_{bf}\, \nu_{bf} + N_{occ}\, \nu_{occ} \quad (7)$$

where $f_a(\mathbf{x})$ represents the term associated with the image assigned model nodes, while N_{occ} and N_{bf} are the numbers of occluded and back-facing (self-occluded) model nodes.

Notice that occluded model predictions are not simply ignored. They contribute a constant penalty to the overall observation likelihood. This is necessary in order to build likelihoods that preserve their response properties under occlusion and viewpoint change. For instance, good fits from both frontal and side views should ideally have similar peak responses, but it is clear that the number of occluded model points is in general larger in a side view than in a frontal one. This can lead to down-weighting of peaks for side views if only the visible nodes are taken

into account. An additional difficulty arises, for example, in cases where the legs pass each other (in a side-view) and the model 'locks' both of its legs onto the same image leg. To avoid such situations, we include all of the model nodes when fusing the likelihood, but we slightly penalize occluded ones in order to make them less attractive. A way to choose the occlusion penalty ν is to fit the model to the data and compute an approximate error per node. By using a slightly higher value for occluded nodes, we make them more attractive than a bad fit but less attractive than other non-occluded states that can exist in the neighborhood of the parameter space. We find this heuristic gives good results in practice, although a more rigorous treatment of occlusion would be desirable in the general case. At present, this is computationally too expensive, but interesting approximations can be found in MacCormick and Blake (1998).

Cue Integration and Assigned Image Descriptors: We use both edge and intensity features in our cost function (see Sminchisescu (2002b) for details). For edges, the images are smoothed with a Gaussian kernel, contrast normalized, and a Sobel edge detector is applied. For intensities, a robust multi-scale optical flow method based on Black and Anandan (1996) implementation gives both a flow field and an associated outlier map (see fig. 3b). The outlier map is processed similar to edges, to obtain a smooth 2D potential field S_p. It conveys useful information about the motion boundaries and is used to weight the significance of edges (see fig. 3b). We typically use diagonal weighting matrices $\mathbf{W}_i$, associated with the predicted feature $\mathbf{r}_i$ and corresponding matched observation $\bar{\mathbf{r}}_i$, of the form $\mathbf{W}_i(\mathbf{r}_i) = 1 - kS_p(\bar{\mathbf{r}}_i)$, where k is a constant that controls the emphasis and confidence in the motion boundary estimation. (The smoothed motion boundary image is a real image with values between 0 and 1 as in fig. 3b. For instance, $k = 0$ will weight all the edges uniformly, while $k = 1$ will entirely exclude the edge responses that are not on motion boundaries). In practice, we found that values of k in the range $k = 0.2$–0.4 worked well. For visible nodes on model occluding contours ($\mathcal{O}$), we perform line search along the normal and retain all possible assignments within the search window (see fig. 3e), weighting them by their importance qualified by the motion boundary map $\mathbf{W}$. For visible model nodes lying inside the object ($\mathcal{I}$), we use the correspondence field derived from the robust optical flow at their corresponding image prediction. This acts as a residual measurement error at each visible model node (see Sminchisescu (2002b) for details). The assigned data term (6) thus becomes:

$$f_a(\mathbf{x}) = \frac{1}{2} \sum_{i \in \mathcal{O},\, e \in \mathcal{E}_i} \rho_{i_e}(\Delta \mathbf{r}_{i_e}(\mathbf{x})\, \mathbf{W}_{i_e}\, \Delta \mathbf{r}_{i_e}(\mathbf{x})^\top) \quad (8)$$

$$+ \frac{1}{2} \sum_{j \in \mathcal{I}} \rho_{j_f}(\Delta \mathbf{r}_{j_f}(\mathbf{x})\, \mathbf{W}_{j_f}\, \Delta \mathbf{r}_{j_f}(\mathbf{x})^\top) \quad (9)$$

where the subscripts "i_e" denote multiple edges $\mathcal{E}_i$ assigned to model prediction i, and "j_f" denote the flow term assigned to model prediction j.

3.2 Model Priors

The complete prior penalty over model parameters is a sum of negative log likelihoods $f_p = f_{an} + f_s + f_{pa}$ corresponding to the following prior densities p_{an}, p_s, p_{pa}:

Anthropometric data p_{an}: The internal proportions for a standard humanoid (based on statistical measurements) are collected from (Group, 2002) and used effectively as a Gaussian prior, $p_{an} = \mathcal{N}(\boldsymbol{\mu}_{an}, \boldsymbol{\Sigma}_{an})$, to estimate a concrete model for the subject to be tracked. Left-right symmetry of the body is assumed: only "one side" of the internal proportions parameters are estimated while collecting image measurements from the entire body.

Parameter stabilizers p_s: Certain modeling details are far more important than one might think. For example, it is impossible to track common turning and reaching motions unless the clavicle joints in the shoulder are modeled accurately. However, these parameters have fairly well defined equilibrium positions and leaving them unconstrained would often lead to ambiguities that produce nearly singular (flat) cost surfaces. We control these hard-to-estimate parameters with long-tailed "sticky prior" stabilizers scaling their Gaussian equilibria, $p_s = \mathcal{N}(\boldsymbol{\mu}_s, \boldsymbol{\Sigma}_s)$. This ensures that in the absence of strong observations, the parameters are constrained to lie near their default values, whereas stronger observations can "unstick" them from the defaults and effectively turn off the prior.

Anatomical joint angle limits C_{bl}: 3D consistency requires that the values of joint angles evolve within anatomically consistent intervals. Also, when estimating internal body proportions during initialization, we ensure that they remain within a certain range of deviation from the standard humanoid (typically 10%). We model this with a set of inequalities of the form $C_{bl} \cdot \mathbf{x} < 0$, where C_{bl} is a 'box-limit' constraint matrix.

Body part interpenetration avoidance p_{pa}: Physical consistency requires that different body parts do not interpenetrate during estimation. We avoid this by introducing repulsive potentials that decay rapidly outside the surface of each body part, $f_{pa} = \exp(-f(\mathbf{x})\,|f(\mathbf{x})|^{p-1})$, where $f(\mathbf{x}) < 0$ defines the interior of the part and p controls the decay rate.

This is particularly effective when combined with the Covariance Scaled Sampling (CSS) algorithm presented in §4. Loss of visibility of certain body parts leads to increased uncertainty in related parameters, and CSS automatically ensures broader sampling in those parameter space regions.

3.3 Distribution Representation

We represent parameter space distributions as sets of separate modes $m_i \in \mathcal{M}$, each having an associated overall probability, mean and covariance matrix $m_i = (\mu_i, \Sigma_i, c_i)$. These can be viewed as Gaussian mixtures. Cham & Rehg (Cham and Rehg, 1999) also use multiple Gaussians, but they had to introduce a special piecewise representation as their modes seem to occur in clusters after optimization. We believe that this is an artifact of their cost function design. In our case, as the modes are the result of robust continuous optimization, they are necessarily either separated or confounded. Our 3D-from-monocular application also requires a more effective sampling method than the 2D one of Cham and Rehg (1999), as explained in §4.2.

3.4 Temporal Propagation

Equation 2 reflects the search for the model parameters in a static image, under likelihood terms and model priors but without a temporal or initialization prior. For temporal observations $\mathbf{R}_t = \{\bar{\mathbf{r}}_1, \bar{\mathbf{r}}_2 ..., \bar{\mathbf{r}}_t\}$, and sequence of states $\mathbf{X}_t = \{\mathbf{x}_1, \mathbf{x}_2 ..., \mathbf{x}_t\}$, the posterior distribution over model parameters becomes:

$$p(\mathbf{x}_t|\mathbf{R}_t) \propto p(\bar{\mathbf{r}}_t|\mathbf{x}_t)\, p(\mathbf{x}_t) \int_{\mathbf{x}_{t-1}} p(\mathbf{x}_t|\mathbf{x}_{t-1})\, p(\mathbf{x}_{t-1}|\mathbf{R}_{t-1}) \tag{10}$$

where $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ is a dynamical prior and $p(\mathbf{x}_{t-1}|\mathbf{R}_{t-1})$ is the prior distribution from $t-1$. Together they form the temporal prior $p(\mathbf{x}_t|\mathbf{R}_{t-1})$ for initializing the static image search (2).

4 Search Algorithm

Our parameter search technique combines robust constraint-consistent local optimization with a more global discrete sampling method.

4.1 Mode Seeking using Robust Constrained Continuous Optimization

The cost function is a negative log likelihood. In order to optimize a sample $\mathbf{x}$ to find the center of its associated likelihood peak, we employ an iterative second order robust constrained local optimization procedure. At each iteration, the log-likelihood gradients and Hessians of the observations and the soft priors are assembled from (3):

$$\mathbf{g} = \frac{\mathbf{d}f}{\mathbf{dx}} = \mathbf{g}_o + \nabla f_{an} + \nabla f_s + \nabla f_{pa} \tag{11}$$

$$\mathbf{H} = \frac{\mathbf{d}^2 f}{\mathbf{dx}^2} = \mathbf{H}_o + \nabla^2 f_{an} + \nabla^2 f_s + \nabla^2 f_{pa} \tag{12}$$

For local optimization we use a second order trust region method, where a descent direction is chosen by solving the regularized subproblem (Fletcher, 1987):

$$(\mathbf{H} + \lambda \mathbf{W})\, \delta\mathbf{x} \;=\; -\mathbf{g} \quad \text{subject to} \quad C_{bl} \cdot \mathbf{x} < 0 \tag{13}$$

where $\mathbf{W}$ is a symmetric positive definite damping matrix and λ is a dynamically chosen weighting factor. Joint limits C_{bl} are handled as hard bound constraints in the optimizer, by projecting the gradient onto the currently active (*i.e.* currently unlimited) variables. The joint constraints change the character of the cost function and the minima reached very significantly. Fig. 4 plots a 1D slice through the constrained cost function together with a second order Taylor expansion of the unconstrained cost. Owing to the presence of the bounds, the cost gradient is nonzero (orthogonal to the active constraints) at the constrained minimum. The unconstrained cost function is smooth, but the constrained one changes gradient abruptly when a constraint is hit, essentially because the active-set projection method changes the motion direction to maintain the constraint.

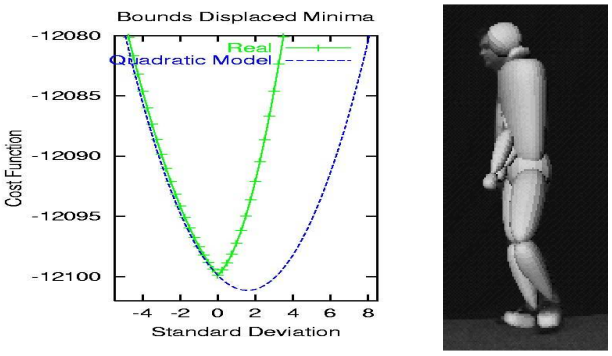


Figure 4: (a) Displaced minimum due to joint limits constraints, (b) Joint limits without body non-self-intersection constraints do not suffice for physical consistency.

4.2 Covariance Scaled Sampling

Although representations based on propagating multiple modes, hypotheses or samples tend to increase the robustness of model estimation, the great difficulty with high-dimensional distributions is finding a sampleable proposal

In practice, at any given time step we work on a negative log-likelihood 'energy' function that is essentially static, being based on both the current observation likelihood and the parameter space priors, as in (3) on page 11. The samples from the temporal prior $p(\mathbf{x}_t|\mathbf{R}_{t-1})$ are used as initialization seeds for local energy minimization. The different minima found will represent the components of the posterior mixture representation.

'Soft' means that these terms are part of the cost surface, whereas 'hard' constraints such as joint limits restrict the range of variation of their corresponding parameters.

density that often hits their **typical sets** — the areas where most of their probability mass is concentrated. Here we develop a proposal density based on local parameter estimation uncertainties. The local sample optimizations give us not only local modes, but also their (robust, constraint consistent) Hessians and hence estimates of the local posterior parameter estimation uncertainty at each mode.

The main insight is that alternative cost minima are most likely to occur along local valleys in the cost surface, *i.e.* along highly uncertain directions of the covariance. It is along these directions that cost-modeling imperfections and noise, and 3D nonlinearities and constraints, have the most likelihood of creating multiple minima, as the cost function is shallowest and the 3D movements are largest there. This is particularly true for monocular 3D estimation, where the covariance is unusually ill-conditioned owing to the many poorly observable motion-in-depth d.o.f. Some examples of such multimodal behavior along high covariance eigen-directions are given in fig. 7. Also, it is seldom enough to sample at the scale of the estimated covariance. Samples at this scale almost always fall back into the same local minimum, and significantly deeper sampling is necessary to capture nearby but non-overlapping modes lying further up the valley. Hence, we sample according to rescaled covariances, typically scaling by a factor of 8 or so. Finally, one can sample either randomly, or according to a regular pattern. For the experiments showed here, we use random sampling using CSS with Gaussian tails. Fig. 6 summarizes the resulting covariance-scaled search method.

Given the explanations above, we must implement the following steps:

(*i*) Generate fair samples from a prior with known modes. This is easy. In our case we propagate Gaussian mixtures can be used as importance sampling distributions, and correction weighting is readily performed. Mixtures provide a compact, explicitly multi-modal representation and accurate localization, advantages emphasized by Heap and Hogg (1998) and Cham and Rehg (1999) (§5.1). However, both papers use sampling stages based on the unmodified process model (*i.e.* dynamics with fixed, near-spherical noise), which therefore have trapping and sample wastage problems analogous to CONDENSATION.

(*ii*) Recover new modes of a distribution for which only *some* of the modes are known. This is significantly more difficult. A-priori, the distribution of unknown modes is not available, nor are the boundaries of the basins of attraction of the existing modes (in order to find their neighbors). Also, such likelihood peaks are often well-separated in configuration space (*e.g.* the forwards/backwards flipping ambiguities for human pose, or the cascades of incorrect matches when a model limb is assigned to the incorrect side of an image limb). For typical distributions of minima in parameter space and in cost, see fig. 13 on page 22 and the results in table 1 on page 23. For well-separated peaks, sampling based purely on the known (and potentially incomplete) ones is inadequate, as most of the samples will simply fall back into the peaks they arose from. So broader sampling is necessary, but it is also important to focus the samples in relatively low cost regions (see also fig. 1). To achieve this we propose to use the local cost surface to shape a broad sampling distribution. As expected on theoretical grounds, this turns out to give significantly improved results for CSS (for sample cost median, number of minima found, their cost) than competing methods based on either pure prior-based sampling or prior-based sampling plus spherical 'dynamical' noise (see table 1 on page 23).

(*iii*) Sample a prior under dynamic observations but without making restrictive assumptions on the motion of its peaks. In this case the modes from time $t-1$ are available, and it is critical that the sampling procedure cover the

Related variable metric ideas can be found in global optimization, in the context of continuous annealing (Vanderbilt and Louie, 1984) and have been applied by Black (1992) to low-dimensional (2D) optical flow computation.

A sample is optimized to convergence to obtain the corresponding mode. The Hessian matrix at the convergence mode gives the principal curvature directions and magnitude around the mode and its inverse gives the covariance matrix, reflecting the cost local uncertainty structure. The Hessian is estimated by the algorithm §4.1 during optimization (using (11)), and the covariance is readily obtained from there.

In part this is due to imperfect modeling, which easily creates biases greater than a few standard deviations, particularly in directions where the measurements are weak. Also, one case in which multiple modes are likely to lie so close together in position and cost that they cause confusion is when a single mode fragments due to smooth evolutions of the cost surface. In this case, singularity ('catastrophe') theory predicts that generically, exactly two modes will arise (bifurcation) and that they will initially move apart very rapidly (at a speed proportional to $1/\sqrt{t}$). Hence, it is easy for one mode to get lost if we sample too close to the one we are tracking.

For efficiency purposes, an implementation could sample regularly, in fact only along lines corresponding to the lowest few covariance eigen-directions. Although this gives a very sparse sampling indeed, this is an avenue that can be explored in practice.

There are at least two ways to obtain a mixture. One is by clustering a set of posterior samples generated, *e.g.*, by CONDENSATION updates. This may produce centers that are not necessarily well-separated, and that may not actually reflect the true modes of the posterior owing to sampling artifacts. Another possibility, followed here, is to optimize the samples locally. In this case the modes found are true local peaks that are, necessarily, either separated or confounded.

Several metrics exist for assessing the efficiency of particle filters (Liu, 1996; MacCormick and Isard, 2000). The 'survival diagnostic' (also called 'effective sample size') measures how many particles will survive a resampling operation. If the weights are unbalanced very few may survive, thus reducing search diversity. But balanced weights do not imply that all peaks have been well explored: samples trapped in a single mode have reasonably well-balanced weights. The same criticism applies to the 'survival rate'. This tries to characterize the ratio of the volume of support of the posterior to that of the prior. Low values suggest that the filter may produce inaccurate density estimates, but again, trapping leaves the survival rate reasonably high.

peaks of the observation likelihood in the next time step t. This means that samples should be generated in the basins of attraction of the density peaks *after* applying the dynamical update. In the absence of knowledge about the peaks' motion (*i.e*. known system dynamics), we exploit the local uncertainty structure in the distribution, and shape the search region based on it. Again, broader sampling is necessary, as the tracked object moves between frames. Also, as explained above, the mode tracking process is not one-to-one. New modes might emerge or split under the effect of increased uncertainty, and it is important that the sampling process does not miss such events by sampling too close to a given mode core, which may both move and split between two temporal observations. Our quantitative results in §6 directly support such findings *e.g*. for mode splitting reflecting bi-modality generated by locally-planar versus in-depth motion explanations (see below).

In this paper, we have not used specific motion models as we want to be able to track general human motions (see for instance, the sequences given in fig. 9–11). For the experiments shown in the next section, we used trivial driftless diffusion dynamics, so CSS has to account for local uncertainty and sample widely enough to cover moving peaks. One could also use constant velocity dynamics, or more sophisticated learned motion models such as walking Rohr (1994); Deutscher et al. (2000); Sidenbladh et al. (2000). When they hold, such models can significantly stabilize tracking, but note that they often turn out to be misleading, *e.g*. when the subject makes unexpected motions like turning or switching activities.

To build up intuition about the shape of our cost surface, we studied it empirically by sampling along uncertain covariance directions (in fact eigenvectors of the covariance matrix), for various model configurations. With our carefully selected image descriptors, the cost surface is smooth apart from the apparent gradient discontinuities caused by active-set projection at joint constraint activation points. Hence, our local optimizer reliably finds a local minimum. We find that multiple modes do indeed occur for certain configurations, usually separated by cost barriers that a classical (uninflated) sampling strategy would have difficulty crossing. For example, fig. 7 shows the two most uncertain modes of the fig. 9 human tracking sequence at times 0.8 s and 0.9 s. (These are minima only within the sampled slice of parameter space, but they do lie in the attraction zones of full parameter space minima). Secondary minima like those shown here occur rather often, typically for one of two reasons. The first is incorrect registration and partial loss of track when both edges of a limb model are attracted to the same image edge of the limb. This is particularly critical when there is imperfect body modeling and slightly misestimated depth. The second occurs when the character of a motion in depth is misinterpreted. Image registration is maintained until the incorrect 3D interpretation becomes untenable, at which point recovery is difficult. This situation occurs in fig. 7 (see also fig. 12). Identifying and tracking such ambiguous behaviors is critical, as incorrect depth interpretations quickly lead to tracking failure.

Fig. 8a shows some typical slices along cost eigendirections at much larger scales in parameter space. Note that we recover the expected robust shape of the matching distribution, with some but not too many spurious local minima. This is crucial for efficiency and robustness, as the tracker can only follow a limited number of possible minima.

5 Model Initialization

Our tracker starts with a set of initial hypotheses produced by a model initialization process. Correspondences need to be specified between model joint locations and approximate joint positions of the subject in the initial image, and a non-trivial optimization process is run to estimate certain body dimensions and the initial 3D joint angles. Previous approaches to single-view model initialization (Taylor, 2000; Barron and Kakadiaris, 2000) do not fully address the generality and consistency problems, failing to enforce the joint limit constraints, and assuming either restricted camera models or restricted human poses in the image. An algorithm like the one we propose could also probably be bootstrapped using estimates of 2D joint positions derived from learned models of silhouette appearance (Rosales and Sclaroff, 2000).

For stability, parameters are initialized in three stages, each based on the formulation described in §4.1. Hard joint limits are enforced at all stages by the constrained optimization procedure, and corresponding parameters on the left and right sides of the body are held equal, whereas measurements are collected from the entire body (see below). The first stage estimates joint angles $\mathbf{x}_a$, internal proportions $\mathbf{x}_i$ and a few simple shape $\mathbf{x}_d$ parameters, subject to the given 3D to 2D joint correspondences and prior intervals on the internal proportions and body part sizes. The second stage uses both the given joint correspondences and the local contour signal from image edges to optimize the remaining volumetric body parameters (limb cross-sections and their tapering parameters $\mathbf{x}_d$) while holding the other parameters fixed. Finally, we refine the full model ($\mathbf{x}$) using similar image information to the second stage. The covariance matrix corresponding to the final estimate is used to generate an initial set of hypotheses, which are propagated in time using the algorithm described in §4. While the process is heuristic, it gives a balance between stability and flexibility. In practice we find that enforcing the joint constraints, mirror information and prior bounds on the variation of body parameters gives far more stable and satisfactory results. However, with monocular images,

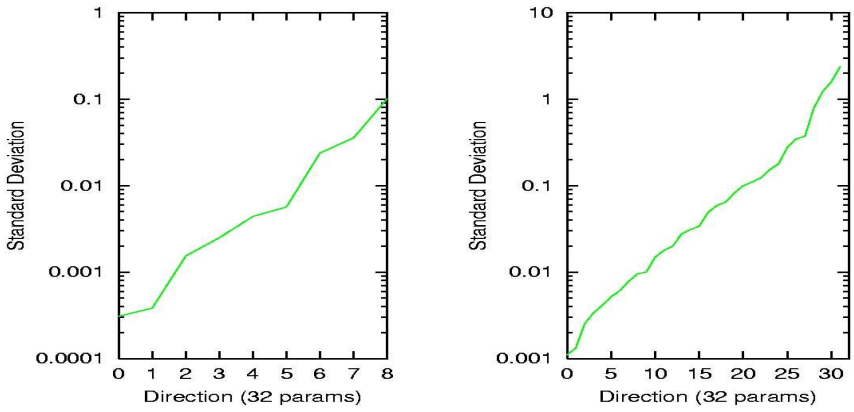


Figure 5: Typical covariance eigenvalue spectra plotted on a logarithmic scale, for a local minimum. $\sigma_{\max}/\sigma_{\min}$ is 350 for the 8 d.o.f. arm model, and 2000 for the 32 d.o.f. body one.

From the 'old' mixture prior $p(\mathbf{x}_{t-1}|\mathbf{R}_{t-1}) = \sum_{i=1}^{K} \pi_i^{t-1}\mathcal{N}(\mu_i^{t-1}, \Sigma_i^{t-1})$, at time $t-1$, build 'new' mixture posterior $p(\mathbf{x}_t|\mathbf{R}_t) = \sum_{i=1}^{K} \pi_i^{t}\mathcal{N}(\mu_i^{t}, \Sigma_i^{t})$, at time t, as follows:

1. Build covariance scaled proposal density $p_{t-1}^* = \sum_{i=1}^{K} \pi_i^{t-1}\mathcal{P}(\mu_i^{t-1}, \Sigma_i^{t-1})$. For the experiments we have used Gaussian tails. The covariance scaled Gaussian component proposals are $\mathcal{P} = \mathcal{N}(\mu_i^{t-1}, s\Sigma_i^{t-1})$ with s=4–14 in our experiments.

2. Generate components of the posterior at time t by sampling from p_{t-1}^* as follows. Iterate over $j = 1...N$ until the desired number of samples N are generated:
2.1. Choose component i from p_{t-1}^* with probability π_i^{t-1}.
2.2. Sample from $\mathcal{P}(\mu_i^{t-1}, \Sigma_i^{t-1})$ to obtain $\mathbf{s}_j$.
2.3. Optimize $\mathbf{s}_j$ over the observation likelihood at time t, $p(\mathbf{x}|\bar{\mathbf{r}}_t)$ defined by (2), using the local continuous optimization algorithm (§4.1). The result is the parameter space configuration at convergence μ_j^t, and the covariance matrix $\Sigma_j^t = \mathbf{H}(\mu_j^t)^{-1}$. If the μ_j^t mode has been previously found by a local descent process, discard it (For notational clarity, without any loss of generality, consider all the modes found are different).

3. Construct an un-pruned posterior for time t as: $p_t^u(\mathbf{x}_t|\mathbf{R}_t) = \sum_{j=1}^{N} \pi_j^t\mathcal{N}(\mu_j^t, \Sigma_j^t)$ where $\pi_j^t = \frac{p(\mu_j^t|\bar{\mathbf{r}}_t)}{\sum_{j=1}^{N} p(\mu_j^t|\bar{\mathbf{r}}_t)}$.

4. Prune the posterior p_t^u to keep the best K components with highest probability π_j^t (rename indices $j = 1...N$ into the set $k = 1...K$) and renormalize the distribution as follows: $p_t^p(\mathbf{x}_t|\mathbf{R}_t) = \sum_{k=1}^{K} \pi_k^t\mathcal{N}(\mu_k^t, \Sigma_k^t)$ where $\pi_k^t = \frac{p(\mu_k^t|\bar{\mathbf{r}}_t)}{\sum_{j=1}^{K} p(\mu_j^t|\bar{\mathbf{r}}_t)}$.

5. For each mixture component $j = 1...K$ in p_t^p, find the closest prior component i in $p(\mathbf{x}_{t-1}|\mathbf{R}_{t-1})$, according to a Bhattacharyya distance $\mathcal{B}_{ij}(\mu_i^{t-1}, \Sigma_i^{t-1}, \mu_j^t, \Sigma_j^t) = (\mu_i^{t-1} - \mu_j^t)^T \left[\frac{\Sigma_i^{t-1}+\Sigma_j^t}{2}\right]^{-1} (\mu_i^{t-1} - \mu_j^t) + \frac{1}{2}\log \frac{\left|\frac{\Sigma_i^{t-1}+\Sigma_j^t}{2}\right|}{\sqrt{|\Sigma_i^{t-1}||\Sigma_j^t|}}$.
Recompute $\pi_j^t = \pi_j^t * \pi_i^{t-1}$. Discard the component i of $p(\mathbf{x}_{t-1}|\mathbf{R}_{t-1})$ from further consideration.

6. Compute the posterior mixture $p(\mathbf{x}_t|\mathbf{R}_t) = \sum_{k=1}^{K} \pi_k^t\mathcal{N}(\mu_k^t, \Sigma_k^t)$ where $\pi_k^t = \frac{\pi_k^t}{\sum_{j=1}^{K} \pi_j^t}$.

Figure 6: The steps of our covariance-scaled sampling algorithm.

the initialization always remains ambiguous and highly uncertain in some parameter space directions, especially under 3D-2D joint correspondence data. In our case, we employ a suitable coarse pose initialization and use the above process for fine refinement, but if available, one could fuse pose information from multiple images.

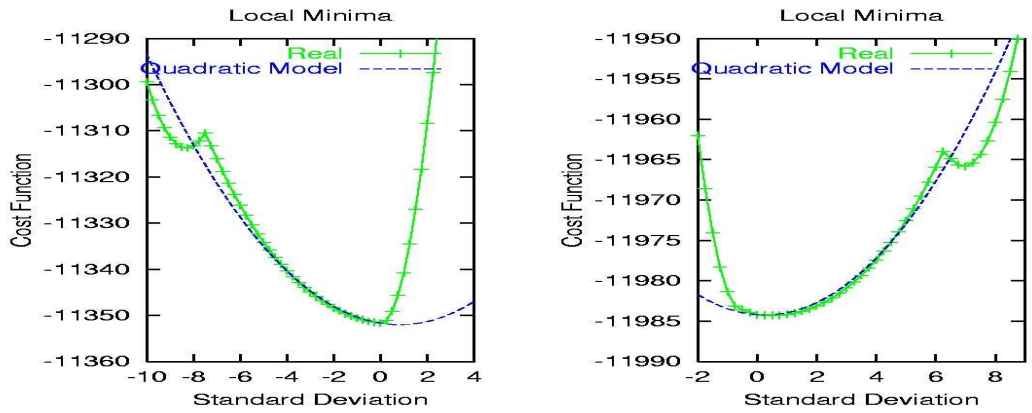


Figure 7: Multimodality along several uncertain eigen-directions (0.8 and 0.9 s in cluttered body tracking sequence).

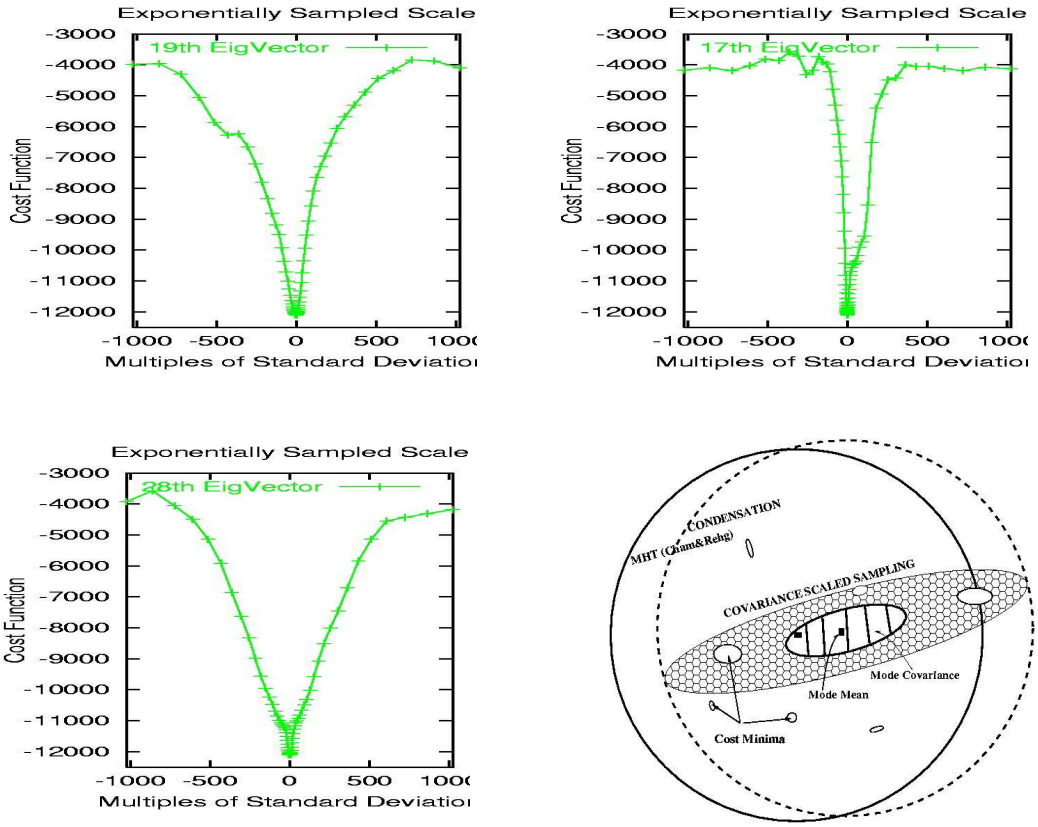


Figure 8: (a,b,c) Cost function slices at large scales, (d) Comparison of sampling methods: (1) CONDENSATION (dashed circle coverage) randomizes each sample by dynamic noise, (2) MHT, solid circle ((Cham and Rehg, 1999), section 3.2 page 3) samples within covariance support (dashed ellipse) and applies the same noise policy as (1), finally, our (3) *Covariance Scaled Sampling* (pattern ellipse) targets good cost minima (flat filled ellipses) by inflating or heavy tail sampling the local robust covariance estimation (dashed ellipse)).

6 Experiments

For the experiments shown here we use an edge and intensity based cost function and a body model incorporating priors and constraints as explained in §3.1 and §3.2. We use Gaussian tails for CSS. A quantitative evaluation of differ-

ent Gaussian scalings appears in table 1 on page 23.

To illustrate our method we show results for an 8 second arm tracking sequence and two full body ones (3.5 s and 4 s). All three sequences contain both self-occlusion and significant relative motion in depth. The first two (fig. 9) were shot at 25 frames (50 fields) per second against a cluttered, unevenly illuminated background. The third (fig. 11) is at 50 non-interlaced frames per second against a dark background, but involves a more complex model and motions. In our unoptimized implementation, a 270 Mhz SGI O2 required about 5 s per field to process the arm experiment and 180 s per field for the full body ones, most of the time being spent in cost function evaluation. The figures show the current best candidate model overlayed on the original images. We also explore the characteristic failure modes of various tracker components, as follows. By a *Gaussian single mode tracker* we mean a single hypothesis tracker doing local continuous optimization based on Gaussian error distributions and without enforcing any physical constraints. A *Robust single mode tracker* improves this by using robust matching distributions. A *Robust single mode tracker with joint limits* also enforces physical constraints. For multimodal trackers, the sampling strategy can be either CONDENSATION-based or CSS-based, as introduced in previous sections.

Cluttered background sequences: These sequences explore 3D estimation behavior with respect to image assignment and depth ambiguities, for a bending rotating arm under an 8 d.o.f. model and a pivoting full-body motion under a 30 d.o.f. one. They have cluttered backgrounds, specular lighting and loose fitting clothing. In the arm sequence, the deformations of the arm muscles are significant and other imperfections in our arm model are also apparent.

The *Gaussian single mode tracker* manages to track 2D frontoparallel motions in moderate clutter, although it gradually slips out of registration when the arm passes the strong edges of the white pillar (0.5 s and 2.2 s for the arm sequence and 0.3 s for the human body sequence). Any significant motion in depth is untrackable.

The *robust single mode tracker* tracks frontoparallel motions reasonably well even in clutter, but quickly loses track during in-depth motions, which it tends to misinterpret as frontoparallel ones. In the arm tracking sequence, shoulder motion towards the camera is misinterpreted as frontoparallel elbow motion, and the error persists until the upper bound of the elbow joint is hit at 2.6 s and tracking fails. In the full body sequence, the pivoting of the torso is underestimated, being partly interpreted as quasi-frontoparallel motion of the left shoulder and elbow joints. Despite the presence of anatomical joint constraints, the fist eventually collapses into the body if non-self-intersection constraints are not present.

The *robust joint-limit-consistent CSS multi-mode tracker* tracks the motion of the entire arm and body sequence without failure. We retain just the 3 best modes for the arm sequence and the 7 best modes for the full human body sequence. As discussed in §4.2, multimodal behavior occurs mainly during significantly non-frontoparallel motions, between 2.2–4.0 s for the arm sequence, and over nearly the entire full body sequence (0.2–1.2 s). For the latter, the modes mainly reflect the ambiguity between true pivoting motion and its incorrect "frontoparallel explanation".

We also compared our method with a 3D version of that of (Heap and Hogg, 1998; Cham and Rehg, 1999). These methods were developed for 2D tracking and we were interested in how well they would behave in the far less well controlled monocular 3D case. We used a parametric Gaussian mixture representation, local descent for mode refinement (as in Heap and Hogg (1998); Cham and Rehg (1999)) and a process model based on constant velocity plus dynamical noise sampling as in Cham and Rehg (1999) (section 3.2, page 3), on the cluttered full body tracking sequence. However, note that unlike the original methods, ours uses robust (rather than least squares) image matching and robust optimization by default, and also incorporates physical constraints and model priors. We used 10 modes to represent the distribution over our 30 d.o.f. 3D configurations, whereas Cham and Rehg (1999) used 10 for their 38 d.o.f. 2D SPM model. Our first set of experiments used a non-robust SSD image matching metric and a Levenberg-Marquardt routine for local sample optimization (as in Cham and Rehg (1999), except that we use analytical Jacobians). With this cost function, we find that outliers cause large fluctuations, bias and frequent convergence to physically invalid configurations. Registration is lost early in the turn (0.5 s), as soon as the motion becomes significantly non-frontoparallel. Our second experiments used our robust cost function and optimizer, but still with sampling as in Cham and Rehg (1999). The track survived further into the turn, but was lost at 0.7 s when the depth variation became larger. As expected, we find that a dynamical noise large enough to provide sufficiently deep sampling along uncertain in-depth directions produces much too deep sampling along well-controlled transversal ones, so that most of the samples are lost on uninformative high-cost configurations. Similar arguments apply to standard CONDENSATION, as can be seen in the monocular 3D experiments of Deutscher et al. (2000).

Black background sequence: In this experiment we focus on 3D errors, in particular depth ambiguities and the influence of physical constraints and parameter stabilization priors. We use an improved body model with 34 d.o.f. The four extra parameters control the left and right clavicle joints in the shoulder complex, which we find to be essential for following many arm motions. Snapshots from the full 4 s sequence are shown in fig. 11, and various failures

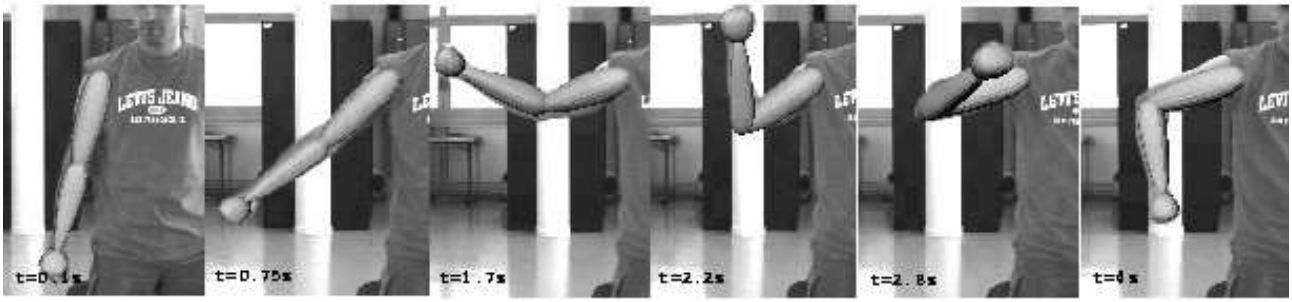


Figure 9: Arm tracking against a cluttered background.

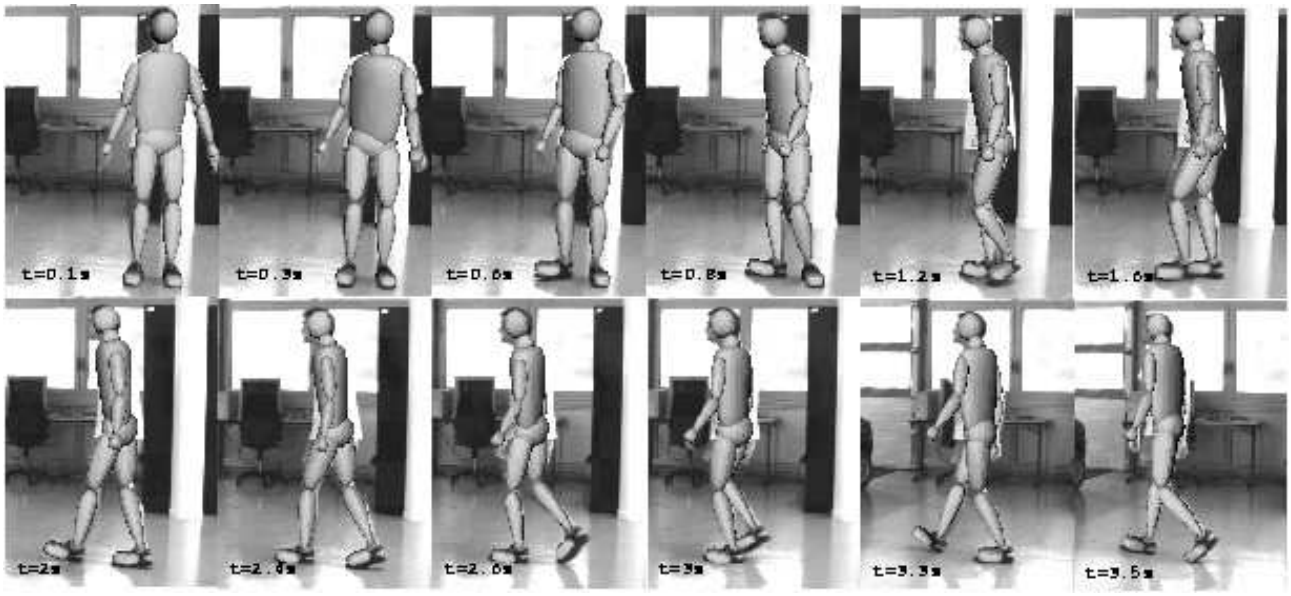


Figure 10: Human tracking against a cluttered background. See plate 14 on page 26 for details.

modes in fig. 12.

The *Gaussian single mode tracker* manages to follow near-frontoparallel motions fairly reliably owing to the absence of clutter, but it eventually loses track after 0.5 s (fig. 12a–d). The *robust single mode tracker* tracks the non-frontoparallel motion somewhat longer (about 1 s), although it significantly misestimates the depth (fig. 12e,f — the right leg and shoulder are pushed much too far forward and the head is pushed forward to match subject contour, *c.f*. the "correct" pose in fig. 11). It eventually loses track during the turn. The *robust multi-mode tracker with joint-limits* is able to track quite well, but, as body non-self-intersection constraints are not enforced, the modes occasionally converge to physically infeasible configurations (fig. 12g) with terminal consequences for tracking. Finally, the *robust fully constrained multi-mode tracker* is able to deal with significantly more complex motions and tracks the full sequence without failure (fig. 11).

7 Sampling Distributions

We also ran some more quantitative experiments aimed at studying the behavior of the different sampling regimes, particularly the efficiency with which they locate minima or low-cost regions of parameter space. We are interested in how the sampling distribution, as characterized by the shape of its core and the width of its tails, impacts the search efficiency. For the study here we used the simple, but still highly multi-modal, 3D joint to image joint likelihood surface that we use for initializing our 34 d.o.f. articulated model. We only estimated joint parameters, not body dimensions. We ran experiments involving Covariance Scaled Sampling (CSS) and Spherical Sampling (SS) for Gaussian distributions with scalings 1,2 8. To allow a fair comparison, at each scale we kept the volume of the sphere (proportional to R^n) equal to the volume of the corresponding rescaled unit covariance CSS ellipsoid (proportional to $\lambda_1 ... \lambda_n$, the product of eigenvalues). Also note

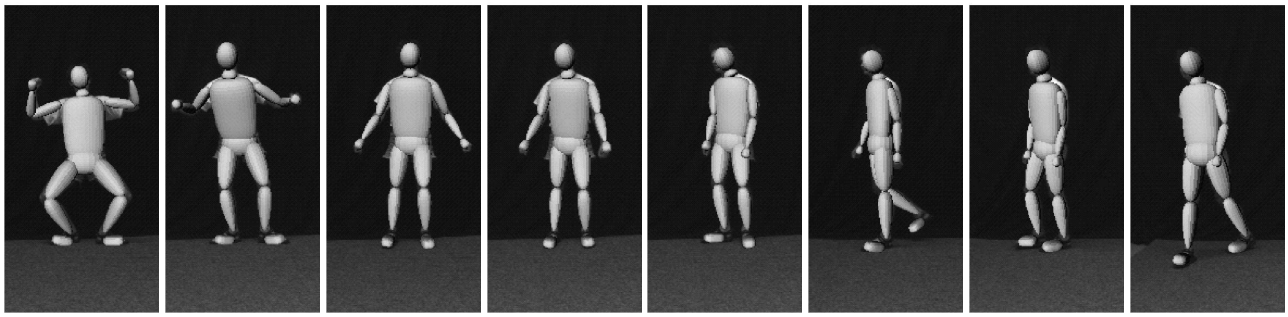


Figure 11: Human tracking under complex motion. See figure 15 on page 27 for details.

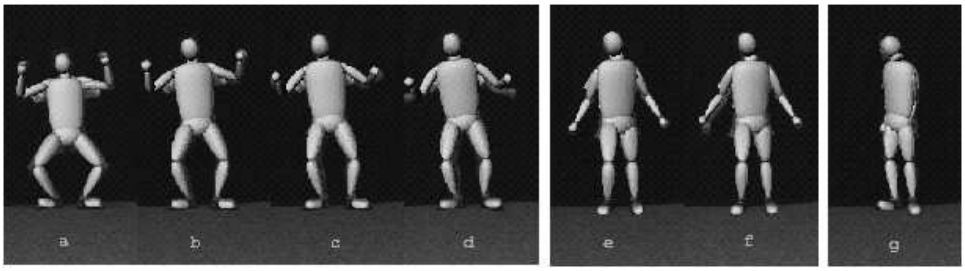


Figure 12: Failure modes of various components of the tracker (see text).

that the final sampling distributions are not exactly Gaussian — in fact they are often noticeably multimodal — because our sampler preserves the physical constraints by projecting inadmissible samples back onto the constraint surface. Once made, the samples are locally optimized subject to the physical constraints using the method of §4.1. We report the number of minima found by each method, and the medians and standard deviations of their parameter space distances and cost differences, in table 1. Fig. 13 shows distributions of numbers of samples and minima versus parameter space distance, standard deviation and cost, for scaling 8. Note that CSS finds significantly more minima, and also places samples at positions of significantly lower cost, than SS. One can also see the large cost difference between optimized and unoptimized samples. SS appears to find minima of slightly lower median cost than CSS, but this is misleading. CSS still finds the few minima found by SS, but it also finds many other more distant ones, which, being further away, tends to increase its median cost.

8 Approximation Accuracy

The tracking experiments in §6 illustrated the practical behavior and failure modes of some of the components of the CSS algorithm, and §7 presented a more quantitative evaluation. Now we turn to more technical points.

The CSS algorithm involves both local continuous optimization and somewhat more global covariance-scaled sampling. It therefore has a natural mechanism to trade-off speed and robustness. When tracking fails, both the number of modes used to represent the distribution and the number of samples produced in the sampling stage can be increased. This increases the computational cost, but it may allow the tracker to follow more difficult portions of the image sequence. In principle, a sufficiently long run of any sampling method would visit every region of the parameter space, so that the basin of attraction of each mode was sampled and all minima were found. It has been argued that mixed continuous/discrete trackers (Heap and Hogg, 1998; Cham and Rehg, 1999) will 'diverge' if the visual information is ambiguous and converge to a 'best' mode when the target in the image is easily detectable. However, this kind of divergence is not that important here. We are working with likelihood surfaces that have multiple peaks with individual probabilities. Local optimization methods can converge to any of these peaks and sampling methods will eventually 'condense' near them if they use enough samples. Given the sampling/dynamics stage, both methods have a chance of jumping between peaks (*i.e.* escaping spurious ones), although this may be a very slow process. The method presented here is designed to address the problems of more efficient and systematic multi-modal exploration. Note also that CSS can be viewed as an importance sampling distribution and correction weighting for fair sample

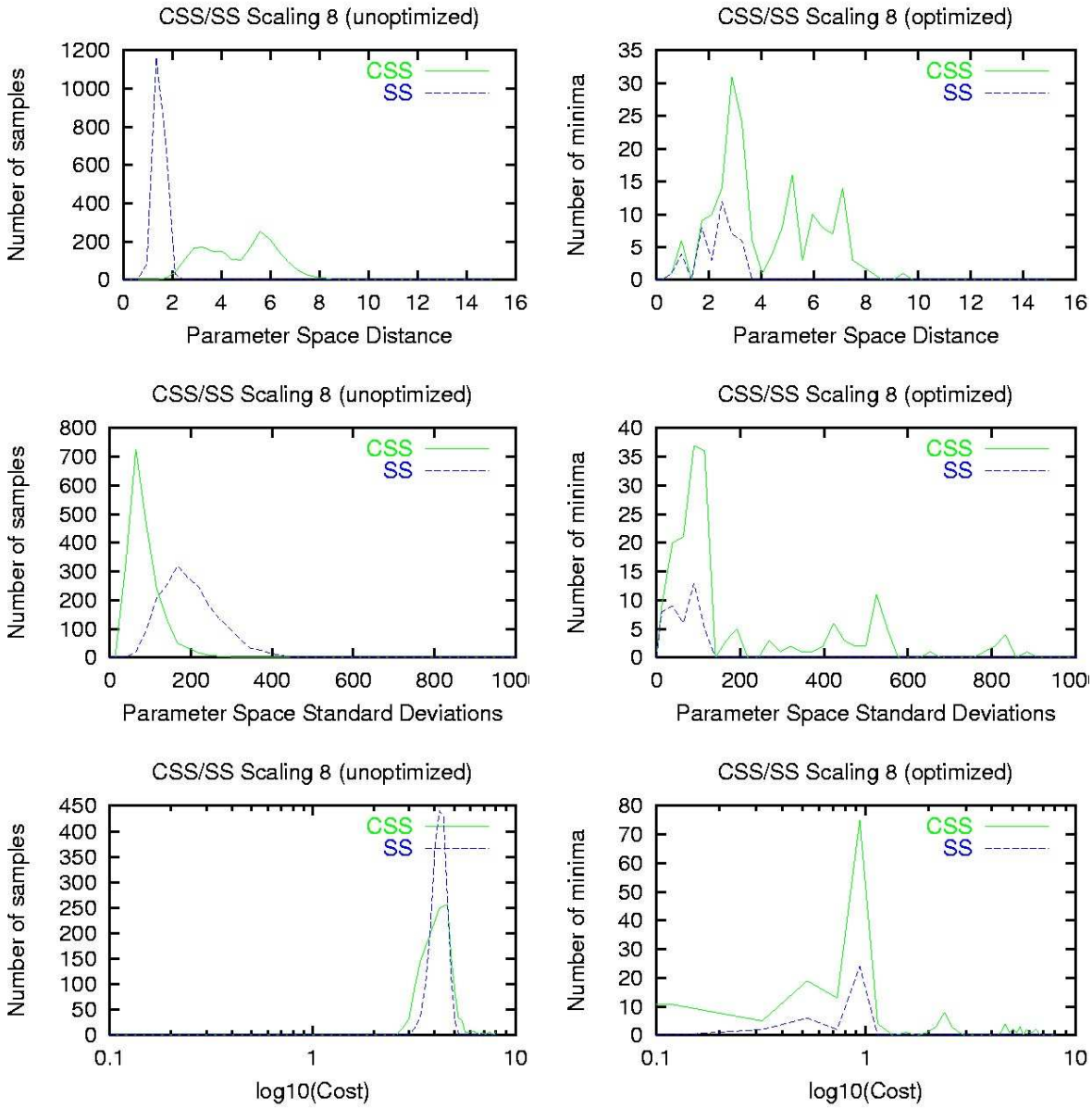


Figure 13: Optimized and unoptimized sample statistics for Spherical (SS) and Covariance Scaled (CSS) Sampling with scaling factor 8 and runs of 2000 samples. Note the significantly larger number of minima found by CSS than by SS, and also that the samples are placed at much lower cost.

generation can be performed with respect to the true prior.

A second issue concerns the algorithm's efficiency versus its bias behavior. In tracking, assuming temporal coherence, one may want to confine the search to the neighborhood of the configurations propagated from the previous tracking step. This can be done implicitly by designing a likelihood surface that emphasizes local responses, or by tuning the search process for locality, using short range dynamics. In either case, a global estimate of the posterior distribution is too expensive, both for sampling and optimization, while restricting attention to nearby states carries the risk of missing distant but significant peaks.

A third issue concerns the approximation accuracy of a Gaussian mixture for arbitrary multi-modal distributions.

For example, an optical flow correspondence field, like that described in §3.1 but based on least squares brightness matching, can behave as a locality prior, forcing local image velocity explanations and pruning away remote, potentially 'objective' multi-modality.

METHOD	SCALE	NUMBER OF MINIMA	PARAMETER DISTANCE MEDIAN		STANDARD DEVIATIONS MEDIAN		COST MEDIAN	
			UNOPT	OPT	UNOPT	OPT	UNOPT	OPT
CSS	1	8	1.148	2.55242	10.9351	47.6042	116.951	8.49689
CSS	4	59	3.21239	2.9474	35.2918	55.3163	1995.12	6.98109
CSS	8	180	4.969	3.34661	75.1119	109.813	16200.8	7.09866
SS	1	0	0.199367	-	24.5274	-	273.509	-
SS	4	11	0.767306	2.04928	96.1519	39.0745	4291.12	6.28014
SS	8	42	1.47262	2.54884	188.157	56.8268	16856.1	6.96481

Table 1: Quantitative results for the distribution of minima found. Note again that CSS finds more minima and places raw samples at lower cost than SS.

The mixture model is likely to be inaccurate away from the mode cores, and this may affect the accuracy of statistical calculations based on it. However for tracking and localization applications we are mainly interested in the modes themselves, not the low-probability regions in their remote tails. Sampling methods are non-parametric so in principle they do not have this limitation, but in practice so few samples fall deep in the tails that noisiness of the estimates is a problem. Pure sample-based representations also provide little insight into the structure of the uncertainty and the degree of multi-modality of the likelihood surface. In any case, the issue of approximation accuracy in low-probability regions is not a main concern here. Provided initial seeds are available in the individual mode's basins of attraction, sampling methods can generate fair samples from the modes and optimization methods can precisely identify their means and covariances by local descent. The two techniques can be use inter-changeably, depending on the application. It is the process of finding the initial seeds for each mode that represents the major difficulty for high-dimensional multi-modal distributions.

A fourth and important practical issue concerns the properties of the likelihood function. For many complex models a good likelihood is difficult to build, and the one used may be a poor reflection of the desired observation density. In these situations, the strength of true and spurious responses is similar and this may affect the performance of the tracking algorithm, irrespective how much computational power is used. In such contexts, it can be very difficult to identify the true tracked trajectory in a temporal flow of spurious responses. This is a particularly complex problem, since many likelihoods commonly used in vision degrade ungracefully under occlusion/disocclusion events and viewpoint change. At present, we do not have good mechanisms for detecting disocclusion events in complex backgrounds. The CSS algorithm has an elegant mechanism that accounts for high-uncertainty if particular degrees of freedom are not observed (like occluded limbs, *etc*.) and it will automatically sample broadly there. However, for sub-sequences with long occlusion events, it is still likely to attach occluded limbs to background clutter, rather than maintaining them as occluded. Global silhouettes, or a human contour detector (Papageorgiu and Poggio, 1999), or higher-order matching consistency (Sminchisescu, 2002a) may help here. As an indication of the potential benefits of this, we currently use foreground/background segmentation and the motion boundaries from the robust optical flow computation to weight the importance of contours, and this significantly improves the results in the sequences we have analyzed.

9 Conclusions and Future Work

We have presented a new method for monocular 3D human body tracking, based on optimizing a robust model-image matching cost metric combining robustly extracted edges, flow and motion boundaries, subject to 3D joint limits, non-self-intersection constraints, and model priors. Optimization is performed using Covariance Scaled Sampling, a novel high-dimensional search strategy based on sampling a hypothesis distribution followed by robust constraint-consistent local refinement to find a nearby cost minima. The hypothesis distribution is determined by propagating the posterior at the previous time step (represented as a Gaussian mixture defined by the observed cost minima and their Hessians / covariances) through the assumed dynamics (here trivial) to find the prior at the current timestep, then inflating the prior covariances and resampling to scatter samples more broadly. Our experiments on real sequences show that this is significantly more effective than using inflated dynamical noise estimates as in previous ap-

proaches, because it concentrates the samples on low-cost points, rather than points that are simply nearby irrespective of cost. In future work, it should also be possible to extend the benefits of CSS to CONDENSATION by using inflated (diluted weight) posteriors and dynamics for sample generation, then re-weighting the results. Our human tracking work will focus on incorporating better pose and motion priors as well as designing likelihoods that are better adapted for human localization in the image.

Acknowledgments

This work was supported by an EIFFEL doctoral grant and the European Union under FET-Open project VIBES. We would like to thank Alexandru Telea for stimulating discussions and implementation assistance and Frédérick Martin for helping with the video capture and posing as a model.

References

A. Barr. Global and Local Deformations of Solid Primitives. *Computer Graphics*, 18:21–30, 1984.

C. Barron and I. Kakadiaris. Estimating Anthropometry and Pose from a Single Image. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 669–676, 2000.

M. Black. *Robust Incremental Optical Flow*. PhD thesis, Yale University, 1992.

M. Black and P. Anandan. The Robust Estimation of Multiple Motions: Parametric and Piecewise Smooth Flow Fields. *Computer Vision and Image Understanding*, 6(1):57–92, 1996.

A. Blake, B. North, and M. Isard. Learning Multi-Class Dynamics. *Advances in Neural Information Processing Systems*, 11: 389–395, 1999.

M. Brand. Shadow Puppetry. In *IEEE International Conference on Computer Vision*, pages 1237–1244, 1999.

C. Bregler and J. Malik. Tracking People with Twists and Exponential Maps. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 1998.

T. Cham and J. Rehg. A Multiple Hypothesis Approach to Figure Tracking. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 239–245, 1999.

K. Choo and D. Fleet. People Tracking Using Hybrid Monte Carlo Filtering. In *IEEE International Conference on Computer Vision*, 2001.

Q. Delamarre and O. Faugeras. 3D Articulated Models and Multi-View Tracking with Silhouettes. In *IEEE International Conference on Computer Vision*, 1999.

J. Deutscher, A. Blake, and I. Reid. Articulated Body Motion Capture by Annealed Particle Filtering. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2000.

J. Deutscher, A. Davidson, and I. Reid. Articulated Partitioning of High Dimensional Search Spacs associated with Articulated Body Motion Capture. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2001.

J. Deutscher, B. North, B. Bascle, and A. Blake. Tracking through Singularities and Discontinuities by Random Sampling. In *IEEE International Conference on Computer Vision*, pages 1144–1149, 1999.

T. Drummond and R. Cipolla. Real-time Tracking of Highly Articulated Structures in the Presence of Noisy Measurements. In *IEEE International Conference on Computer Vision*, 2001.

R. Fletcher. Practical Methods of Optimization. In *John Wiley*, 1987.

D. Gavrila and L. Davis. 3-D Model Based Tracking of Humans in Action:A Multiview Approach. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 73–80, 1996.

L. Gonglaves, E. Bernardo, E. Ursella, and P. Perona. Monocular Tracking of the human arm in 3D. In *IEEE International Conference on Computer Vision*, pages 764–770, 1995.

N. Gordon and D. Salmond. Bayesian State Estimation for Tracking and Guidance Using the Bootstrap Filter. *Journal of Guidance, Control and Dynamics*, 1995.

N. Gordon, D. Salmond, and A. Smith. Novel Approach to Nonlinear/Non-Gaussian State Estimation. *IEE Proc. F*, 1993.

Group. *Specifications for a Standard Humanoid*. Hanim-Humanoid Animation Working Group, http://www.h-anim.org/Specifications/H-Anim1.1/, 2002.

T. Heap and D. Hogg. Wormholes in Shape Space: Tracking Through Discontinuities Changes in Shape. In *IEEE International Conference on Computer Vision*, pages 334–349, 1998.

N. Howe, M. Leventon, and W. Freeman. Bayesian Reconstruction of 3D Human Motion from Single-Camera Video. *Neural Information Processing Systems*, 1999.

M. Isard and A. Blake. CONDENSATION – Conditional Density Propagation for Visual Tracking. *International Journal of Computer Vision*, 1998.

S. Ju, M. Black, and Y. Yacoob. Cardboard People: A Parameterized Model of Articulated Motion. In *2nd Int.Conf.on Automatic Face and Gesture Recognition*, pages 38–44, October 1996.

I. Kakadiaris and D. Metaxas. Model-Based Estimation of 3D Human Motion with Occlusion Prediction Based on Active Multi-Viewpoint Selection. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 81–87, 1996.

H. J. Lee and Z. Chen. Determination of 3D Human Body Postures from a Single View. *Computer Vision, Graphics and Image Processing*, 30:148–168, 1985.

J. Liu. Metropolized Independent Sampling with Comparisons to Rejection Sampling and Importance Sampling. *Statistics and Computing*, 6, 1996.

J. MacCormick and A. Blake. A Probabilistic Contour Discriminant for Object Localisation. In *IEEE International Conference on Computer Vision*, 1998.

J. MacCormick and M. Isard. Partitioned Sampling, Articulated Objects, and Interface-Quality Hand Tracker. In *European Conference on Computer Vision*, volume 2, pages 3–19, 2000.

R. Merwe, A. Doucet, N. Freitas, and E. Wan. The Unscented Particle Filter. Technical Report CUED/F-INFENG/TR 380, Cambridge University, Department of Engineering, May 2000.

C. Papageorgiu and T. Poggio. Trainable Pedestrian Detection. In *International Conference on Image Processing*, 1999.

M. Pitt and N. Shephard. Filtering via Simulation: Auxiliary Particle Filter. *Journal of the American Statistical Association*, 1997.

R. Plankers and P. Fua. Articulated Soft Objects for Video-Based Body Modeling. In *IEEE International Conference on Computer Vision*, pages 394–401, 2001.

J. Rehg and T. Kanade. Model-Based Tracking of Self Occluding Articulated Objects. In *IEEE International Conference on Computer Vision*, pages 612–617, 1995.

K. Rohr. Towards Model-based Recognition of Human Movements in Image Sequences. *Computer Vision, Graphics and Image Processing*, 59(I):94–115, 1994.

R. Rosales and S. Sclaroff. Inferring Body Pose without Tracking Body Parts. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 721–727, 2000.

H. Sidenbladh and M. Black. Learning Image Statistics for Bayesian Tracking. In *IEEE International Conference on Computer Vision*, 2001.

H. Sidenbladh, M. Black, and D. Fleet. Stochastic Tracking of 3D Human Figures Using 2D Image Motion. In *European Conference on Computer Vision*, 2000.

H. Sidenbladh, M. Black, and L. Sigal. Implicit Probabilistic Models of Human Motion for Synthesis and Tracking. In *European Conference on Computer Vision*, 2002.

C. Sminchisescu. Consistency and Coupling in Human Model Likelihoods. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 27–32, Washington D.C., 2002a.

C. Sminchisescu. *Estimation Algorithms for Ambiguous Visual Models–Three-Dimensional Human Modeling and Motion Reconstruction in Monocular Video Sequences*. PhD thesis, Institute National Politechnique de Grenoble (INRIA), July 2002b.

C. Sminchisescu, D. Metaxas, and S. Dickinson. Improving the Scope of Deformable Model Shape and Motion Estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 485–492, Hawaii, 2001.

C. Sminchisescu and A. Telea. Human Pose Estimation from Silhouettes. A Consistent Approach Using Distance Level Sets. In *WSCG International Conference for Computer Graphics, Visualization and Computer Vision*, Czech Republic, 2002.

C. Sminchisescu and B. Triggs. A Robust Multiple Hypothesis Approach to Monocular Human Motion Tracking. Technical Report RR-4208, INRIA, 2001a.

C. Sminchisescu and B. Triggs. Covariance-Scaled Sampling for Monocular 3D Body Tracking. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 447–454, Hawaii, 2001b.

C. Sminchisescu and B. Triggs. Building Roadmaps of Local Minima of Visual Models. In *European Conference on Computer Vision*, volume 1, pages 566–582, Copenhagen, 2002a.

C. Sminchisescu and B. Triggs. Hyperdynamics Importance Sampling. In *European Conference on Computer Vision*, volume 1, pages 769–783, Copenhagen, 2002b.

C. Sminchisescu and B. Triggs. Kinematic Jump Processes for Monocular 3D Human Tracking. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2003.

C. J. Taylor. Reconstruction of Articulated Objects from Point Correspondences in a Single Uncalibrated Image. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 677–684, 2000.

B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle Adjustment - A Modern Synthesis. In Springer-Verlag, editor, *Vision Algorithms: Theory and Practice*, 2000.

D. Vanderbilt and S. G. Louie. A Monte Carlo Simulated Annealing Approach over Continuous Variables. *J. Comp. Physics*, 56, 1984.

S. Wachter and H. Nagel. Tracking Persons in Monocular Image Sequences. *Computer Vision and Image Understanding*, 74(3): 174–192, 1999.

S. C. Zhu and D. Mumford. Learning Generic Prior Models for Visual Computation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11), 1997.

Figure 14: Clutter human tracking sequence detailed results for the CSS algorithm in §6, on page 18.

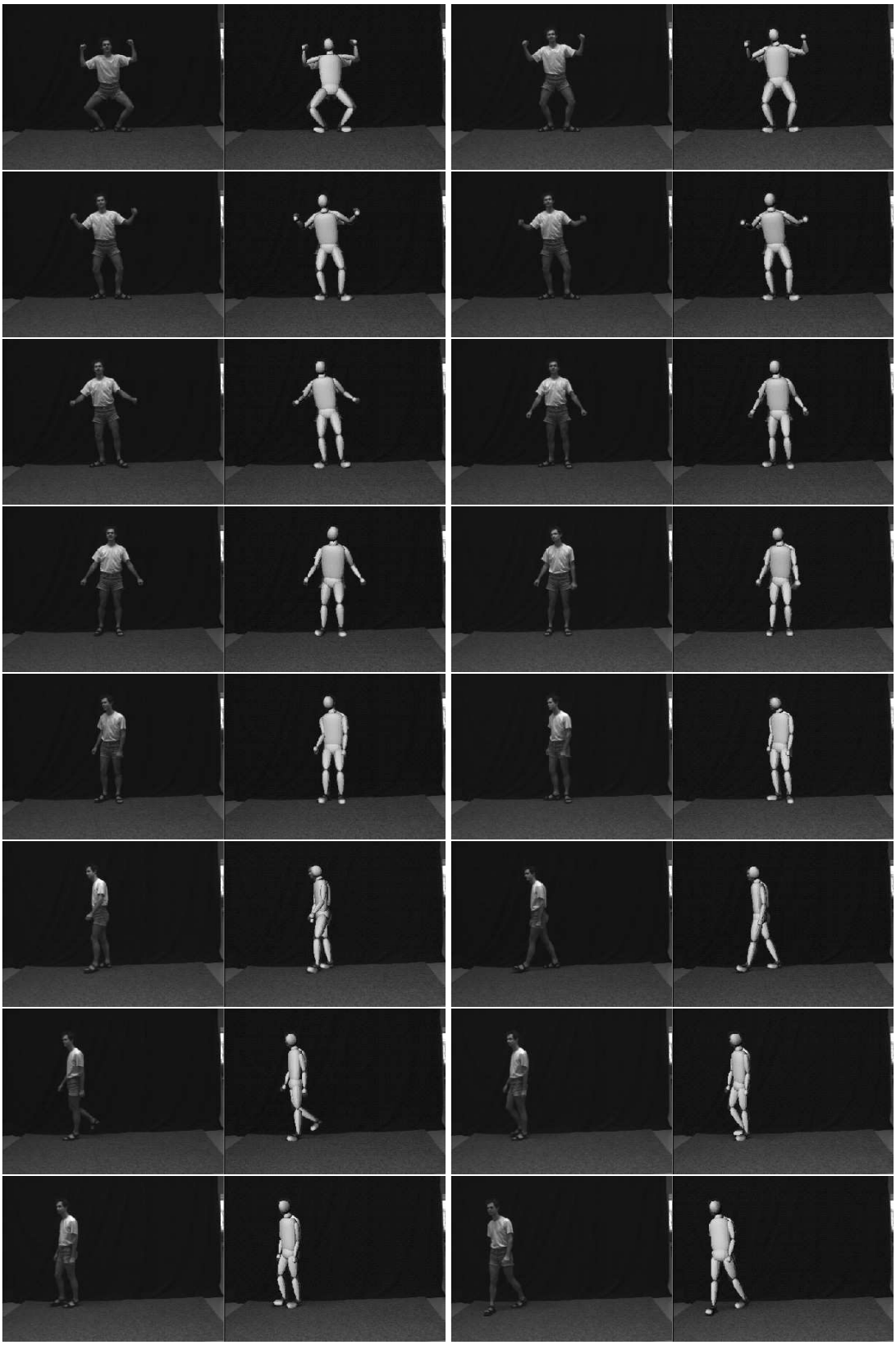


Figure 15: Complex motion tracking sequence detailed results for the CSS algorithm in §6 on page 18.

Building Roadmaps of Minima and Transitions in Visual Models

Cristian Sminchisescu
University of Toronto, Department of Computer Science
Toronto, Ontario, M5S 3G4, Canada
crismin@cs.toronto.edu, www.cs.toronto.edu/~crismin
Bill Triggs
INRIA Rhône-Alpes, GRAVIR-CNRS
655 avenue de l'Europe, 38330 Montbonnot, France
Bill.Triggs@inrialpes.fr, www.inrialpes.fr/movi/people/Triggs

Abstract

Becoming trapped in suboptimal local minima is a perennial problem when optimizing visual models, particularly in applications like monocular human body tracking where complicated parametric models are repeatedly fitted to ambiguous image measurements. We show that trapping can be significantly reduced by building 'roadmaps' of nearby minima linked by transition pathways *— paths leading over low 'mountain passes' in the cost surface — found by locating the* transition state *(codimension-1 saddle point) at the top of the pass and then sliding downhill to the next minimum. We present two families of transition-state-finding algorithms based on local optimization. In* eigenvector tracking*, unconstrained Newton minimization is modified to climb uphill towards a transition state, while in* hypersurface sweeping*, a moving hypersurface is swept through the space and moving local minima within it are tracked using a constrained Newton method. These widely applicable numerical methods, which appear not to be known in vision and optimization, generalize methods from computational chemistry where finding transition states is critical for predicting reaction parameters. Experiments on the challenging problem of estimating 3D human pose from monocular images show that our algorithms find nearby transition states and minima very efficiently, but also underline the disturbingly large numbers of minima that can exist in this and similar model based vision problems.*

Keywords: *Model based vision, global optimization, saddle points, 3D human tracking.*

1 Introduction

Many visual modeling problems can be reduced to cost minimization in a high dimensional parameter space. Local minimization is usually feasible, but practical cost functions often have large numbers of local minima and it can be very difficult to ensure that the desired one is found. Exhaustive search is seldom feasible in more than about 2–3 dimensions, so global optimizers are typically built around heuristics for finding 'good places to look next'. This includes both deterministic methods like branch-and-bound and pattern search, and stochastic importance samplers like simulated annealing [26], genetic algorithms and tabu search [18].

Unfortunately, global optimization remains expensive with any of these methods. In this paper we develop an alternative strategy based on building 1-D 'roadmaps' linking the salient nearby minima. Each local minimum has a basin of attraction within which local optimization converges to it. Roughly speaking, our strategy is to start from a given minimum and to search for low lying

To appear *Int. J. Computer Vision* in early 2005.

'mountain passes' that lead away from its basin of attraction, finding the new minima they lead to by sliding downhill using local optimization. The procedure is repeated recursively to build up a roadmap linking the neighboring minima. More precisely, any minimum-peak-cost path connecting two minima passes through a special state at its highest point, at which the path crosses the brow of the pass and begins to descend. Such "transition states" are saddle points (the gradient of the cost function vanishes there) at which the cost has a local maximum in the direction of the path and a local minimum in all orthogonal directions. Technically, transition states are "codimension 1" saddle points — ones with one negative and (in n dimensions) $n-1$ positive principal curvatures (Hessian eigenvalues). At the heart of our approach are two families of efficient numerical methods for locating the transition states surrounding a given local minimum (see, for instance fig. 4 on page 41).

Although there are many algorithms for finding local minima, it seems that finding transition states has generally been considered to be intractable, and we know of little previous work in the vision or optimization communities on methods for this. However such methods do exist in the computational chemistry / solid state physics community, where transition states are central to the theory of chemical reactions[1]. Both of our classes of transition-state-locating algorithms have roots in chemistry, and both are based on modified forms of local Newton minimization. **Eigenvector tracking** redefines a standard damped Newton minimization to converge uphill to a saddle point of the desired signature, while **hypersurface sweeping** sweeps a moving hypersurface through the space, using a constrained Sequential Quadratic Programming like iteration to track moving local minima within it. These methods should be useful in many visual modeling problems where local minima cause difficulties. Examples include model based tracking, reconstruction under correspondence ambiguities, and various classes of camera pose and calibration problems. In this paper, we present experimental results on monocular model based human pose estimation.

Contents: §2 describes our motivation and overall strategy. §3 discusses related trajectory methods in computational chemistry and global optimization. §4 introduces damped Newton methods for locating transition states, and discusses their relation to local minimization algorithms. §5 and §6 detail our Eigenvector Tracking and Hypersurface Sweeping algorithms. §7 illustrates the methods on a 2D toy problem. §8 presents our main experiments on locating minima during monocular reconstruction and tracking of 3D human body pose. §9 concludes the paper and discusses possible directions for future research.

2 Motivation and Overall Strategy

Overall strategy: Below we will focus on numerical strategies for locating nearby transition states starting from a given minimum. Here we briefly describe how these are used to build "roadmaps" of local minima, thus allowing quasi-global (or at least, somewhat less local) minimization of smooth cost functions with many local minima. Our basic strategy is simply to locate transition states (saddle points that have exactly one negative principal curvature) with one of the methods described below, then to slide downhill using local optimization to find the neighboring minima they lead to.

[1] Atomic assemblies can be modeled in terms of the potential energy induced by interactions among their atoms, *i.e.* by an energy function defined over the high-dimensional configuration space of the atoms' relative positions. A typical assembly spends most of its time near an energy minimum (a stable or quasi-stable state), but thermal perturbations may sometimes cause it to cross a transition state to an adjacent minimum — a chemical reaction. The peak energy of the lowest transition path joining two minima is the main determinant of the likelihood of such a perturbation, and hence determines the dominant reaction rate and mechanism.

Transition states are interesting because the highest point of any locally-minimal-peak-cost path between two minima of a smooth function is necessarily a transition state, and lower-cost paths tend to lead to lower-cost minima. However, the methods studied here can be generalized to find saddle types with larger numbers of negative principal curvatures, if desired.

The transition-state based search method is initialized at one or more known minima, and a working queue of local minima is maintained. At each step, the lowest cost minimum is popped off the queue, transition state searches in several directions are initialized from it, and for each successful search the corresponding neighboring minimum is found by local descent, checked for rediscoveries (within numerical tolerance), and if novel, re-enqueued. In favorable cases the algorithm can be run to completion[2], but if there are too many minima to enumerate explicitly, it can be limited to a fixed number of searches or function evaluations, in which case it tends to find nearby and relatively low-lying minima. This is useful in tracking, as mistracking is more likely to lead to a nearby minimum than a distant one.

Advantages of the transition state based approach: The above method works efficiently in many dimensions, and in practice it is close to linear in the number of minima that need to be enumerated. This good performance is made possible by the fact that transition state location methods are able to use local properties of the cost function to steer the search towards globally meaningful regions (transition states), from which other minima are easily found. In comparison, exhaustive search is infeasible in more than a few dimensions, and methods that search for nearby minima by patterned or purely random sampling also tend to become inefficient in high dimensional problems. Volume increases rapidly with radius in high dimensions, so the low-cost 'cores' of minima tend to be extremely small compared to their higher-cost surroundings. Thus, if random samples based at one minimum are spread widely enough to reach an adjacent minimum — and neighboring minima are often separated by significant distances in parameter space (see fig. 7 below) — they will necessarily also be spread rather thinly owing to the large volume covered, so they can easily miss the minimum altogether, and even if not, they are much more likely to hit the high-cost shoulders of its basin of attraction than in its low-cost 'core'. Hence, even with random or pattern sampling, it is necessary to add a local optimization stage to push samples towards the high-likelihood regions in the 'cores' of the minima.

Moreover, in many applications where the cost function is very ill-conditioned, the sampling pattern needs to be tailored to the local shape of the cost surface to avoid massive sample wastage, as local minima tend to be distributed preferentially along low-cost-change / ill-conditioned directions. In particular, in monocular human tracking, there is substantial ill-conditioning owing to depth recovery ambiguities, and for any given image of the person there are many — usually thousands — of possible 3D kinematic solutions that need to be well sampled, all distributed along the ill-conditioned depth degrees of freedom. The above observations led us to develop the Covariance Scaled Sampling method [43, 46], but transition state search methods turn out to be even more effective at discovering such minima, as they easily find the transition states at the heads of elongated low-cost valleys. See [47] for a complementary minimum enumeration method, that exploits the forward/backward symmetry in the monocular human pose cost function in order to locate kinematic minima efficiently. See also [42] for a smoothing algorithm that reconstructs multiple plau-

[2] It is straightforward to modify this heuristic search method to run indefinitely and converge with probability one simply by adding a random sampling element that is applied, perhaps with an exponentially decreasing probability in order not to degrade the efficiency of the method (*e.g.* consider the following: any time the number of function evaluation or saddle searches reaches an upper bound, say U_b, sample a point at random, optimize to find a local minimum, include it in the working set and increase U_b).

sible state trajectories for models where the multiplicity of solutions persists during tracking. Prior knowledge on typical human poses can be further used to learn application specific low-dimensional continuous generative models that are less ambiguous and allow more efficient search [41].

Difficulties of transition state based approach: Many efficient methods exist for finding local minima of smooth high dimensional cost surfaces. Minimization allows strong theoretical guarantees as the reduction in the function value provides a clear criterion for monitoring progress. For example, for a bounded-below function in a bounded search region, any method that ensures 'sufficient decrease' in the function at each step is 'globally convergent' to some local minimum [15]. Finding saddle points is much harder as there is no universal progress criterion and no obvious analogue of a 'downhill' direction. Newton-style iterations provide rapid *local* convergence near the saddle, but it is not so obvious how to find sufficiently nearby starting points. We will consider several methods that extend the convergence zone. We are mainly interested in saddles as starting points for finding adjacent minima, so we will focus on methods that can be started from a minimum and tuned to find nearby transition states. Efficient 'rubber band relaxation' methods also exist for finding the transition state(s) linking two given minima [35], but we will not need to consider this.

3 Transition State Location Strategies

We start with a brief overview of the computational chemistry / solid state physics literature on locating transition states. This literature should be accessible to vision workers with high-school chemistry and a working knowledge of optimization. However the underlying ideas can be difficult to disentangle from chemistry-specific heuristics, and some of the references are rather naive about numerical optimization issues. We therefore give a self-contained treatment of generalizations of two of the most promising approaches below, in the language of numerical analysis.

A transition state is a local minimum along its $n-1$ positive curvature directions in parameter space, but a local maximum along its remaining negative curvature one. So transition state search methods are often formulated as a series of $(n-1)$-D minimizations, while moving or maximizing along the remaining direction. The main differences lie in the methods of choosing which directions to use.

Eigenvector tracking methods [10, 23, 7, 53, 54, 31, 22, 38, 33, 21, 11, 5] choose the ascent direction to be an eigendirection of the local cost function's Hessian (second derivative) matrix. A Newton based local minimization method is modified to locally increase the cost along the chosen curvature eigendirection, while still reducing it along the remaining eigendirections. In particular, if the lowest (most negative) curvature direction is chosen, the method attempts to find the lowest gradient path to a transition state by walking along the 'floor' of the local cost 'valley'. However success can not be guaranteed, as such valleys may continue indefinitely without leading to a saddle point [25, 49, 24]. Also, the method is not as canonical as it may seem: the eigendecomposition depends non-trivially on the coordinate system used, so, *e.g.*, changing the relative scaling of variables changes the trajectory followed; and to ensure progress, "the same" eigenvector must be followed at each step, but there is no natural identification between eigenvectors at different points, so there is no canonical way to ensure this sameness.

An early eigenvector tracking method [23] used explicit Newton minimization in the $(n-1)$-D space obtained by eliminating the coordinate with the largest overlap with the desired up-hill direction. Later quasi-Newton methods use Lagrange multipliers [7, 53, 54] or shifted Hessian eigenvalues [38, 33, 21, 11, 5] to ensure that the cost function is increased along the 'uphill' eigendi-

rection while still being minimized in the orthogonal subspace. Owing to the lack of a global identification between eigenspaces at different points, maintaining a consistent direction to follow can be delicate and several competing methods exist, including using a fixed eigenvector index [23, 7, 53, 54, 31, 22] and following the eigendirection that is best aligned (has greatest dot product) with the current one [38, 33, 21, 11, 5].

Eigenvector tracking can be viewed as a 'virtual cost minimization' obtained by inverting the sign of both the 'uphill' eigenvalue and its corresponding gradient component [30, 21] (see below). This gives an intuitive algebraic analogy with minimization, but none of minimization's convergence guarantees as the virtual cost function changes at each step.

Trajectory methods from global optimization [14, 20, 6] are based on the idea of tracing routes through stationary points of a high-dimensional cost function. Some early 2D methods are based on alternating descents and ascents along eigendirections [14], without any clear ascent heuristic. "Golf" methods [20], aim at exploring different minima and equilibrate at a certain energy level (assumed known a-priori). There are also randomized methods that speed up classical molecular dynamics simulations by generating trajectories on a modified potential with reduced well depth [51, 52] and lower potential at transition states. This insures both increased inter-minimum transition rates (and thus faster simulations), and unbiased behavior, in that the correct relative rates of different transitions are preserved in the modified potential. An application of a generalization of these methods to computer vision appears in [45]. Branin type methods [6, 3] are based on solving differential equations derived from the gradient stationarity condition, rather than explicitly minimizing the cost function. However, managing the singularities and bifurcations that arise during the iteration is an open problem.

Space Sweeping and Filling Methods: The above methods can fail to make global progress, *e.g.* owing to looping. Space sweeping methods [10, 1, 2, 4, 30, 19, 28, 17] guarantee more systematic progress by using a moving potential or hypersurface to sweep candidate points though the space in such a way that they often pass through transition states. Crippen & Sheraga's early method [10] builds an uphill path by stepping along a prespecified direction and minimizing in the hyperplane orthogonal to the direction at each step. Mousseau & Barkema use a similar but less rigorous technique based on changing the gradient sign in one direction followed by conjugate root-finding in the other directions [30]. Barkema [4] pushes the solution away from a known minimum by adding a steadily expanding repulsive spherical bias potential centred on the minimum, and optimizing subject to this soft constraint. New minima are found but the method does not attempt to pass exactly through saddle points. Abashkin & Russo [1, 2] minimize on successively larger-radius hyperspheres centered at a minimum, and also provide a method for refining approximate saddle point locations. The use of hyperspheres forces the search to move initially along the valley floor of the cost surface [24], so usually at most two distinct saddles can be found. Below we will show how to steer the initial search along any desired direction by using ellipsoidal surfaces. Closely related "*penalty methods*" from global optimization (filled function, tunneling) attempt to modify the cost function to prevent reconvergence to the currently known local minima, thus forcing the discovery of new ones [19, 28, 17]. As the extents of the basins of attraction are not known in advance, this requires the addition of increasingly strong repulsive potentials centered at the known minima, until a transition state is crossed and a new solution is found. The optimization becomes more complex as more minima are discovered because a repeller potential must be included for each known minimum. Including a large number of repellers also leads to an increasingly flat and contorted cost function. Repeller ideas are similar to those used in 'tabu search' methods [18], which try to avoid trapping and cyclic behavior by forbidding or penalizing moves that return

to recently-visited states. Tabu search is traditionally a combinatorial method, but it can also be applied in the continuous domain by choosing a suitable problem discretization.

All of the above methods depend on particular choices. The hypersurface and repulsion based methods depend on the local shape and alignment of the moving hypersurfaces or potentials, while the eigenvector and valley following methods all depend on the local coordinate system — even linear changes of coordinates completely change the eigenvalues and vectors, the notion of slowest-ascent valleys, *etc*. These dependencies are much stronger than in minimization, where they enter only via Levenberg-Marquardt style step damping (pure Newton iteration being coordinate-independent). They can be a nuisance, but they also allow initial search directions to be varied, so that many transition states can potentially be found starting from any given minimum.

The following sections detail several basic saddle point location methods. Damped Newton iteration §4 is useful for refining estimated saddle points but its convergence domain is too limited for general use. Eigenvector tracking §5 extends the convergence domain by operating a virtual Newton optimization (thus remaining relatively lightweight), but may be sensitive to the 'eigenvector following' heuristics. Hypersurface sweeping §6 is better founded in that it provides more guarantees of global progress, but no single sweep finds all saddle points and it is more complex to implement.

4 Damped Newton Methods for Finding Transition States

We first consider local damped Newton-like methods for finding saddle points. These provide rapid local convergence near saddle points, but higher level strategies are needed to ensure more global convergence. Let $f(\mathbf{x})$ be the cost function being optimized over n-D parameter vector $\mathbf{x}$, and let $\mathbf{g} \equiv \frac{\partial f}{\partial \mathbf{x}}$ be the function's **gradient** and $\mathbf{H} \equiv \frac{\partial^2 f}{\partial \mathbf{x}^2}$ be its **Hessian** at the current point $\mathbf{x}$. We seek transition states, *i.e*.stationary points ($\mathbf{g}(\mathbf{x})=\mathbf{0}$) at which the Hessian has one negative and $n-1$ positive eigenvalues. If there is a stationary point at $\mathbf{x}+\boldsymbol{\delta}\mathbf{x}$, first order Taylor approximation at $\mathbf{x}$ gives:

$$\mathbf{0} \;=\; \mathbf{g}(\mathbf{x}+\boldsymbol{\delta}\mathbf{x}) \;\approx\; \mathbf{g}(\mathbf{x}) + \mathbf{H}\,\boldsymbol{\delta}\mathbf{x} \tag{1}$$

Solving this linear system for $\boldsymbol{\delta}\mathbf{x}$ and iterating to refine the approximation gives the **Newton iteration**:

$$\mathbf{x} \;\leftarrow\; \mathbf{x}+\boldsymbol{\delta}\mathbf{x} \qquad \text{with update} \qquad \boldsymbol{\delta}\mathbf{x} \;=\; -\mathbf{H}^{-1}\mathbf{g} \tag{2}$$

When started sufficiently close to any regular stationary point ('regular' means roughly that $\mathbf{H}$ is nonsingular and 2^{nd} order Taylor expansion converges), Newton's method converges to it, but how close you need to be is a delicate point in practice.

For Newton-based *minimization*, the convergence can be globalized by adding suitable damping to shorten the step and stabilize the iteration. Most formulations use the **damped Newton** update:

$$\boldsymbol{\delta}\mathbf{x} \;=\; -(\mathbf{H}+\lambda\mathbf{D})^{-1}\mathbf{g} \tag{3}$$

where $\mathbf{D}$ is a positive diagonal matrix (often the identity). The **damping factor** $\lambda > 0$ is manipulated by the algorithm to ensure stable and reliable progress downhill towards the minimum. Damping can be viewed as Newton's method applied to a modified local model for f, whose gradient at $\mathbf{x}$ is unchanged but whose curvature is steepened to $\mathbf{H}+\lambda\mathbf{D}$.

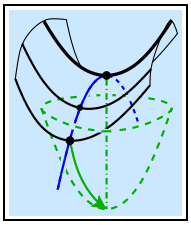
More generally, to stabilize Newton iteration near a saddle point, the negative Hessian curvatures must be made more negative, and the positive ones more positive. More globally, 'uphill

motion' directions must be given sufficiently strong negative curvature, and 'downhill motion' ones sufficiently strong positive curvature. This can be conveniently expressed in a Hessian eigenbasis $\mathbf{H} = \mathbf{V}\,\mathbf{E}\,\mathbf{V}^\top$, where $\mathbf{E} = \mathrm{diag}(\lambda_1, ..., \lambda_n)$ are the eigenvalues and the columns of $\mathbf{V}$ are the eigenvectors of $\mathbf{H}$ with respect to the current coordinates. In this basis, the undamped Newton update becomes $\boldsymbol{\delta}\mathbf{x} = -\mathbf{V}\,(\bar{g}_1/\lambda_1, \ldots, \bar{g}_n/\lambda_n)^\top$, where $\bar{g}_i \equiv (\mathbf{V}^\top\mathbf{g})_i$ are the eigen-components of the gradient. Damping can be introduced by replacing this with:

$$\boldsymbol{\delta}\mathbf{x} = -\mathbf{V}\,\mathbf{u}(\lambda), \quad \text{where} \quad \mathbf{u}(\lambda) \equiv \left(\tfrac{\bar{g}_1}{\lambda_1+\sigma_1\lambda}, ..., \tfrac{\bar{g}_n}{\lambda_n+\sigma_n\lambda}\right)^\top = \left(\tfrac{\sigma_1\bar{g}_1}{\sigma_1\lambda_1+\lambda}, ..., \tfrac{\sigma_n\bar{g}_n}{\sigma_n\lambda_n+\lambda}\right)^\top \tag{4}$$

where $\sigma_i = \pm 1$ is a desired sign pattern for the λ_i. Damping $\lambda > \max_i(-\sigma_i\lambda_i, 0)$ ensures that the denominators are positive, so that the iteration moves uphill to a maximum along the eigendirections with $\sigma_i = -1$ and downhill to a minimum along the others. At each step this can be viewed as the minimization of a virtual local function with curvatures $\lambda + \sigma_i\lambda_i$ and sign-flipped gradients $\sigma_i\bar{g}_i$. However the virtual model changes at each step and f itself is *not* minimized, so none of the usual convergence guarantees of well-damped minimization apply.

As in minimization, λ must be varied to ensure smooth progress. There are two main strategies for this: **Levenberg-Marquardt** methods manipulate λ directly, while **trust region** ones maintain a local region of supposed-'trustworthy' points, and choose λ to ensure that the step stays within it, for instance $\|\boldsymbol{\delta}\mathbf{x}(\lambda)\| = \|\mathbf{u}(\lambda)\| \lesssim r$ where r is a desired 'trust radius'. (Such a λ can be found efficiently with a simple 1-D Newton iteration started at large λ [15]). In either case, one monitors model accuracy metrics such as the second-order-Taylor based relative f-prediction error:



$$\beta = \left|\tfrac{f(\mathbf{x}+\boldsymbol{\delta}\mathbf{x})-f(\mathbf{x})}{\mathbf{g}^\top\boldsymbol{\delta}\mathbf{x}+\boldsymbol{\delta}\mathbf{x}^\top\mathbf{H}\boldsymbol{\delta}\mathbf{x}/2} - 1\right| \tag{5}$$

as well as convergence criteria. Low accuracy indicates that the damping should be increased (larger λ or smaller r), and high accuracy that it can safely be decreased to allow longer steps and speed convergence (*e.g.*, by scaling λ or r up or down by fixed constants).

As in minimization, if the exact Hessian is unavailable, quasi-Newton approximations based on previously computed gradients can be used. Since positive definiteness is no longer required, non-positive update rules such as Powell's are generally preferred to the standard BFGS one [15, 5]:

$$\mathbf{H} \leftarrow \mathbf{H} - \frac{\boldsymbol{\delta}\mathbf{x}^\top\boldsymbol{\xi}}{\|\boldsymbol{\delta}\mathbf{x}\|^4}\boldsymbol{\delta}\mathbf{x}\,\boldsymbol{\delta}\mathbf{x}^\top + \frac{\boldsymbol{\xi}\,\boldsymbol{\delta}\mathbf{x}^\top + \boldsymbol{\delta}\mathbf{x}\,\boldsymbol{\xi}^\top}{\|\boldsymbol{\delta}\mathbf{x}\|^2} \quad \text{where} \quad \boldsymbol{\xi} = \mathbf{g}(\mathbf{x}+\boldsymbol{\delta}\mathbf{x}) - \mathbf{g}(\mathbf{x}) - \mathbf{H}(\mathbf{x})\boldsymbol{\delta}\mathbf{x} \tag{6}$$

5 Eigenvector Tracking

Eigenvector tracking methods take the above Hessian eigenbasis method and add a heuristic for selecting which eigendirection(s) to modify. Basically, once the coordinate system has been fixed, the remaining freedom in (4) is the choice of the signs σ_i. The damped iteration tries to move uphill to a maximum along directions with $\sigma_i = -1$, and downhill to a minimum along directions with $\sigma_i = +1$, *i.e.* it tries to find a stationary point whose principal curvatures λ_i have the same signs as the σ_i. So for minimization we need $\sigma_i = +1$, while for a transition state search exactly one σ_i must be made negative. (This holds irrespective of the λ_i and $\bar{g}_i$ at the *current* state, which affect only the amount of damping required for stability).

The question of which eigenvalue to change is less obvious than it might seem. To ensure continued progress we need to flip "the same" eigenvalues at each step. Unfortunately, there is no

Eigenvector Tracking Transition State Search

Initialization
Set starting point $\mathbf{x}_0$, initial tracking direction $\mathbf{t}=\mathbf{v}_e$ for some e, and initial trust radius r.

Eigenvector Tracking Loop
A. At $\mathbf{x}_k$, find $f_k, \mathbf{g}_k, \mathbf{H}_k$, the Hessian eigen-decomposition $(\lambda_i, \mathbf{v}_i)$ and eigen-basis gradient $\bar{g}_k$. Set n_- to the number of negative eigenvalues. If the problem has active internal constraints, project $\mathbf{t}$ onto the constraint surface.
B. If $k > 0$ choose $e = \max_i |\mathbf{v}_i^\top \mathbf{t}|$. Set $\alpha = |\mathbf{v}_e^\top \mathbf{t}|$ and $\mathbf{t} = \mathbf{v}_e$.
C. If $\lambda_e > 0$ set $\bar{g}_e = -\bar{g}_e$. Take an undamped Newton step if n_-=1 and $\|\boldsymbol{\delta}\mathbf{x}\| \leq r$. Otherwise take a damped one (4) with λ chosen so that $\|\boldsymbol{\delta}\mathbf{x}\| \leq r$.
D. Find the new f and the modeling error β (5). If $\beta > 0.3$ (say) or $\alpha < 0.7$ (say), shrink the trust radius r by say 50%. Otherwise, if $\beta < 0.2$ (say) and we took a damped step, grow r by say 40%.
E. If $\beta \geq 1$ go to step C (try a shorter step). If $\|\mathbf{g}_k\| < \epsilon$ return success if n_-=1, failure otherwise (convergence / failure exit case). Otherwise, go to step A (do next iteration).

Figure 1: Summary of the eigenvector tracking algorithm for transition state search.

globally well defined correspondence rule linking eigenvectors at different points, so there is no rigorous definition of "sameness" and heuristics must be used. For transition state searches we only need to track a single eigenvector (the one that is given $\sigma_i = -1$), so we will concentrate on this case. A simple approach would be to choose a fixed direction in space (perhaps the initial eigendirection) and take the eigenvector with maximal projection along this direction. But many such directions are possible and the most interesting transition states may happen not to have negative curvature along the particular direction(s) chosen. Alternatively, we can try to track a given eigendirection as we move. The problem is that globally, eigendirections are by no means stable. Eigenvalues change as we move about the space, but generically (in codimension 1) they never cross. When they approach one another, the eigenbasis of their 2D subspace becomes ill-conditioned and slews around through roughly 90°. Seen from a large enough scale, the eigenvalues appear to cross with more or less constant eigenvectors, but on a finer scale there is no crossing, only a smooth but rapid change of eigendirection that is difficult to track accurately. Whichever of the two behaviors is desired, it is difficult to choose a step length that reliably ensures it, so the numerical behavior of eigenvector-tracking methods can sometimes be sensitive to fine scale steps. In fact, the imprecise coarse scale view is probably the desired one: if we are tracking a large eigenvalue and hoping to reduce it to something negative, it will have to "pass through" each of the smaller eigenvalues. Tracking at too fine a scale is fatal as it (correctly) prevents such crossings, instead making the method veer off at right angles to the desired trajectory. Even without these problems, there is no guarantee that a saddle point of the desired signature is found — the trajectory might simply continue to climb indefinitely. Also, as with other damped Newton methods, the whole process is dependent on the affine coordinate system used. Nevertheless, eigenvector tracking is relatively lightweight, simple to implement, and it often works well in practice. Generalization to k-th order saddles is straightforward: the above procedure is applied to the lowest k (or some k) eigenvectors, and these are tracked simultaneously. At each step, we simply choose the damping factor λ to ensure that the corresponding directions have negative augmented curvatures.

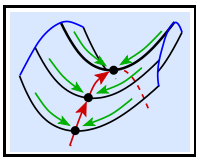
5.1 Implementation Details

Our implementation of eigenvector tracking is summarized in fig. 1. We use the damped Newton saddle step (4), moving away from the minimum by reversing the sign of the gradient in the tracked eigendirection if this has positive curvature. The damping $\lambda > 0$ is controlled to keep the step within a trust radius r and to dominate any undesired negative eigenvalues. The trust radius is set by monitoring the accuracy (5) of the local model for f.

In some of our target applications, the underlying problem also has bound constraints that need to be maintained. For hypersurface sweeping (below) this just adds additional constraints to the within-hypersurface minimizations, but for eigenvector tracking we introduced a trust region step calculation routine that uses a projection strategy to handle constraints on $\mathbf{x}$, and also projects the eigenvector-tracking direction $\mathbf{t}$ along the constraints to ensure stability.

6 Hypersurface Sweeping

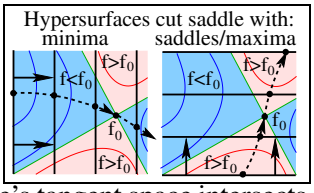
Eigenvector trackers do not enforce any notion of global progress, so they can sometimes cycle or stall. To prevent this we can take a more global approach to the transition state searches '$(n-1)$-D minimization and 1D maximization'. Hypersurface sweeping approaches sweep an $(n-1)$-D hypersurface across the parameter space — typically a moving hyperplane or an expanding hyper-ellipsoid centered at the initial minimum — tracking local minima within the hypersurface and looking for temporal maxima in their function values. The intuition is that as the hypersurface sweeps towards a transition state, and assuming that it approaches along its cone of negative curvature directions, the $(n-1)$-D minimization forces the hypersurface-minimum to move along the lowest path leading up to the saddle's 'col', and the 1-D maximization detects the moment at which the col is crossed. The method can not stall or cycle as the hypersurface sweeps through each point in the space exactly once.



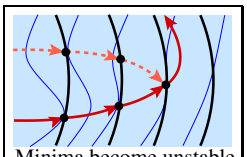
The moving hypersurface can be defined either implicitly, in terms of the level sets $c(\mathbf{x}) = t$ of some parameter space function $c(\mathbf{x})$ (a linear form for hyperplanes, a quadratic one for hyper-ellipsoids...), or explicitly in terms of a local parameterization $\mathbf{x} = \mathbf{x}(\mathbf{y}, t)$ with respect to some hypersurface-t-parameterizing $(n-1)$-D vector $\mathbf{y}$. The minimum-tracking problem becomes:

$$\text{local\_max}_t \; f_c(t) \qquad \text{where} \qquad f_c(t) \equiv \begin{cases} \text{local\_min}_{c(\mathbf{x})=t} \, f(\mathbf{x}) \\ \text{local\_min}_{\mathbf{y}} \, f(\mathbf{x}(\mathbf{y}, t)) \end{cases} \tag{7}$$

There are some caveats. Firstly, we may need to find and track several minima, as different local minima on the hypersurface typically lead to different transition states. Secondly, even if we could track every local minimum within the hypersurface, this would not suffice to find every transition state: each family of hypersurfaces is 'blind' to some transition state orientations. Transitions that are approached in 'downhill' rather than 'uphill' directions — ones that are cut in negative curvature directions (the hypersurface's tangent space intersects the transition states' cone of negative curvature directions) — appear as saddle points or local maxima within the hypersurface, and so can not be found by tracking minima alone. Finally, some local maxima of $f_c(t)$ do not indicate transition states. The minimum being tracked can also simply disappear (topologically annihilate with another within-hypersurface saddle point), causing the tracked



point to suddenly fall away to some other minimum on the hypersurface. This generates an abrupt 'sawtooth'-shaped maximum in $f_c(t)$. In more detail, at any stationary-within-hypersurface point, the projection of $\mathbf{g}=\frac{\partial f}{\partial \mathbf{x}}$ vanishes, so f's isosurface is necessarily tangent to the local hypersurface: $\mathbf{g} \propto \frac{\partial c}{\partial \mathbf{x}}$ or $\mathbf{g}^\top \frac{\partial \mathbf{x}}{\partial \mathbf{y}} = \mathbf{0}$. The point is a within-hypersurface-minimum, -saddle, or -maximum respectively as the cost isosurface has higher/mixed/lower signed curvature than the local hypersurface (*i.e.* as the isosurface is locally inside/mixed/outside the hypersurface). At points where the moving hypersurface transitions from being outside to being mixed w.r.t. the local isosurface, the minimum being tracked disappears and the solution drops abruptly to some other hypersurface-minimum, producing a 'sawtooth' maximum in $f_c(t)$. The sweep can continue from the new minimum so this is not a problem as far as finding subsequent minima is concerned, but if transition states are needed it is important to eliminate the sawtooth maxima. (Similarly, minima of $f_c(t)$ correspond to true local minima of $f(\mathbf{x})$ — and there is no problem of 'blindness' in this case — *except* when they result from a sawtooth transition).


Minima become unstable when isosurfaces curve less than hypersurfaces

As each family of hypersurfaces is blind to some transition state orientations, it is useful to try several different families. For example, hyperplanes find forwards-looking transitions while hyper-ellipsoids find outward-looking ones. If we start the track at a minimum $\mathbf{x}_0$ of $f(\mathbf{x})$, the initial tracking direction is determined by the hyperplane normal or the ellipsoid shape and the Hessian $\mathbf{H} = \mathbf{H}(\mathbf{x}_0)$. First consider the family of hyper-ellipsoids $c(\mathbf{x}) = (\mathbf{x}-\mathbf{x}_0)^\top \mathbf{A} (\mathbf{x}-\mathbf{x}_0) = t$, where $\mathbf{A}$ is some positive definite matrix. To second order, $f(\mathbf{x})$ generically has exactly two local minima on an infinitesimal ellipsoid $c(\mathbf{x}) = t$: the $\pm$ directions of the smallest eigenvector of the matrix pencil $\mathbf{A}+\lambda\mathbf{H}$. (Numerically, these can be found by generalized eigendecomposition of $(\mathbf{A}, \mathbf{H})$, or standard eigendecomposition of $\mathbf{L}^{-\top}\mathbf{H}\mathbf{L}^{-1}$ where $\mathbf{L}\mathbf{L}^\top$ is the Cholesky decomposition of $\mathbf{A}$). For most $\mathbf{A}$ there are thus only two possible initial trajectories for the moving minimum, and so at most two nearest transition states will be found. To find other nearby transition states we need to modify $\mathbf{A}$. We can enforce any desired initial direction $\mathbf{u}$ by taking the 'neutral' search ellipsoids $\mathbf{A} = \mathbf{H}$ (on which f is constant to second order, so that all initial directions are equally good) and flattening them slightly relative to the cost isosurfaces in the direction $\pm\mathbf{u}$. To satisfy the Lagrange multiplier condition for a constrained minimum, $\frac{\partial c}{\partial \mathbf{x}} \propto \frac{\partial f}{\partial \mathbf{x}}$, we can take:

$$\mathbf{A} = \mathbf{H} + \mu \frac{\mathbf{g}\,\mathbf{g}^\top}{\mathbf{u}^\top \mathbf{g}} \tag{8}$$

where $\mathbf{g} = \mathbf{H}\,\mathbf{u}$ is the cost gradient (and hence isosurface normal) for a small displacement $\mathbf{u}$, and μ is a positive constant, say $\mu \sim 0.1$ for mild flattening. Similarly, for hyperplanes $c(\mathbf{x}) = \mathbf{n}^\top (\mathbf{x}-\mathbf{x}_0) = t$ with normal $\mathbf{n}$, the initial tracking direction is $\mathbf{u} = \pm\mathbf{H}^{-1}\mathbf{n}$, so to search in direction $\mathbf{u}$ we need to take $\mathbf{n} = \mathbf{H}\,\mathbf{u}$. (Note that, particularly if $\mathbf{H}$ is ill-conditioned, more neutrally scaled $\mathbf{A}, \mathbf{n}$ tend to produce to initial directions that are closely aligned with $\mathbf{H}$'s smallest eigenvector, which is likely to lead to poor search diversity).

By finding and tracking within-hypersurface saddle points rather than just within-hypersurface minima, the sweeping method can in principle be generalized to find saddle points with several negative eigen-curvatures, but similar caveats would apply.

6.1 Hypersurface Sweeping Equations

This section summarizes the equations needed to implement hypersurface sweeping, for both implicitly-specified and parametrically-specified hypersurfaces. For the implicit approach, let $\mathbf{g}_c \equiv \frac{\partial c}{\partial \mathbf{x}}$ and

$\mathbf{H}_c \equiv \frac{\partial^2 c}{\partial \mathbf{x}^2}$ be the gradient and Hessian of the hypersurface constraint function $c(\mathbf{x})$. The hypersurface constraint is enforced with a Lagrange multiplier λ, solving:

$$\frac{\partial}{\partial \mathbf{x}} (f + \lambda\, c) \;=\; \mathbf{g} + \lambda\, \mathbf{g}_c \;=\; \mathbf{0} \qquad \text{subject to} \qquad c = t \tag{9}$$

If we are currently at $(\mathbf{x}, \lambda)$, second order Taylor expansion of these equations for a constrained minimum at $(\mathbf{x}{+}\boldsymbol{\delta}\mathbf{x}, \lambda{+}\delta\lambda)$ gives the standard **sequential quadratic programming** update rule for $(\boldsymbol{\delta}\mathbf{x}, \delta\lambda)$:

$$\begin{pmatrix} \mathbf{H}_\lambda & \mathbf{g}_c \\ \mathbf{g}_c^\top & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\delta}\mathbf{x} \\ \delta\lambda \end{pmatrix} \;=\; -\begin{pmatrix} \mathbf{g} + \lambda\, \mathbf{g}_c \\ c - t \end{pmatrix} \qquad \text{where} \qquad \mathbf{H}_\lambda \;\equiv\; \mathbf{H}{+}\lambda\, \mathbf{H}_c \tag{10}$$

(The $\lambda\, \mathbf{H}_c$ term in the Hessian is often dropped for simplicity. This slows the convergence but still gives correct results).

Similarly, in the parametric approach let $\mathbf{J} \equiv \frac{\partial}{\partial \mathbf{y}} \mathbf{x}(\mathbf{y}, t)$. The chain rule gives the reduced gradient $\mathbf{g}_\mathbf{y}$ and Hessian $\mathbf{H}_\mathbf{y}$:

$$\mathbf{g}_\mathbf{y} = \mathbf{J}\, \mathbf{g} \tag{11}$$

$$\mathbf{H}_\mathbf{y} = \mathbf{J}\, \mathbf{H}\, \mathbf{J}^\top + (\tfrac{\partial}{\partial \mathbf{y}} \mathbf{J}) \cdot \mathbf{g} \tag{12}$$

These can be used directly in the Newton update rule $\boldsymbol{\delta}\mathbf{y} = -\mathbf{H}_\mathbf{y}^{-1}\, \mathbf{g}_\mathbf{y}$. In particular, if we eliminate one $\mathbf{x}$-variable — say x_n so that $\mathbf{y} = (x_1, ..., x_{n-1})$ and $x_n = x_n(\mathbf{y}, t)$ — we have:

$$\mathbf{J} = \left(\mathbf{I} \,|\, \tfrac{\partial x_n}{\partial \mathbf{y}} \right), \qquad \mathbf{g}_\mathbf{y} \;=\; \tfrac{\partial f}{\partial \mathbf{y}} + g_n\, \tfrac{\partial x_n}{\partial \mathbf{y}}, \qquad \mathbf{H}_\mathbf{y} \;=\; \mathbf{J}\, \mathbf{H}\, \mathbf{J}^\top + g_n\, \tfrac{\partial^2 x_n}{\partial \mathbf{y}^2} \tag{13}$$

To save optimization work and for convergence testing and step length control, it is useful to be able to extrapolate the position and value of the next minimum from existing values. This can be done, *e.g.*, by linear extrapolation from two previous positions, or analytically by solving the constrained minimum state update equations $(\mathbf{g}{+}(\lambda{+}\delta\lambda)\mathbf{g}_c)(\mathbf{x}{+}\boldsymbol{\delta}\mathbf{x}) \;=\; \mathbf{0}$ or $\mathbf{g}_\mathbf{y}(\mathbf{y}{+}\boldsymbol{\delta}\mathbf{y}, t{+}\delta t) \;=\; \mathbf{0}$ to first order, assuming that $\mathbf{x}, t$ is already a minimum and $t \to t{+}\delta t$:

$$(\boldsymbol{\delta}\mathbf{x},\; \delta\lambda) \;=\; \tfrac{1}{\mathbf{g}_c^\top\, \mathbf{H}_\lambda^{-1}\, \mathbf{g}_c} \left(\mathbf{H}_\lambda^{-1}\, \mathbf{g}_c,\; -1 \right)\, \delta t \tag{14}$$

$$\boldsymbol{\delta}\mathbf{y} \;=\; -\mathbf{H}_\mathbf{y}^{-1} \left(\tfrac{\partial \mathbf{J}}{\partial t}\, \mathbf{g} + \mathbf{J}\, \mathbf{H}\, \tfrac{\partial \mathbf{x}}{\partial t} \right)\, \delta t \qquad \boldsymbol{\delta}\mathbf{x} \;=\; \mathbf{J}\, \boldsymbol{\delta}\mathbf{y} + \tfrac{\partial \mathbf{x}}{\partial t}\, \delta t, \tag{15}$$

Standard Taylor expansion of $f(\mathbf{x}{+}\boldsymbol{\delta}\mathbf{x}(t))$ then gives:

$$f_c(t{+}\delta t) \;\approx\; f_c(t) + f_c'\, \delta t + \frac{1}{2} f_c''\, \delta t^2 \tag{16}$$

with $f_c' \;=\; \mathbf{g}\, \frac{\boldsymbol{\delta}\mathbf{x}}{\delta t}$ and $f_c'' \;=\; \frac{\boldsymbol{\delta}\mathbf{x}}{\delta t}^\top \mathbf{H}\, \frac{\boldsymbol{\delta}\mathbf{x}}{\delta t}$. For step length control, we can either fix δt and solve for $\boldsymbol{\delta}\mathbf{x}$ or $\boldsymbol{\delta}\mathbf{y}$ (and hence $\mathbf{x}{+}\boldsymbol{\delta}\mathbf{x} \equiv \mathbf{x}(\mathbf{y}{+}\boldsymbol{\delta}\mathbf{y}, t{+}\delta t)$), or fix a desired trust region for $\boldsymbol{\delta}\mathbf{x}$ or $\boldsymbol{\delta}\mathbf{y}$ and work backwards to find a δt giving a step within it.

6.2 Implementation Details

The hyper-ellipsoid sweeping method that we used in the experiments below is summarized in fig. 2. We start at a local minimum and use centered, curvature-eigenbasis-aligned ellipsoidal hypersurfaces flattened along one eigendirection, say the e^{th}. This restricts the initial search to an eigendirection (the e^{th}). This limitation could easily be removed, but it gives a convenient, not-too-large

Hyper-ellipsoid Sweeping Transition State Search

1. Initialization
Given initial minimum $\mathbf{x}_0$ with Hessian $\mathbf{H}$, eigen-decompose $\mathbf{H}$ to $(\lambda_i, \mathbf{v}_i)$ with principal radii $\sigma_i = 1/\sqrt{\lambda_i}$. Choose an initial search eigen-direction e. Shrink σ_e by say 20% and prepare to eliminate x_e. Set initial step $\mathbf{x}_1 = \mathbf{x}_0 + t_1\sigma_e\mathbf{v}_e$ where t_1 is say 3. Go to step 2.B.

2. Loop, Updating Hypersurface and Minimizing
A. k=k+1. Estimate an initial $\mathbf{x}_k$ by linear extrapolation to the trust radius. Compute the resulting t_k.
B. Minimize f on the t_k ellipsoid to get $f_c(t_k)$: $\mathbf{y}_k = \arg\min_{\mathbf{y}} f(\mathbf{x}_k(\mathbf{y}, t_k))$.
C. Compute $f_c' = \frac{\partial}{\partial t} f_c(t_k)$. If $f_c' < \epsilon$ we are near or past saddle: go to step 3.A. Otherwise go to step 2.A.

3. Line Search for Transition State Refinement
A. If $|f_c'| < \epsilon$, exit.
B. k=k+1. Estimate t_{saddle} by linear interpolation of last two f_c' values.
B. Optimize $\mathbf{y}_k$ as in step 2.B and go to step 3.A.

Figure 2: Our hyper-ellipsoid sweeping algorithm for transition state search.

set of directions to try. All calculations are performed in eigen-coordinates and the minimum is tracked using variable elimination (13) on x_e. In eigen-coordinates, the on-hypersurface constraint becomes:

$$\sum_i (x_i/\sigma_i')^2 = t^2 \tag{17}$$

where the σ_i' are the principal standard deviations, except that the e^{th} (eliminated) one is shrunk by say 20%. Taking $\mathbf{y} = (x_1, ..., x_{e-1}, x_{e+1}, ..., x_n)$ and solving for x_e gives:

$$x_e(\mathbf{y}, t) \;=\; \pm\sigma_e'\sqrt{t^2 - \textstyle\sum_{i\neq e}(x_i/\sigma_i')^2} \tag{18}$$

The necessary derivatives are easily found. We use an $\mathbf{x}$-based trust region to choose the time step. First we estimate an initial $\mathbf{x}_{k+1}$ by linear extrapolation:

$$\mathbf{x}_{k+1} \;\approx\; \mathbf{x}_k + r\,\frac{\mathbf{x}_k - \mathbf{x}_{k-1}}{\|\mathbf{x}_k - \mathbf{x}_{k-1}\|} \tag{19}$$

where r is a trust region radius for $\boldsymbol{\delta}\mathbf{x}$, then we solve for the corresponding t_{k+1} using the ellipsoid constraint, and finally we optimize $\mathbf{x}_{k+1}$ on this ellipsoid.

In our target application there are also underlying equality or bound constraints on the model. These are handled simply by including them as additional constraints in the hypersurface minimization.

7 2D Examples

This section illustrates our eigenvector tracking and hypersurface sweeping methods on the **Müller potential**, a very simple 2D example that is often used to demonstrate such methods in computa-

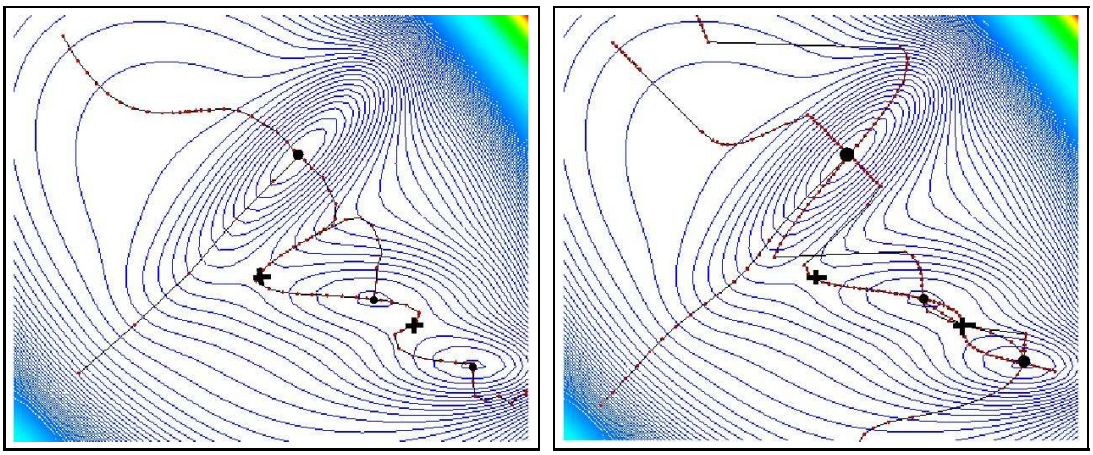


Figure 3: Trajectories for the eigenvector following (left) and hyper-ellipsoid sweeping (right) algorithms on the Müller cost surface, initialized along the $\pm$ eigendirections of the 3 minima (the position of minima is shown with dots, the saddles are shown with crosses).

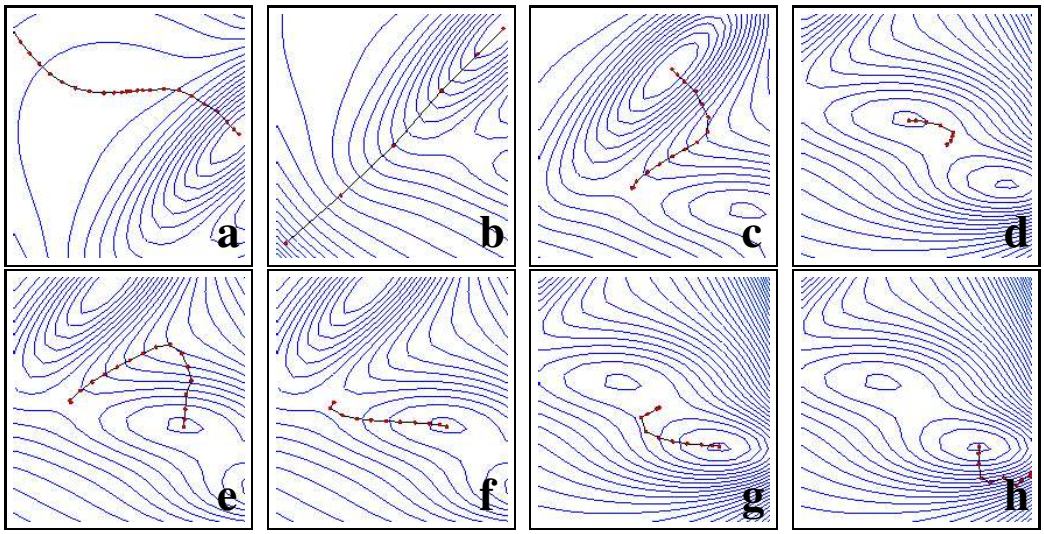


Figure 4: **(a)-(h)** Trajectories for eigenvector following on the Müller cost surface, started from different minima along different principal curvature directions. Note that the trajectories **(a), (b)** and **(h)** do not converge — in fact, no saddles exist in those regions of the space.

tional chemistry. The Müller potential has the form

$$V(x,y) = \sum_{i=1}^{4} A_i \, e^{a_i(x-x_i)^2 + b_i(x-x_i)(y-y_i) + c_i(y-y_i)^2} \tag{20}$$

where $\mathbf{A} = (-200, -100, -170, 15)$, $\mathbf{a} = (-1, -1, -6.5, 0.7)$, $\mathbf{b} = (0, 0, 11, 0.6)$, $\mathbf{c} = (-10, -10, -6.5, 0.7)$, $\mathbf{x} = (1, 0, -0.5, -1)$, $\mathbf{y} = (0, 0.5, 1.5, 1)$. It has three local minima M_1, M_2, M_3 separated by two saddle points S_1, S_2. The minima are separated by around one length unit, and the transition states are around 100–150 energy units above the lowest minimum. See fig. 3 for a concise summary of results for the Eigenvector tracking (ET) and Hypersurface Sweeping (HS)

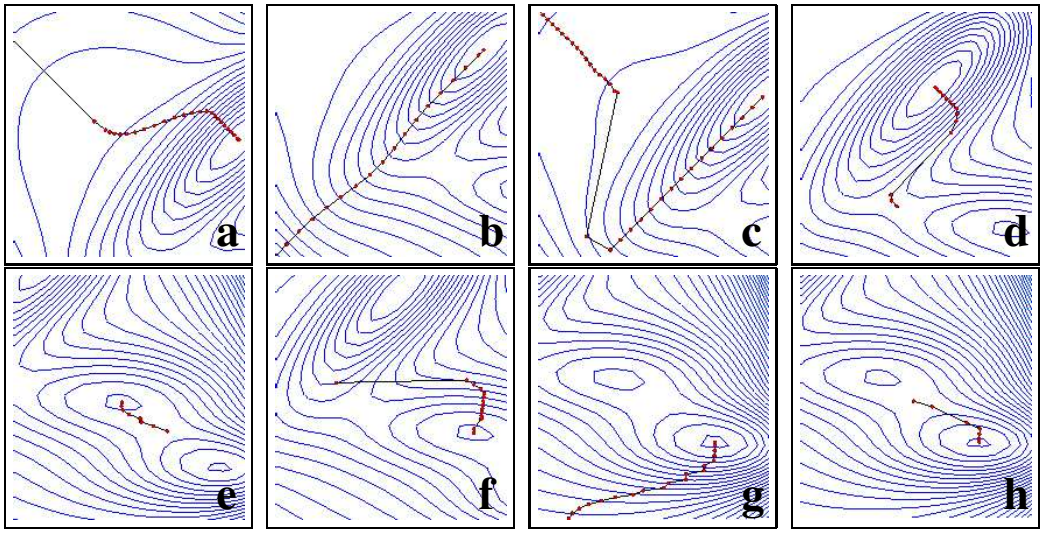


Figure 5: **(a)-(h)** Trajectories for hypersurface sweeping on the Müller cost surface, started from different minima along different principal curvature directions. The trajectories **(a), (b), (c)** and **(g)** do not converge to saddles. **(b)** and **(c)** use the same initial search direction but different ellipsoid flattening factors. **(c)** is flattened less than **(b)**, which allows a 'sawtooth' transition to occur, so that the track jumps abruptly northwards to a new within-hypersurface minimum. **(f)** shows another example — the saddle is crossed in an abrupt jump, without passing through the transition state (but the subsequent minimum will later be found, as required). These effects are intrinsic: they are *not* caused by discretization, step size effects, *etc*.

methods. In the figure, the position of minima is shown with dots and that of the saddles with crosses.

Figs 4 and 5 respectively plot the trajectories of the eigenvector tracking and hypersurface sweeping methods, with initial search along all four principal curvature directions of each minimum. The hypersurface sweeping algorithm is also run for extended trajectories through several saddles and minima in fig. 6. In this simplistic example all of the minima and transition states can actually be found by a single sweep, but this is unusual in more complex problems.

8 Monocular Human Pose Reconstruction and Inter-Frame Tracking

In this section, we apply the transition state location and local minima mapping methods to the difficult problem of 3D articular human tracking from monocular image sequences.

8.1 Previous Work on Monocular Human Tracking

There is a large literature on human motion tracking, but relatively few works tackle the 3D-from-monocular case, where local minima are particularly troublesome owing to poor control of the depth degrees of freedom. We will mention only a few works that make contributions to the search for local minima problem in the context of generative 3D models. Deutscher *et al* use an annealed sampling method and multiple cameras to widen the search and limit the presence of spurious minima [12]. (Their sampling procedure resembles Neal's [32], except that Neal also includes an addi-

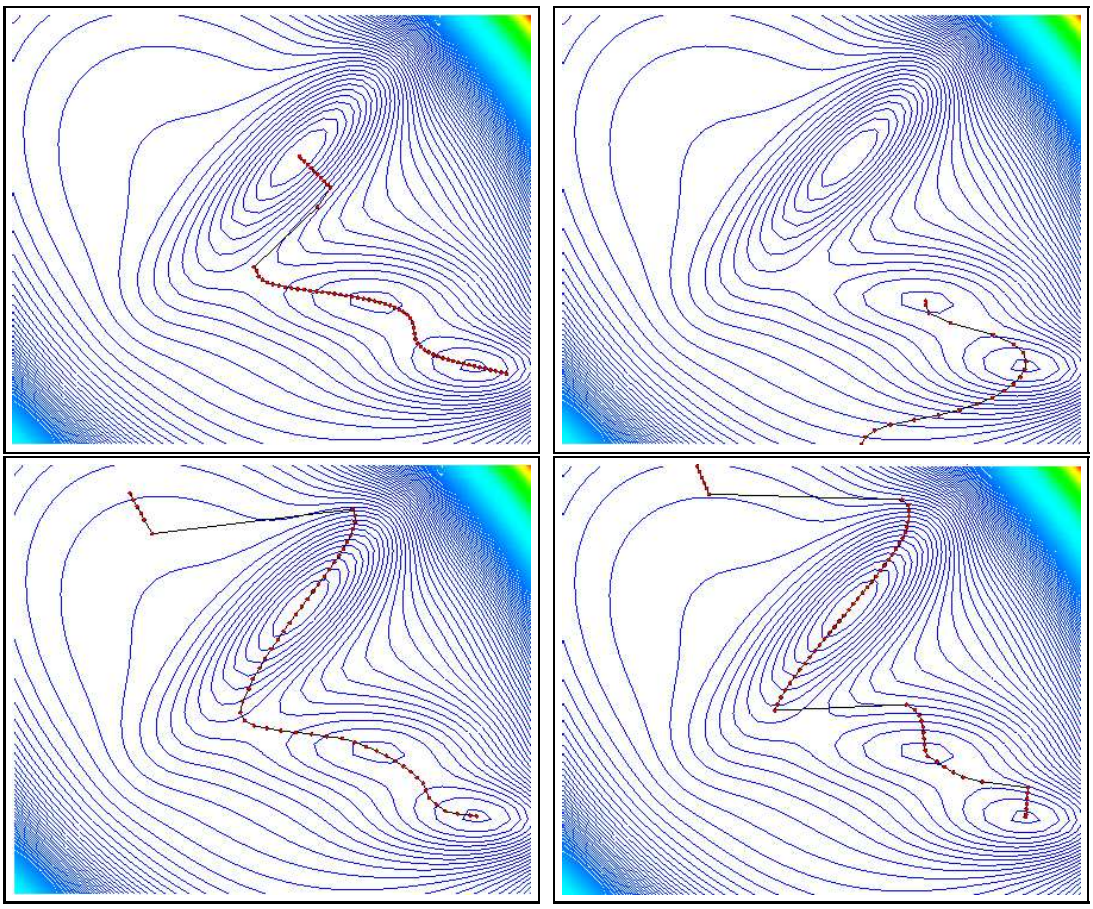


Figure 6: Trajectories for hypersurface sweeping on the Müller cost surface, for trajectories started in different minima, but not stopped after the first saddle detection. Several saddles and minima are found in each sweep.

tional importance sampling correction designed to improve mixing). During annealing, the search for parameters is driven by noise proportional with their individual variances [13]. Considered as an improved (implicit) search space decomposition mechanism, an early method of this type was proposed by Gavrila & Davis [16] to efficiently sample partial kinematic chains. Adaptively identifying and sampling parameters with high variance is useful, but kinematic parameters usually have quite strong interactions that make simple axis-aligned sampling questionable. For monocular reconstruction, it is important to realize that the principal axes of uncertainty change drastically depending on the viewing direction (as noted for instance in [43, 40, 46]). Sidenbladh *et al* use an intensity based cost function and focus search in the neighborhood of known trajectory pathways by particle filtering with importance sampling based on either a learned walking model or a database of motion snippets [36, 37]. Choo & Fleet [9] combine particle filtering and hybrid Monte Carlo sampling to estimate 3D human motion, using a cost function based on joint re-projection error given input from motion capture data. Sminchisescu & Triggs [43, 40, 46] argue that an effective random sampler must combine all three of cost-surface-aware covariance scaling, a sampling distribution with widened tails for deeper search, and local optimization (because deep samples usually have very high costs, and hence will not be resampled even if they ultimately lead to other minima). All

of these works note the difficulty of the multiple-minimum problem and attempt to develop techniques or constraints (on the scene, motion, number of cameras or background) to tackle it. In a complementary approach to the one presented here, Sminchisescu & Triggs [45] proposed MCMC procedures that actively modify the effective cost function to stochastically drive the samples towards transition states, thus reducing the trapping effects that plague classical MCMC sampling. The cost modifications use gradient and curvature information to focus the search on local neighborhoods that have transition-state-like characteristics. Sminchisescu *et al* [48] give modified MCMC methods that speed up sampling from the equilibrium distribution by including long-range jumps based on prior knowledge of the function's dominant minima (which could be provided, *e.g.*, by the algorithms presented in this paper). Methods that search thoroughly over the static model-image matching cost are an effective basis for very general trackers. Sminchisescu & Jepson [42] propose a smoothing algorithm that computes multiple plausible trajectories for weakly identifiable non-linear dynamical systems (like the ones resulting from 3d human modeling and tracking) where the multiplicity of solutions in the model-image matching cost persists over temporal states. Therefore, the trajectory distribution remains multimodal after both filtering and smoothing. Ambiguities can be resolved to a certain extent using prior knowledge. Sminchisescu & Jepson [41] use joint angle training sets typical of various application domains to learn constrained low-dimensional generative models using non-linear embedding. Because the learned representations are global and continuous, methods like the ones proposed here can be used for efficient search in a lower-dimensional space.

8.2 Human Modeling

Here, we very briefly review the human model, priors, and image features that we use for the below experiments. For more details, see [43, 40, 46].

Representation $\mathbf{x}$: The 3D body model used in the human pose and motion estimation experiments here consists of a kinematic 'skeleton' of articulated joints controlled by angular joint parameters, covered by a 'flesh' built from superquadric ellipsoids with additional global deformations. For the experiments here we estimate typically 32 joint parameters.

Observation Likelihood $p(\mathbf{r}|\mathbf{x})$: Robust model-to-image matching cost metrics are evaluated for each predicted image feature, and the results are summed over all observations to produce the image contribution to the parameter space cost function. We use a robust combination of extracted-feature-based metrics and intensity-based ones such as optical flow and robustified normalized edge energy. We also give results for a simpler joint correspondence likelihood designed for model initialization, based on squared distances between reprojected model joints and their specified image positions.

Prior Constraints $p_s(\mathbf{x})$: The model incorporates both hard constraints (for joint angle limits) and soft priors for collision avoidance between body parts. These ensure the parameter estimates remain in the feasible region.

Estimation: We apply Bayes rule and minimize the total negative log-likelihood posterior probability, to give multiple locally MAP parameter estimates:

$$\log p(\mathbf{x}|\mathbf{r}) \;\propto\; \log p_s(\mathbf{x}) + \log p(\mathbf{r}|\mathbf{x}) \;=\; \log p_s(\mathbf{x}) - \sum e(\mathbf{r}_i|\mathbf{x}) \tag{21}$$

Here, $p_s(\mathbf{x})$ is the prior on the model parameters that ensures feasibility, $e(\mathbf{r}_i|\mathbf{x})$ is the cost density associated with observation i, and the sum is over all observations (assumed independent).

Equation (21) gives the model likelihood in a single image, under the model constraints but without initial state or temporal priors[3]. Adopting the Bayesian tracking framework, the temporal prior at time t is determined by the previous posterior $p(\mathbf{x}_{t-1}|\mathbf{R}_{t-1})$ and the system dynamics $p_d(\mathbf{x}_t|\mathbf{x}_{t-1})$, where we have collected the observations at time t into vector $\mathbf{r}_t$ and defined $\mathbf{R}_t = \{\mathbf{r}_1, \ldots, \mathbf{r}_t\}$. The posterior at t becomes:

$$p(\mathbf{x}_t|\mathbf{R}_t) \propto p(\mathbf{r}_t|\mathbf{x}_t)\, p_s(\mathbf{x}_t) \int_{\mathbf{x}_{t-1}} p_d(\mathbf{x}_t|\mathbf{x}_{t-1})\, p(\mathbf{x}_{t-1}|\mathbf{R}_{t-1}) \tag{22}$$

Together $p_d(\mathbf{x}_t|\mathbf{x}_{t-1})$ and $p(\mathbf{x}_{t-1}|\mathbf{R}_{t-1})$ form the time t prior $p(\mathbf{x}_t|\mathbf{R}_{t-1})$ for the image correspondence search (21). The integral on the r.h.s. of (22) is approximated as a mixture distribution of MAP estimates given by the local optima and their covariances from (21). See [46] for details on the approximation and [48] for methods to speed-up MCMC sampling from the equilibrium distribution using long-range jumps based on prior knowledge about minima structure.

8.3 Human Pose Estimation and Inter-Frame Tracking Experiments

Here we show examples from a set of experiments on locating local minima for pose estimation and inter-frame tracking in monocular images, using cost surfaces based on various different combinations of image cues and a 32 d.o.f. articulated human body model. The figures show examples of minima found in likelihood models based on image contours and optical flow (fig. 9), contours and silhouette-image data (fig. 11 – note that some of these minima can be removed using more complex silhouette-based cost functions, as proposed in [39]), and model-to-image joint correspondences (figs 12 and 13). We also give more extensive quantitative results in table 1. In each case, the model was initialized in a minimum found using the local optimization algorithm from [43, 40, 46], and transition state searches were initiated along its principal curvature directions. For each transition state found, local descent [43, 40, 46] was used to find the corresponding neighboring minimum. The algorithm was globalized by repeating the process from the new minima found (checking for duplicates), until either no new minima are found or the permitted total run time has elapsed.

Fig. 7 captures some more quantitative information about the methods, here for the joint correspondence cost function (see §8.2). This problem is well adapted to illustrating the algorithm, as its cost surface is highly multimodal. Of the 32 kinematic d.o.f., about 10 are subject to 'reflective' ambiguities (forwards *vs*. backwards slant in depth). This potentially creates around $2^{10} = 1024$ local minima in the cost surface [27], although some of these are physically infeasible and hence forbidden in any constraint-consistent optimization. Indeed, given the large number of minima that exist, we find that it is very difficult to ensure initialization to the 'correct' pose with this kind of data. The first row in fig. 7 displays the parameter space and cost distances of the 56 minima found during a set of 64 constrained searches (the $\pm$ directions of the 32 eigenvectors of an initial minimum, distances being measured w.r.t. this minimum, in radians and meters for the parameter space). The second row again shows parameter space distances, but now measured in standard deviations and for saddles rather than minima, for the same frontal view and for a slightly more side-on one (fig. 12 and fig. 13). The plots reveal the structure of the cost surface, with nearby saddles at 4–10 standard deviations and progressively more remote ones at 20–50, 80–100 and 150–200 standard deviations. It follows that no multiple-minimum exploration algorithm can afford to search only within the 'natural' covariance scale of its current minima: significantly deeper sampling is needed to capture even nearby minima (as previously noted, *e.g*. by [43, 46]).

[3] Local optima of the current observation cost are obtained using the prior initialization followed by robust constraint-consistent local optimization, as described in [43, 40, 46].

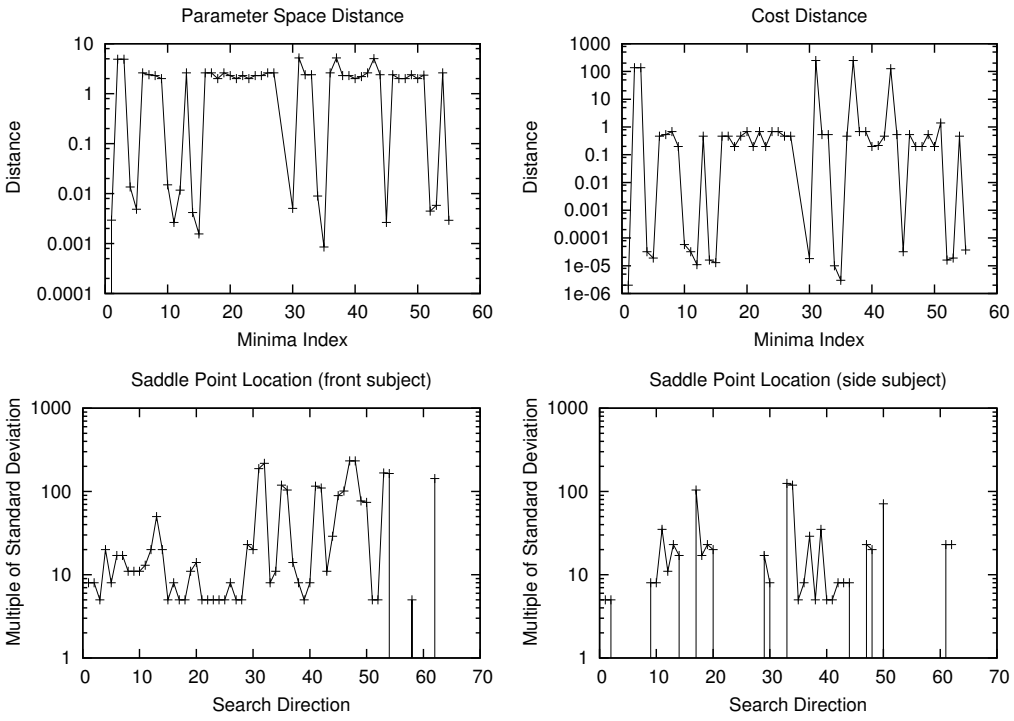


Figure 7: Summary of typical search results from a given minimum along ± 32-eigendirections. Top row: parameter space distance and cost difference between the initial minimum and its discovered neighbors. Second row: initial minimum to transition state distances in standard deviations for a frontal pose with hypersurface sweeping, and a partly side-on one with eigenvector tracking. (See figs 12 and 13 for the corresponding for visual results).

In fig. 8 we show sample cost profiles for typical runs of joint-constrained eigenvector following (top) and hypersurface sweeping (bottom) search. In the eigenvector method, it is preferable to represent the joint limits using a 'hard' active set strategy by projecting the tracking direction onto the active constraint set (row 1 right) rather than soft constraints (row 1 left): the stiff 'cost walls' induced by the soft constraints tend to force the eigenvector follower into head-on collision with the wall, with the cost climbing rapidly to infinity. The active set strategy avoids this problem at the price of more frequent direction changes as the joint limits switch on and off (row 1 right). The hyper-ellipsoid method (row 2) produces trajectories that do not require special joint limit processing, but its cost profiles have characteristic sawtooth edges (row 2 right) associated with sudden state readjustments on the hypersphere at points where the tracked minimum becomes locally unstable[4].

Figs 9 and 11 show minima for costs based on various combinations of image cues. In fig. 9 the minima correspond to a small interframe motion, using contour and robust optical flow information. This case has relatively few, but closely spaced local minima owing to the smoothing/quadratic effect of the flow. (Remoter minima do still exist at points where the robust contributions of sets

[4]Note the different sources of trajectory instabilities in the two methods: in eigenvector tracking they are due to the projection of the current trajectory step onto the joint-constraint surface, while for hypersurface sweeping, the joint constraints can be enforced alongside the hypersurface one but there are instabilities due to topological transitions of within-constraint minima.

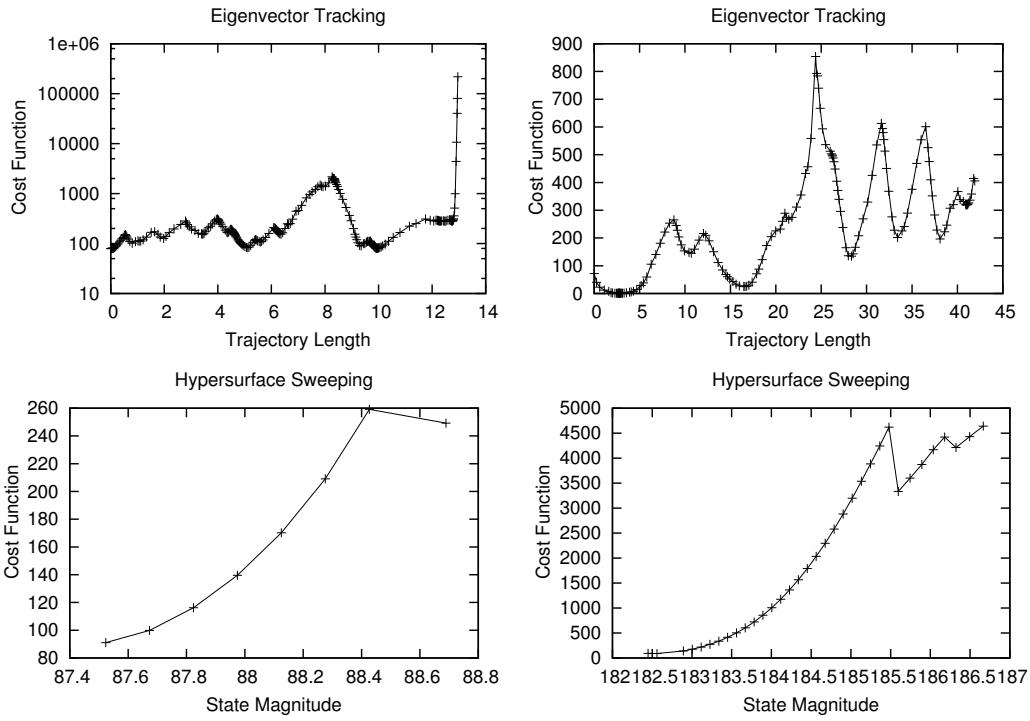


Figure 8: Top row: typical cost profiles for eigenvector tracking with soft (left) and hard (right) joint angle constraints. Soft constraints often lead to 'wall-climbing' divergences when the search hits a joint limit. The active-set projection strategy used in the hard-constraint tracker alleviates this, but introduces abrupt variations of the path direction whenever a constraint is hit, which cause the cost derivative to vary abruptly. Second row: The hyper-ellipsoid method does not require special joint limit processing — the joint constraints can just be included with the hypersurface constraint — but its cost profiles have characteristic 'sawtooth' transitions whenever the tracked minimum becomes locally unstable on the hypersurface.

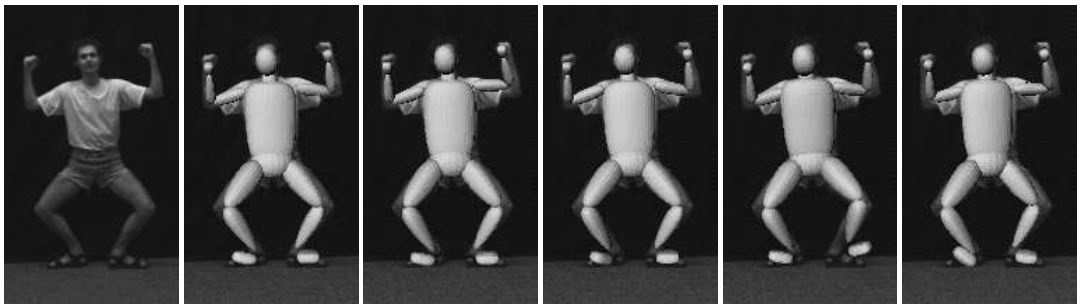
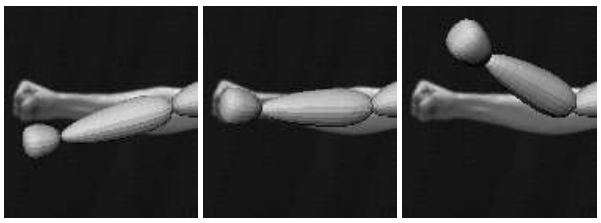


Figure 9: Minima of image-based cost functions: contour and optical flow likelihood. The model is initialized in one minimum (second figure), and search trajectories for other minima (along different principal curvature directions) are initiated from there. The other figures show some of the other minima found. These correspond to incorrect contour assignments or to configurations where the intensity robustifiers turn off (see text).



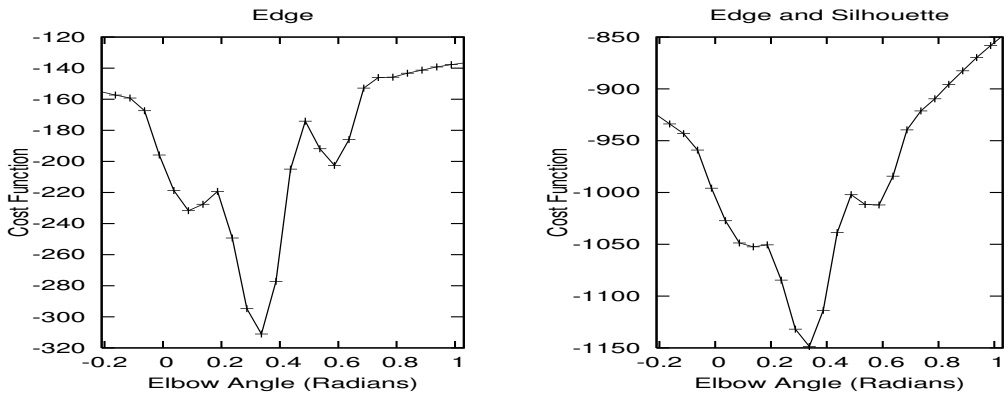


Figure 10: Cost Function Experiment. The elbow joint is varied as shown in the top row, and the corresponding edge cost is monitored. The corresponding cost function profile is multi-modal, with characteristic 'shoulders' in which one edge of the image limb contributes to the matching cost of the opposite edge of the model limb. These persist even when further visual cues (here a silhouette term) are fused into the cost.

of flow measurements turn off, particularly when these coincide with incorrect edge assignments). Fig. 11 shows minima arising from a silhouette and edge based cost function. To illustrate why such minima occur, we first run a toy experiment [39], that moves a model forearm over an image forearm — see fig. 10. As one can see in the bottom row, the cost has multiple minima, essentially owing to the attribution of both model edges to different sections of the same image edge. Fig. 11 shows how such incorrect limb assignments affect the full model. The minima shown include 'reflective' (depth-related) ambiguities, incorrect edge assignments and singular 'inside-silhouette' configurations (some of these can be alleviated to some extent by augmenting the contour and silhouette likelihood terms as in [39]).

Figs 12 and 13 show depth/pose ambiguities for frontal and half-profile views, under the model to image joint correspondence cost function (which is not subject to image matching ambiguities). The flexible and hard-to-observe arm-shoulder complex tends to induce more minima than the legs. We also find that profile views (fig. 13) tend to generate fewer minima than frontal ones (fig. 12), perhaps due to presence of body-part non-self-intersection and joint constraints that render many 'purely reflective' minima infeasible.

Finally, to provide a more quantitative evaluation of the minimum finding efficiency of the eigenvector-tracking and hypersurface-sweeping algorithms, we selected 6 initial minima for a frontal view and 6 more for a half-profile view, using the joint correspondence cost function and our 32 d.o.f. body model. Starting from these configurations, we initiated transition state searches along

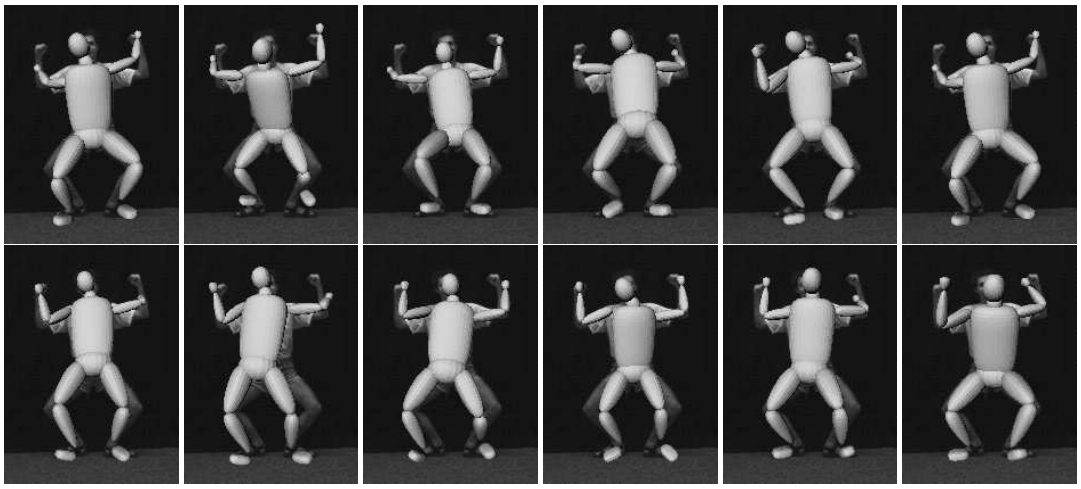


Figure 11: Minima of image-based cost functions: contour and silhouette likelihood. The model is initialized at one minimum (fig. 9, second figure) and search trajectories for other minima are initiated from there. The minima shown include 'reflective' (depth-related) ambiguities, incorrect edge assignments and singular 'inside-silhouette' configurations. More complex silhouette-based cost functions, as proposed in [39] can be further used eliminate some of the latter spurious minima (see text).

the ± 32-eigendirections at each minimum, giving a total of $12 \times 64 = 768$ search trials for each algorithm. In this experiment we also set an upper-bound of 50 iterations per search (the search is counted as a failure if a saddle has not been found at this point). Table 1 reports the number of minima found by each method, and the medians and standard deviations of their parameter space distances and function values. Both methods locate neighboring minima with good success rates. In this experiment, hypersurface sweeping locates a few more minima than eigenvector tracking (60% *vs*. 55%), but also finds slightly more remote and higher cost minima. When successful, the eigenvector tracking method typically needs around 17–19 iterations to locate a transition state, whereas hypersurface sweeping needs about 14–16. In contrast, once a transition state is located, local descent finds the neighboring minimum in around 7–10 iterations.

Computationally, the cost of an iteration is determined by function evaluation costs and by the cost of Hessian manipulation for each method. These in turn depend on the number of measurements and the dimension of the parameter space, respectively. ET needs eigendecomposition of the Hessian which is about 4-5 times more expensive than the Cholesky decomposition required for a local descent iteration. HS needs Hessian eigendecomposition only for the initial selection of the hypersurface shape, and any subsequent step involves a local optimization that in practice, we found, converged in about 3-5 iterations. For the articulated pose problem studied here, we also found that the above differences are dominated by the cost function evaluation, the manipulation of the Hessian being comparatively fast for 32 dimensions. For example, on a 2.2Ghz Pentium processor, the average iteration cost for the simple joint correspondence cost function was about 0.14s for local descent, and about 0.31s for ET and 0.62s for HS. For more expensive image-based cost functions, the timings were 1.21s for local descent, 1.67s for ET and 4.42s for HS.

Figure 12: 'Reflective' kinematic ambiguities under the model/image joint correspondence cost function in a frontal view. The model is initialized at one minimum (second row, leftmost figure), and search trajectories are initiated for other minima along different principal curvature directions. The images show some of the new minima found, from the original camera viewpoint (superposed on the source image) and from a synthetic overhead viewpoint. Note the pronounced forwards-backwards character of these reflective minima, and the large parameter space distances that often separate them.

9 Conclusions and Research Directions

This paper has described two classes of deterministic, local optimization based algorithms for finding 'transition states' (saddle points with 1 negative eigenvalue), and hence neighboring minima, of high-dimensional cost functions with multiple minima. These methods allow us to build topological 'roadmaps' of the nearby local minima and the transition states that lead to them. They are based on methods developed in computational chemistry, but here generalized, clarified and adapted for use in computational vision. Experiments on the difficult problem of articulated 3D human pose from monocular images show that our algorithms can stably and efficiently recover large numbers of nearby transition states and minima, but also serve to underline the very large numbers of minima that exist in this problem.

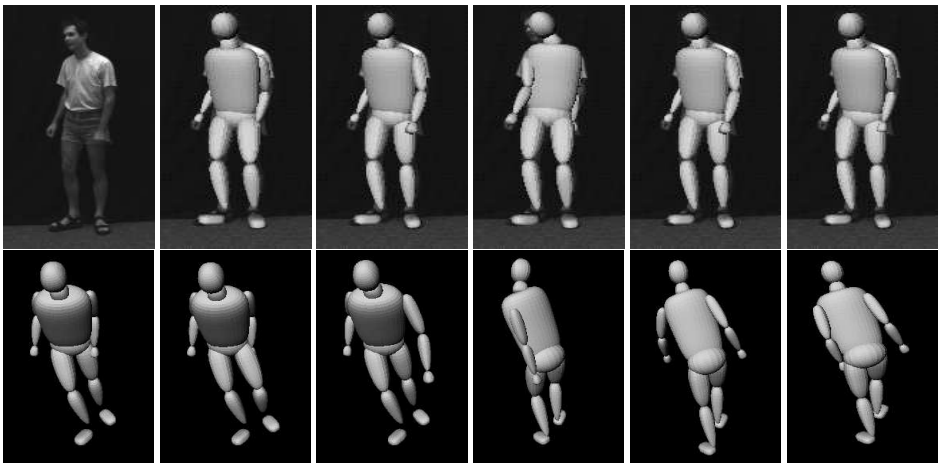


Figure 13: 'Reflective' kinematic ambiguities under the model/image joint correspondence cost function, for a half-profile view. For explanation, see fig. 12 caption.

METHOD	NUMBER OF DETECTED MINIMA	MEDIAN PARAMETER DISTANCE	MEDIAN STANDARD DEVIATIONS	MEDIAN COST
ET	421 (55%)	4.45	79.6	3.39
HS	457 (60%)	5.01	91.6	3.87

Table 1: Quantitative results for the distribution of minima found by eigenvector tracking and hypersurface sweeping. The results represent a total of 768 local searches (forwards and backwards along the 32-eigendirections of 6 frontal and 6 half-profile minima). Hypersurface sweeping finds 9% more minima than eigenvector tracking, but they are about 12–15% further away and have about 14% higher cost on average.

Our methods should be useful for many other minimum-rich problems in vision, including structure from motion. They could also potentially be used to quantify the degree of ambiguity of different cost functions, which in the long term may aid the design of less ambiguous cost functions based on higher-level features and groupings.

Although the current methods are a great improvement over previous ones, they are not foolproof and much remains to be done in this area. Damped Newton iteration is useful for refining estimated saddle points but its convergence domain is too limited for general use. Eigenvector tracking extends the convergence domain but can be sensitive to the 'same eigenvector' heuristic used. Hypersurface sweeping is better founded in that it provides some guarantee of global progress, but no single sweep finds all saddle points and it is more complex to implement. Future research directions include deriving heuristics to select promising up-hill directions for search initialization, and an analysis of the benefits of different types of hypersurfaces and of ways to adaptively evolve them based on the local cost function structure. It should be also interesting to explore the use of local saddle point moves as alternatives to less informed long range Markov-chain steps in stochastic simulations.

Acknowledgements

This work was begun while the first author was a PhD student at INRIA Rhône-Alpes. It was supported by an Eiffel Scholarship, the EU R&D project VIBES. We thank Alexandru Telea at Eindhoven University of Technology, for generous help with the visualization tools used to display the results in this paper.

References

[1] Y. Abashkin and N. Russo. Transition State Structures and Reaction Profiles from Constrained Optimization Procedure. Implementation in the Framework of Density Functional Theory. *J. Chem. Phys.*, 1994.

[2] Y. Abashkin, N. Russo, and M. Toscano. Transition States and Energy Barriers from Density Functional Studies: Representative Isomerization Reactions. *International Journal of Quantum Chemistry*, 1994.

[3] N. Anderson and G. R. Walsh. A Graphical Method for a Class of Branin Trajectories. *Journal of Optimization Theory and Applications*, 1986.

[4] G. T. Barkema. Event-Based Relaxation of Continuous Disordered Systems. *Physical Review Letters*, 77(21), 1996.

[5] J. M. Bofill. Updated Hessian Matrix and the Restricted Step Method for Locating Transition Structures. *Journal of Computational Chemistry*, 15(1):1–11, 1994.

[6] F. Branin and S. Hoo. A Method for Finding Multiple Extrema of a Function of n Variables. *Numerical Methods of Nonlinear Optimization*, 1972.

[7] C. J. Cerjan and W. H. Miller. On Finding Transition States. *J. Chem. Phys.*, 75(6), 1981.

[8] A. Chiuso, R. Brockett, and S. Soatto. Optimal Structure from Motion: Local Ambiguities and Global Estimates. *International Journal of Computer Vision*, 39(3):195–228, 2000.

[9] K. Choo and D. Fleet. People Tracking Using Hybrid Monte Carlo Filtering. In *IEEE International Conference on Computer Vision*, 2001.

[10] G. M. Crippen and H. A. Scheraga. Minimization of Polypeptide Energy. XI. The Method of Gentlest Ascent. *Archives of Biochemistry and Biophysics*, 144:462–466, 1971.

[11] P. Culot, G. Dive, V. H. Nguyen, and J. M. Ghuysen. A Quasi-Newton Algorithm for First-Order Saddle Point Location. *Theoretica Chimica Acta*, 82:189–205, 1992.

[12] J. Deutscher, A. Blake, and I. Reid. Articulated Body Motion Capture by Annealed Particle Filtering. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2000.

[13] J. Deutscher, A. Davidson, and I. Reid. Articulated Partitioning of High Dimensional Search Spacs associated with Articulated Body Motion Capture. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2001.

[14] I. Fiodorova. Search for the Global Optimum of Multiextremal Problems. *Optimal Decision Theory*, 1978.

[15] R. Fletcher. Practical Methods of Optimization. In *John Wiley*, 1987.

[16] D. Gavrila and L. Davis. 3-D Model Based Tracking of Humans in Action:A Multiview Approach. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 73–80, 1996.

[17] R. Ge. The Theory of the Filled Function Method for Finding a Global Minimizer of a Nonlinearly Constrained Minimization Problem. *J. of Comp. Math.*, 1987.

[18] F. Glover. Tabu search - part II. *ORSA Journal of Computing 2*, pages 4–32, 1990.

[19] A. Goldstein and J. Price. On Descent from Local Minima. *Mathematics of Computation*, 1971.

[20] A. Griewank. Generalized Descent for Global Optimization. *Journal of Optimization Theory and Applications*, 1981.

[21] T. Helgaker. Transition-State Optimizations by Trust-Region Image Minimization. *Chemical Physics Letters*, 182(5), 1991.

[22] G. Henkelman and H. Jonsson. A Dimer Method for Finding Saddle Points on High Dimensional Potential Surfaces Using Only First Derivatives. *J. Chem. Phys.*, 111(15):7011–7022, 1999.

[23] R. L. Hilderbrandt. Application of Newton-Raphson Optimization Techniques in Molecular Mechanics Calculations. *Computers & Chemistry*, 1:179–186, 1977.

[24] F. Jensen. Locating Transition Structures by Mode Following: A Comparison of Six Methods on the Ar_8 Lennard-Jones potential. *J. Chem. Phys.*, 102(17):6706–6718, 1995.

[25] P. Jorgensen, H. J. A. Jensen, and T. Helgaker. A Gradient Extremal Walking Algorithm. *Theoretica Chimica Acta*, 73:55–65, 1988.

[26] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by Simulated Annealing. *Science*, 1983.

[27] H. J. Lee and Z. Chen. Determination of 3D Human Body Postures from a Single View. *Computer Vision, Graphics and Image Processing*, 30:148–168, 1985.

[28] A. Levy and A. Montalvo. The Tunneling Algorithm for the Global Minimization of Functions. *SIAM J. of Stat. Comp.*, 1985.

[29] D. Morris and J. Rehg. Singularity Analysis for Articulated Object Tracking. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 289–296, 1998.

[30] N. Mousseau and G. T. Berkema. Traveling Through Potential Energy Lanscapes of Desordered Materials: The Activation-Relaxation Technique. *Physical Review E*, 57(2), 1998.

[31] L. J. Munro and D. J. Wales. Defect Migration in Crystalline Silicon. *Physical Review B*, 59(6):3969–3980, 1999.

[32] R. Neal. Annealed Importance Sampling. *Statistics and Computing*, 11:125–139, 2001.

[33] J. Nichols, H. Taylor, P. Schmidt, and J. Simons. Walking on Potential Energy Surfaces. *J. Chem. Phys.*, 92(1), 1990.

[34] J. Oliensis. The Error Surface for Structure from Motion. Technical report, NECI, 2001.

[35] E. M. Sevick, A. T. Bell, and D. N. Theodorou. A Chain of States Method for Investigating Infrequent Event Processes Occuring in Multistate, Multidimensional Systems. *J. Chem. Phys.*, 98(4), 1993.

[36] H. Sidenbladh, M. Black, and D. Fleet. Stochastic Tracking of 3D Human Figures Using 2D Image Motion. In *European Conference on Computer Vision*, 2000.

[37] H. Sidenbladh, M. Black, and L. Sigal. Implicit Probabilistic Models of Human Motion for Synthesis and Tracking. In *European Conference on Computer Vision*, 2002.

[38] J. Simons, P. Jorgensen, H. Taylor, and J. Ozmen. Walking on Potential Energy Surfaces. *J. Phys. Chem.*, 87:2745–2753, 1983.

[39] C. Sminchisescu. Consistency and Coupling in Human Model Likelihoods. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 27–32, Washington D.C., 2002.

[40] C. Sminchisescu. *Estimation Algorithms for Ambiguous Visual Models–Three-Dimensional Human Modeling and Motion Reconstruction in Monocular Video Sequences*. PhD thesis, Institute National Politechnique de Grenoble (INRIA), July 2002.

[41] C. Sminchisescu and A. Jepson. Generative Modeling for Continuous Non-Linearly Embedded Visual Inference. In *International Conference on Machine Learning*, Banff, 2004.

[42] C. Sminchisescu and A. Jepson. Variational Mixture Smoothing for Non-Linear Dynamical Systems. In *IEEE International Conference on Computer Vision and Pattern Recognition*, Washington D.C., 2004.

[43] C. Sminchisescu and B. Triggs. Covariance-Scaled Sampling for Monocular 3D Body Tracking. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 447–454, Hawaii, 2001.

[44] C. Sminchisescu and B. Triggs. Building Roadmaps of Local Minima of Visual Models. In *European Conference on Computer Vision*, volume 1, pages 566–582, Copenhagen, 2002.

[45] C. Sminchisescu and B. Triggs. Hyperdynamics Importance Sampling. In *European Conference on Computer Vision*, volume 1, pages 769–783, Copenhagen, 2002.

[46] C. Sminchisescu and B. Triggs. Estimating Articulated Human Motion with Covariance Scaled Sampling. *International Journal of Robotics Research*, 22(6):371–393, 2003.

[47] C. Sminchisescu and B. Triggs. Kinematic Jump Processes for Monocular 3D Human Tracking. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 69–76, Madison, 2003.

[48] C. Sminchisescu, M. Welling, and G. Hinton. A Mode-Hopping MCMC Sampler. Technical Report CSRG-478, University of Toronto, submitted to *Machine Learning Journal*, September 2003.

[49] J. Q. Sun and K. Ruedenberg. Gradient Extremals and Stepest Descend Lines on Potential Energy Surfaces. *J. Chem. Phys.*, 98(12), 1993.

[50] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle Adjustment - A Modern Synthesis. In Springer-Verlag, editor, *Vision Algorithms: Theory and Practice*, 2000.

[51] A. F. Voter. A Method for Accelerating the Molecular Dynamics Simulation of Infrequent Events. *J. Chem. Phys.*, 106(11):4665–4677, 1997.

[52] A. F. Voter. Hyperdynamics: Accelerated Molecular Dynamics of Infrequent Events. *Physical Review Letters*, 78(20):3908–3911, 1997.

[53] D. J. Wales. Finding Saddle Points for Clusters. *J. Chem. Phys.*, 91(11), 1989.

[54] D. J. Wales and T. R. Walsh. Theoretical Study of the Water Pentamer. *J. Chem. Phys.*, 105(16), 1996.

Fast Mixing Hyperdynamic Sampling

Cristian Sminchisescu
University of Toronto, Department of Computer Science
Toronto, Ontario, M5S 3G4, Canada
crismin@cs.toronto.edu, www.cs.toronto.edu/~crismin
Bill Triggs
GRAVIR-CNRS-INRIA,
655 avenue de l'Europe, 38330 Montbonnot, France
Bill.Triggs@inrialpes.fr, http://lear.inrialpes.fr/people/triggs

Abstract

Sequential random sampling ('Markov Chain Monte-Carlo') is a popular strategy for many vision problems involving multimodal distributions over high-dimensional parameter spaces. It applies both to importance sampling *(where one wants to sample points according to their 'importance' for some calculation, but otherwise fairly) and to* global optimization *(where one wants to find good minima, or at least good starting points for local minimization, regardless of fairness). Unfortunately, most sequential samplers are very prone to becoming trapped for long periods in unrepresentative local minima, which leads to biased or highly variable estimates. We present a general strategy for reducing MCMC trapping that generalizes Voter's 'hyperdynamic sampling' from computational chemistry. The local gradient and curvature of the input distribution are used to construct an adaptive importance sampler that focuses samples on negative curvature regions that are likely to contain low cost 'transition states' (codimension-1 saddle points representing "mountain passes" connecting adjacent cost basins). This substantially accelerates inter-basin transition rates while still preserving correct relative transition probabilities. Experimental tests on the difficult problem of 3D articulated human pose estimation from monocular images show significantly enhanced minimum exploration.*

Keywords: *Hyperdynamics, Markov-chain Monte Carlo, importance sampling, global optimization, human tracking.*

1 Introduction

Many vision problems can be formulated either as global minimizations of highly non-convex cost functions with many minima, or as statistical inferences based on fair sampling or expectation-value integrals over highly multi-modal distributions. Importance sampling is a promising approach for such applications, particularly when combined with sequential ('Markov Chain Monte-Carlo'), layered or annealed samplers [9, 5, 6], optionally punctuated with bursts of local optimization [11, 4, 33]. Sampling methods are flexible, but they tend to be computationally expensive for a given level of accuracy. In particular, when used on multi-modal cost surfaces, current sequential samplers are very prone to becoming caught for long periods in cost basins containing unrepresentative local minima. This 'trapping' or 'poor mixing' leads to biased or highly variable estimates

To appear in *Journal of Image and Vision Computing*, Special Issue on Selected Papers from the European Conference on Computer Vision, ECCV 2002, Copenhagen.

whose character is at best quasi-local rather than global. Trapping times are typically exponential in a (large) scale parameter, so buying a faster computer helps little. Current samplers are myopic mainly because, when judging 'importance', they consider only *immediate local variations* of the size of the integrand being evaluated or the cost being optimized. *For globally efficient estimates, it is also critically important to include an effective strategy for reducing trapping,* e.g. *by explicitly devoting some fraction of the samples to moving between cost basins.*

This paper describes a method that reduces trapping by 'boosting'[1] the dynamics of the sequential sampler. It is based on A.F. Voter's 'hyperdynamics' [45, 46], which was originally developed in computational chemistry to accelerate the estimation of transition rates between different atomic arrangements in atom-level simulations of molecules and solids. There, the dynamics is basically a thermally-driven random walk of a point in the configuration space of the combined atomic coordinates, subject to an effective energy potential that models the combined inter-atomic interactions. The configuration-space potential is often highly multimodal, corresponding to different large-scale configurations of the molecule being simulated. Trapping is a significant problem, especially as the fine-scale dynamics must use quite short time-steps to ensure accurate physical modeling. Mixing times of 10^6–10^9 or more steps are common. In our target applications in vision the sampler need not satisfy such strict physical constraints, but trapping remains a key problem.

Hyperdynamics reduces trapping by enhancing the sampling rate near 'transition states' — low lying saddle points that the system would typically pass through if it were moving thermally between adjacent energy basins. It does this by modifying the cost function, adding a term based on the gradient and curvature of the original potential that raises the cost near the cores of the local potential basins to reduce trapping there, while leaving the cost intact in regions where the original potential has the low gradient and negative curvature eigenvalue characteristic of transition neighborhoods. Hyperdynamics can be viewed as a generalized form of MCMC importance sampling whose importance measure considers the gradient and curvature as well as the values of the original cost function. The key point is not the specific form adopted for the potential, but rather the refined notion of 'importance': deliberately adding samples to speed mixing and hence reduce global bias ('finite sample effects'), even though the added samples are not directly 'important' for the calculation being performed.

Another general approach to multi-modal optimization is *annealing* [15, 24] (with detailed balance variations like tampering [18]) — initially sampling with a reduced sensitivity to the underlying cost ('higher temperature'), then progressively increasing the sensitivity to focus samples on lower cost regions. Annealing has been used many times in vision and elsewhere[2], *e.g.* [23, 24, 6], but although it works well in many applications, it has important limitations as a general method for reducing trapping. The main problem is that it samples indiscriminately within a certain energy band, regardless of whether the points sampled are likely to lead out of the basin towards another minimum, or whether they simply lead further up an ever-increasing potential wall. In many applications, and especially in high-dimensional or ill-conditioned ones, the cost surface has relatively narrow 'windows' connecting adjacent basins, and it is important to steer the samples towards these using local information about how the cost appears to be changing. Hyperdynamics is a first attempt at doing this. Annealing and hyperdynamics are actually complementary: it may be possible to speed up hyperdynamics by annealing its modified potential, but we will not investigate this here.

Paper organization: We review desirable features of fast mixing high-dimensional samplers in

[1] No relationship to boosting in machine learning is implied.

[2] In chemistry and physics applications of hyperdynamics, raising the temperature is often unacceptable as it would significantly change the problem, *e.g.* the solid being simulated might melt...

§1.1 and prior work in §1.2. Sampling and transition state theory are introduced in §2. Ways to design generic fast-mixing transformations of the energy surface are presented in §3 and one particular approximation is proposed and analyzed in §4. Complementary sampling strategies are discussed in §5. Human modeling and high-dimensional pose estimation experiments are given in §6 and §7. Conclusions and ideas for future research are discussed in §8.

1.1 What is a Good Multiple-Mode Sampling Function ?

'The curse of dimensionality' causes many difficulties in high-dimensional search. In stochastic methods, long sampling runs are often needed to hit the distribution's 'typical set' — the areas where most of the probability mass is concentrated. In sequential samplers this is due to the inherently local nature of the sampling process, which tends to become 'trapped' in individual modes, moving between them only very infrequently. More generally, choosing an importance sampling distribution is a compromise between tractable sampleability and efficient focusing of the sampling resources towards 'good places to look'.

There are at least three issues in the design of a good multi-modal sampler: (*i*) *Approximation accuracy*: in high dimensions, when the original distribution is complex and highly multi-modal (as is the case in vision), finding a sampleable function that gives good approximation accuracy / low reject rate can be very difficult, thus limiting the applicability of the method. It is therefore appealing to look for ways of using a modified version of the original distribution, as for instance in annealing methods [23, 24, 6]. (*ii*) *Trapping*: even when the approximation is locally accurate (*e.g.* by sampling the original distribution, thus avoiding any sample-weighting artifacts), most sampling procedures tend to get caught in the mode(s) closest to the starting point of sampling. Very long runs are needed to sample infrequent inter-mode transition events that lie far out in the tails of the modal distributions, but that can make a huge difference to the overall results. (*iii*) *Biased transition rates*: annealing changes not only the absolute inter-mode transition rates (thus reducing trapping), but also their relative sizes [40]. So there is no guarantee that the modes are visited with the correct relative probabilities implied by the dynamics on the original cost surface. This may seem irrelevant if the aim is simply to discover 'all good modes' or 'the best mode', but the levels of annealing needed to make difficult transitions frequent can very significantly increase the number of modes and the state space volume that are available to be visited, and thus cause the vast bulk of the samples to be wasted in fruitless regions[3]. This is especially important in applications like tracking, where temporal continuity implies that only nearby modes that are separated from the current one by low energy barriers need to be recovered.

To summarize, for complex high dimensional problems, finding good, sampleable approximating distributions is hard, so it is useful to look at sequential samplers based on distributions derived from the original one. There is a trade-off between sampling for local computational accuracy, which requires samples in 'important' regions, usually mode cores, and sampling for good mixing, which requires not only more frequent samples in the tails of the distribution, but also that these should be focused on regions likely to lead to inter-modal transitions. Defining such regions is delicate in practice, but it is clear that steering samples towards regions with low gradient and negative curvatures should increase the likelihood of finding transition states (saddle points with one

[3]There is an analogy with the chemist's melting solid, liquids being regions of state space with huge numbers of small interconnected minima and saddles, while solids have fewer, or at least more clearly defined, minima. Also remember that state space volume increases very rapidly with sampling radius in high dimensions, so dense, distant sampling is simply infeasible.

negative curvature direction) relative to purely cost-based methods such as annealing.

1.2 Related Work

In this section we summarize some relevant work on high-dimensional search, especially in the domain of human modeling and estimation. Deutscher *et al* track 3D body motion using a multi-camera silhouette-and-edge based likelihood function and annealed sampling within a temporal particle filtering framework [6]. Their sampling procedure resembles one used by Neal, except that Neal also includes an additional importance sampling correction designed to improve mixing [23, 24]. Sidenbladh *et al* use an intensity based cost function and particle filtering with importance sampling based on a learned dynamical model to track a 3D model of a walking person in an image sequence [29]. Choo & Fleet combine particle filtering and hybrid Monte Carlo sampling to estimate 3D human motion, using a cost function based on joint reprojection error given input from motion capture data [5]. Sminchisescu & Triggs recover articulated 3D motion from monocular image sequences using an edge and intensity based cost function, with a combination of robust constraint-consistent local optimization and 'oversized' covariance scaled sampling to focus samples on probable low-cost regions [33]. Their more recent work further enhances tracking reliability by explicitly enumerating the possible kinematic minima [37]. See also Sminchisescu & Jepson [32] for a mixture smoother that computes a Bayesian approximation to an entire trajectory distribution. This can be efficiently used in tandem with mixture filters like [4, 36, 37, 42].

Hyperdynamics uses stochastic dynamics with cost gradient based sampling as in [9, 22, 5], but 'boosts' the dynamics with a novel importance sampler constructed from the original probability surface using local gradient and curvature information. All of the annealing methods try to increase transition rates by sampling a modified distribution, but only the one given here specifically focuses samples on regions likely to contain transition states. There are also deterministic local-optimization-based methods designed to find transition states. See our companion paper [34, 38] for references. A complementary class of methods [44, 1, 27, 20, 13, 39] assumes that the basins of attraction of the dominant local minima are already known, and uses this to speed-up sampling in a way that satisfies detailed balance. If exact sampling (as opposed to mode finding) is required, such methods can be used in tandem with fast mixers like hyperdynamics or annealing, but we will not investigate this here.[4]

2 Sampling and Transition State Theory

2.1 Importance Sampling

Importance sampling works as follows. Suppose that we are interested in quantities depending on the distribution of some quantity $\mathbf{x}$, whose probability density is proportional to $f(\mathbf{x})$. Suppose that it is feasible to evaluate $f(\mathbf{x})$ pointwise, but that we are not able to sample directly from the distribution it defines, but only from an approximating distribution with density $f_b(\mathbf{x})$. We will base our estimates on a sample of N independent points, $\mathbf{x}_1, ..., \mathbf{x}_N$ drawn from $f_b(\mathbf{x})$.

[4] One should be particularly careful to preserve detailed balance when selecting a transition kernel (jump proposal mechanism) for MCMC sampling. Naively proposing jumps that do not take the relative volume factors of the source and target regions into account is simply incorrect, whereas computing even approximate (but necessarily higher-order) correction factors may lead to inefficiencies, if required at each simulation step. Methods like [1, 39] are designed to address such trade-offs. They are provably correct asymptotically, allow correction factors to be precomputed and take advantage of fast-mixing, local optimum search methods like [36, 38, 37].

The expectation value of some quantity $V(\mathbf{x})$ with respect to $f(\mathbf{x})$ can then be estimated as $\bar{V} = \sum_{i=1}^{N} w_i\, V(\mathbf{x}_i) / \sum_{i=1}^{N} w_i$, where the **importance weighting** of $\mathbf{x}_i$ is $w_i = f(\mathbf{x}_i)/f_b(\mathbf{x}_i)$ (this assumes that $f_b(\mathbf{x}) \neq 0$ whenever $f(\mathbf{x}) \neq 0$). It can be proved that the importance sampled estimator converges to the mean value of V as N increases, but it is difficult to assess how reliable the estimate $\bar{V}$ is in practice. Two issues affect this accuracy: the variability of the importance weights due to deviations between $f(\mathbf{x})$ and $f_b(\mathbf{x})$, and statistical fluctuations caused by the improbability of sampling infrequent events in the tails of the distribution, especially if these are critical for estimating $\bar{V}$.

2.2 Stochastic Dynamics

Various methods are available for speeding up sampling. Here we use a stochastic dynamics method on the potential surface defined by our cost function (the negative log-likelihood of the state probability given the observations, $f(\mathbf{x}) = -\log p(\mathbf{x}|\ldots)$). Canonical samples from $f(\mathbf{x})$ can be obtained by simulating the phase space dynamics defined by the Hamiltonian function:

$$H(\mathbf{x}, \mathbf{p}) = f(\mathbf{x}) + K(\mathbf{p}) \tag{1}$$

where $K(\mathbf{p}) = \mathbf{p}^\top \mathbf{p}/2$ is the kinetic energy, and $\mathbf{p}$ is the momentum variable. Averages of variables V over the canonical ensemble can be computed by using classical 2N-dimensional phase-space integrals:

$$\langle V \rangle \;=\; \frac{\iint V(\mathbf{x}, \mathbf{p})\, e^{-\alpha f(\mathbf{x})}\, e^{-\alpha K(\mathbf{p})}\, \mathbf{dx}\, \mathbf{dp}}{\iint e^{-\alpha f(\mathbf{x})}\, e^{-\alpha K(\mathbf{p})}\, \mathbf{dx}\, \mathbf{dp}} \tag{2}$$

where $\alpha = 1/T$ is the temperature constant. Dynamics (and hence sampling) is done by locally integrating the Hamilton equations:

$$\frac{\mathbf{dx}}{dt} \;=\; \mathbf{p} \qquad \text{and} \qquad \frac{\mathbf{dp}}{dt} \;=\; -\frac{df(\mathbf{x})}{\mathbf{dx}} \tag{3}$$

using a Langevin Monte Carlo type integration/rejection scheme that is guaranteed to perform sampling from the canonical distribution over phase-space:

$$\mathbf{x}_{i+1} \;=\; \mathbf{x}_i - \frac{\Delta t_{sd}^2}{2}\, \frac{df(\mathbf{x})}{\mathbf{dx}} + \Delta t_{sd} \mathbf{n}_i \tag{4}$$

Here, $\mathbf{n}_i$ is a vector of independently chosen Gaussian variables with zero mean and unit variance, and Δt_{sd} is the stochastic dynamics integration step. Compared to so called 'hybrid' methods, the Langevin method can be used with a larger step size and this is advantageous for our problem, where the step calculations are relatively expensive (see [22] and its references for a more complete discussion of the relative advantages of hybrid and Langevin Monte Carlo methods)[5]. For physical dynamics, t represents the physical time, while for statistical calculations it simply represents the number of steps performed since the start of the simulation. The simulation time is used in §3 below to estimate the acceleration of infrequent events produced by the proposed biased potential.

[5]Note that the momenta are only represented implicitly in the Langevin formulation. There is no need to update their values after each leapfrog step as they are immediately replaced by new ones drawn from the canonical distribution at the start of each iteration.

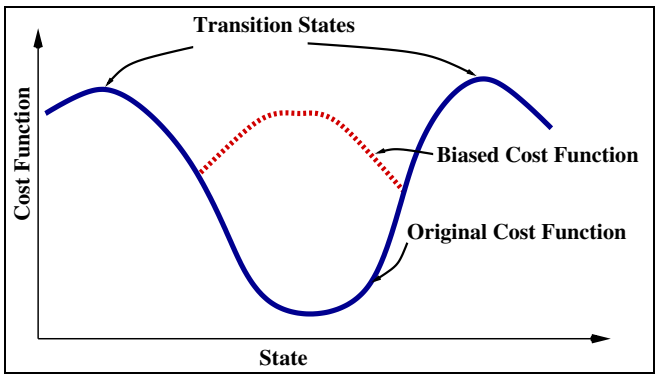


Figure 1: The original cost function and the bias added for hyperdynamics. The bias prevents extended trapping in the minimum by raising the cost there, while leaving it unchanged in the transition neighborhoods.

2.3 Transition State Theory

Continuing the statistical mechanics analogy begun in the previous section, the behavior of the physical system can be characterized by long periods of 'vibration' within one 'state' (energy basin), followed by infrequent transitions to other states via saddle points. In the 'transition state theory' (TST) approximation, the transition rates between states are computed using the sample flux through the *dividing surface* separating them. For a given state S, this is the $N-1$ dimensional surface separating the state S from its neighbors. The rate of escape from state S is:

$$k_{S\rightarrow}^{tst} = \langle\, |\nu_S|\, \delta_S(\mathbf{x}) \,\rangle_S \tag{5}$$

where $\delta_S(\mathbf{x})$ is a Dirac delta function positioned on the dividing surface of S and ν_S is the velocity normal to this surface. Crossings of the dividing surface correspond to true state change events, and we assume that the system loses all memory of this transition before the next event.

3 Accelerating Transition State Sampling

This section explains how transforming the sampling potential can accelerate transitions between minima, and hence allow fairer sampling of the minima that are present. It describes general properties that such transformations should obey and derives quantitative estimates for the expected acceleration factor. A specific functional form that approximates the required transformation properties is detailed in §4.

According to the transition state theory formalism presented in the previous section, the TST rate can be evaluated as follows, using (5) and (2):

$$k_{S\rightarrow}^{tst} = \frac{\iint\, |\nu_S|\, \delta_S(\mathbf{x})\, e^{-\alpha f(\mathbf{x})}\, e^{-\alpha K(\mathbf{p})}\, \mathbf{dx}\, \mathbf{dp}}{\iint\, e^{-\alpha f(\mathbf{x})}\, e^{-\alpha K(\mathbf{p})}\, \mathbf{dx}\, \mathbf{dp}} \tag{6}$$

Now consider adding a positive bias or boost cost $f_b(\mathbf{x})$ (with a corresponding 'biased' state S_b) to the original cost $f(\mathbf{x})$, with the further property that $f_b(\mathbf{x}) = 0$ whenever $\delta_S(\mathbf{x}) \neq 0$, *i.e.* the

potential is unchanged in the transition state regions. The TST rate becomes:

$$k^{tst}_{S\rightarrow} \;=\; \frac{\iint\; |\nu_S|\,\delta_S(\mathbf{x})\, e^{-\alpha[f(\mathbf{x})+f_b(x)]}\, e^{\alpha f_b(\mathbf{x})}\, e^{-\alpha K(\mathbf{p})}\, \mathbf{dx\,dp}}{\iint\; e^{-\alpha f(\mathbf{x})}\, e^{-\alpha K(\mathbf{p})}\, \mathbf{dx\,dp}} \tag{7}$$

$$= \frac{\left\langle\, |\nu_S|\,\delta_S(\mathbf{x})\, e^{\alpha f_b(\mathbf{x})}\,\right\rangle_{S_b}}{\left\langle\, e^{\alpha f_b(\mathbf{x})}\,\right\rangle_{S_b}} \;=\; \frac{\left\langle\, |\nu_S|\,\delta_S(\mathbf{x})\,\right\rangle_{S_b}}{\left\langle\, e^{\alpha f_b(\mathbf{x})}\,\right\rangle_{S_b}} \tag{8}$$

The boost term increases every escape rate from state S as the cost well is made shallower, but it leaves the *ratios* of escape rates from S, S_b to other states S_1, S_2 invariant:

$$\frac{k^{tst}_{S\rightarrow S_1}}{k^{tst}_{S\rightarrow S_2}} \;=\; \frac{k^{tst}_{S_b\rightarrow S_1}}{k^{tst}_{S_b\rightarrow S_2}} \tag{9}$$

This holds because all escape rates from S have the partition function of S as denominator, so replacing this with the partition function of S_b leaves their ratios unchanged. Concretely, suppose that during N_t steps of classical dynamics simulation on the biased cost surface, we encounter N_e escape attempts over the dividing surface. For the computation, let us also assume that the simulation is artificially confined to the basin of state S by reflecting boundaries. (This does not happen in real simulations: it is used here only to estimate the 'biased boost time'). The TST escape rate from state S can be estimated simply as the ratio of the number of escape attempts to the total trajectory length: $k^{tst}_S = N_e/(N_t \Delta t_{sd})$. Consequently, the mean escape time (inverse transition rate) from state S can be estimated from (7) as:

$$\tau^S_{esc} \;=\; \frac{1}{k^{tst}_{S\rightarrow}} \;=\; \frac{\left\langle\, e^{\alpha f_b(\mathbf{x})}\,\right\rangle_{S_b}}{\left\langle\, |\nu_S|\,\delta_S(\mathbf{x})\,\right\rangle_{S_b}} \;=\; \frac{\frac{1}{N_t}\sum_{i=1}^{N_t} e^{\alpha f_b(\mathbf{x}_i)}}{N_e/(N_t\,\Delta t_{sd})} \;=\; \frac{1}{N_e}\sum_{i=1}^{N_t} \Delta t_{sd}\, e^{\alpha f_b(\mathbf{x}_i)} \tag{10}$$

The effective simulation time boost achieved in step i thus becomes simply:

$$\Delta t_{b_i} \;=\; \Delta t_{sd} e^{\alpha f_b(\mathbf{x}_i)} \tag{11}$$

The dynamical evolution of the system from state to state is still correct, but it works in a distorted time scale that depends exponentially on the bias potential. As the system passes through regions with high f_b, its equivalent time Δt_b increases rapidly — owing to the large energy differential, the original dynamics would have tended to linger in (or return to) these regions much more often on average than the boosted dynamics suggests. Conversely, in zones with small f_b the equivalent time progress at the standard stochastic dynamics rate. Of course, in reality the simulation's integration time step and hence its sampling coarseness are the same as they were in the unboosted simulation. The boosting time (11) just gives an intuition for how much time an unaccelerated sampler would probably have wasted making 'uninteresting' samples near the cost minimum. But that is largely the point: the wastage factors are astronomical in practice — unboosted samplers can not escape from local minima.

4 The Biased Cost

The main requirements on the bias potential are that it should be zero on all dividing surfaces, that it should not introduce new sub-wells with escape times comparable to the main escape time from the original cost well, and that its definition should not require prior knowledge of the cost wells or

saddle points (if we knew these we could avoid trapping much more efficiently by including explicit well-jumping samples (*c.f.* [39, 37]). For sampling, the most 'important' regions of the cost surface are minima, where the Hessian matrix $\mathbf{H}$ has strictly positive eigenvalues, and transition states, where it has exactly one negative eigenvalue $e_1 < 0$. The gradient vector vanishes in both cases. The rigorous definition of the TST boundary is necessarily global[6], but locally near a transition state the boundary contains the state itself and adjacent points where the Hessian has a negative eigenvalue and vanishing gradient component along the corresponding eigenvector:

$$g_{p1} \;=\; \mathbf{V}_1^\top \mathbf{g} \;=\; 0 \qquad \text{and} \qquad e_1 < 0 \tag{12}$$

where $\mathbf{g}$ is the gradient vector and $\mathbf{V}_1$ is the first Hessian eigenvector. Voter [45, 46] therefore advocates the following bias cost for hyperdynamics:

$$f_b \;=\; \frac{h_b}{2}\left[1 + \frac{e_1}{\sqrt{e_1^2 + g_{p1}^2/d^2}}\right] \tag{13}$$

where h_b is a constant controlling the strength of the bias and d is a length scale (*e.g.* an estimate of the typical nearest-neighbor distance between minima, if this is available). In the neighborhood of any first-order saddle point, (12) gives a good approximation to the true TST dividing surface. Far away from the saddle, the approximation may not hold. For instance, the equation (12) may be satisfied in regions internal to a minimum that do not represent state boundaries or for some parts of the TST surface, the Hessian may have no negative eigenvalues and the gradient may not be zero along the lowest Hessian eigenvector (but some higher one). However, the zones of the dividing surface near low cost saddle points are the most likely regions for the inter-minimum transition, and given that the above bias potential is small in these neighborhoods, it provides a useful approximation to the true transition dynamics.

Increasing h_b increases the bias and hence the nominal boosting. In principle it is even possible to raise the minimum above the level of its surrounding transition states, but there is a risk that doing so would entirely block the sampling pathways through and around the minimum, thus causing the system to become trapped in a newly created well at one end of the old minimum. Hence, it is usually safer to select a more moderate boosting. Regardless of the choice of h_b, the bias potential (13) has the desirable property that it automatically decreases, and ultimately vanishes, in regions with the eigenvalue structure of transition neighborhoods (12).

4.1 Estimating the Gradient of the Bias Potential

Efficient sampling or optimization methods often require the gradient of the cost function. Direct differentiation of Voter's potential (13) for gradient-based dynamics requires third order derivatives of $f(\mathbf{x})$, but an inexpensive numerical estimation method based on first order derivatives was proposed in [46]. For completeness we summarize this here. The calculations are more complex than those needed for standard gradient based stochastic simulation, but in practice the exponential speed-up provided by the bias easily dominates the additional constant factors associated with this.

[6]The basin of state S can be defined as the set of configurations from which gradient descent minimization leads to the minimum S. This basin is surrounded by an $(n-1)$-D hypersurface, outside of which local descent leads to states other than S.

An eigenvalue can be computed by numerical approximation along its corresponding eigenvector direction $\mathbf{s}$:

$$e(\mathbf{s}) \;=\; [f(\mathbf{x}+\eta\mathbf{s}) + f(\mathbf{x}-\eta\mathbf{s}) - 2f(\mathbf{x})]/\eta^2 \tag{14}$$

The eigenvector direction can be estimated numerically using any gradient descent method, based on a random initialization $\mathbf{s}$ or on the one from the previous dynamics step, using:

$$\frac{de}{d\mathbf{s}} \;=\; [\mathbf{g}(\mathbf{x}+\eta\mathbf{s}) - \mathbf{g}(\mathbf{x}-\eta\mathbf{s})]/\eta \tag{15}$$

The lowest eigenvector obtained from the minimization (15) is then used to compute the corresponding eigenvalue via (14). The procedure can be repeated for higher eigenvalue-eigenvector pairs by maintaining orthogonality with previous directions. The derivative of the projected gradient g_{1p} can then be obtained by applying the minimization to the matrices $\mathbf{H} + \lambda\,\mathbf{g}\,\mathbf{g}^\top$ and $\mathbf{H} - \lambda\,\mathbf{g}\,\mathbf{g}^\top$. One thus minimizes:

$$\frac{de_i}{\mathbf{dx}} \;=\; \left\{[\mathbf{g}(\mathbf{x}+\eta\mathbf{s}) + \mathbf{g}(\mathbf{x}-\eta\mathbf{s}) - 2\mathbf{g}(\mathbf{x})]/\eta^2\right\}_{\mathbf{s}=\mathbf{s}_i} \tag{16}$$

where:

$$e_{\pm\lambda} \;=\; e(\mathbf{s}) \pm \lambda\Big[\frac{f(\mathbf{x}+\eta\mathbf{s}) - f(\mathbf{x}-\eta\mathbf{s})}{2\eta}\Big]^2 \tag{17}$$

A good approximation to g_{p1} can be obtained from [46]:

$$g_{p1} \;=\; \frac{1}{2\lambda}(e_{+\lambda} - e_{-\lambda}), \qquad \text{and} \qquad \frac{dg_{p1}}{\mathbf{dx}} \;=\; \frac{1}{2\lambda}\left(\frac{de_{+\lambda}}{\mathbf{dx}} - \frac{de_{-\lambda}}{\mathbf{dx}}\right) \tag{18}$$

5 Complementary Hyperdynamic Sampling Strategies

Hyperdynamic sampling provides rapid mixing, but not fair samples from the equilibrium distribution. Inter-mode transition probabilities are preserved (*c.f.* §3) but the cores of minima are strongly de-emphasized. To recover fair samples, some form of correction or post-processing is needed. There are several approaches, depending on the application. One possibility is importance sampling based reweighting (§2.1). Asymptotically, this will give correct results, but variability tends to be high as hyperdynamics deliberately makes as few samples as it can in the mode cores. Practically, improvements are possible: (*i*) For fair sampling, the 'bias' can provide initial seeds for a subsequent classical stochastic dynamics simulation operating on the original cost. By using such well-mixed seeds spread over a substantially large number of minima the structure of the density will be better represented. (*ii*) If precise localization of the modes is important *e.g.* in global optimization, hyperdynamics provides seeds for local descent on the original energy surface. A third option, reviewed next, combines the two ideas above and uses the known local minima within a fair sampling run.

Darting methods assume that the positions of local minima are known a priori, and use these to speed-up sampling from the equilibrium distribution [27, 1, 39]. They involve 'smart' moves that ensure both fair sampling and rapid jumping between minima. Without some prior knowledge of the structure of the distribution, long-range random jumps would stand a very low chance of being accepted, as they will most likely hit high-energy regions, especially in high-dimensions. To avoid

this, the initially proposed methods [1] work by placing spheres with equal radius at each local minimum. The sampler resembles a standard MCMC simulation with local steps, except that with fixed probability P a test is made to see whether the current configuration is inside one of the known spheres (*i.e*. neighborhoods of minima). If the sample is outside any sphere, it is simply recounted. If it is inside a sphere, its relative position with respect to the center is computed, a different minimum/sphere is selected with uniform probability, and a jump to the corresponding relative location with respect to that minimum is proposed. The move is accepted or rejected according to the usual Boltzmann criteria. One can prove that long-range jumps of this type obey detailed balance provided the spheres do not overlap [1]. This is true because the probability of entering a sphere from its outside is the same as the reverse move, namely $1 - P$ times the probability of a local move. For inter-minimum moves the probability is equal in both directions. See [1, 39] for details, and [39] for a generalization to different moves, shapes of local minima and overlapping regions.

6 Human Domain Modeling

This section briefly describes the humanoid visual tracking models used in our hyperdynamic boosting experiments. For more details see [33, 36].

Representation Our body models contain kinematic 'skeletons' of articulated joints controlled by angular joint parameters, covered by 'flesh' built from superquadric ellipsoids with additional global deformations [2]. A typical model has: about 30-35 joint parameters $\mathbf{x}_a$; 8 internal proportion parameters $\mathbf{x}_i$ encoding the positions of the hip, clavicle and skull tip joints; and 9 deformable shape parameters for each body part, gathered into a vector $\mathbf{x}_d$. The complete model is thus encoded as a single large parameter vector $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_d, \mathbf{x}_i)$. During tracking or static pose estimation we usually estimate only joint parameters.

The model is used as follows. Superquadric surfaces are discretized into meshes parameterized by angular coordinates in a 2D topological domain. Mesh nodes $\mathbf{u}_i$ are transformed into 3D points $\mathbf{p}_i(\mathbf{x})$, then into predicted image points $\mathbf{r}_i(\mathbf{x})$ using composite nonlinear transformations $\mathbf{r}_i(\mathbf{x}) = P(\mathbf{p}_i(\mathbf{x})) = P(A(\mathbf{x}_a, \mathbf{x}_i, D(\mathbf{x}_d, \mathbf{u}_i)))$, where D represents a sequence of parametric deformations that construct the corresponding part in its own reference frame, A represents a chain of rigid transformations that map it through the kinematic chain to its 3D position, and P represents perspective image projection. During model estimation, prediction-to-image matching cost metrics are evaluated between each predicted model feature $\mathbf{r}_i$ and nearby associated image features $\bar{\mathbf{r}}_i$, and the results are summed over all features to produce the image contribution to the overall parameter space cost function. The cost is thus a robust function of the prediction errors $\mathbf{\Delta r}_i(\mathbf{x}) = \bar{\mathbf{r}}_i - \mathbf{r}_i(\mathbf{x})$. The cost gradient $\mathbf{g}_i(\mathbf{x})$ and Hessian $\mathbf{H}_i(\mathbf{x})$ are also computed and assembled over all observations.

Estimation: We aim for a probabilistic interpretation and optimal estimates of the model parameters by maximizing the total probability according to Bayes rule:

$$p(\mathbf{x}|\bar{\mathbf{r}}) \;\propto\; p(\bar{\mathbf{r}}|\mathbf{x})\, p(\mathbf{x}) \;=\; \exp\left(-\sum_i e(\bar{\mathbf{r}}_i|\mathbf{x})\right) p(\mathbf{x}) \tag{19}$$

where $e(\bar{\mathbf{r}}_i|\mathbf{x})$ is the cost density associated with observation i, the integral is over all observations, and $p(\mathbf{x})$ is the prior on the model parameters. Discretizing the continuous problem, our MAP

approach minimizes the negative log-likelihood for the total posterior probability:

$$f(\mathbf{x}) \;=\; -\log p(\bar{\mathbf{r}}|\mathbf{x}) - \log p(\mathbf{x}) \;=\; f_l(\mathbf{x}) + f_p(\mathbf{x}) \tag{20}$$

Observation Likelihood: In the below experiments we actually only used a very simple Gaussian likelihood based on given model-to-image joint correspondences. The negative log-likelihood for the observations is just the sum of squared model joint reprojection errors. Our full tracking system uses this cost function only for initialization, but it still provides an interesting (and difficult to handle) degree of multimodality owing to the kinematic complexity of the human model and the large number of parameters that are unobservable in a singular monocular image. In practice we find that globalizing the search is at least as important for initialization as for tracking, and this cost function is significantly cheaper to evaluate than our full image based one, allowing more extensive sampling experiments.

Priors and Constraints: Both hard and soft priors are accommodated in our framework. They include anthropometric priors on model proportions, parameter stabilizers for hard to estimate but useful modelling parameters, terms for collision avoidance between body parts, and joint angle limits. During estimation, the values, gradients and Hessians of the priors are evaluated and added to the contributions from the observations.

7 Experiments

In this section we illustrate the hyperdynamics method on a toy problem involving a two-dimensional multi-modal cost surface, and on the problem of initial pose estimation for an articulated 3D human model based on given joint-to-image correspondences. In both cases we compare the method with standard stochastic dynamics on the original cost surface. The parameters of the two methods (temperature, integration step, number of simulation steps, *etc*.) are identical, except that hyperdynamics requires values for the two additional parameters h_b and d that control the properties of the bias potential (13).

7.1 The Müller Cost Surface

Müller's Potential (fig. 2, left) is a simple 2D analytic cost function with three local minima M_1, M_2, M_3, and two saddle points S_1, S_2, which is often used in the chemistry literature to illustrate transition state search methods. It has the form $V(x,y) = \sum_{i=1}^{4} A_i\, e^{a_i(x-x_i)^2 + b_i(x-x_i)(y-y_i) + c_i(y-y_i)^2}$ where $\mathbf{A} = (-200, -100, -170, 15)$, $\mathbf{a} = (-1, -1, -6.5, 0.7)$, $\mathbf{b} = (0, 0, 11, 0.6)$, $\mathbf{c} = (-10, -10, -6.5, 0.7)$, $\mathbf{x} = (1, 0, -0.5, -1)$, $\mathbf{y} = (0, 0.5, 1.5, 1)$. The inter-minimum distance is of order 1 length unit, and the transition states are around 100–150 energy units above the lowest minimum.

Fig. 2 (right) shows the result of standard stochastic dynamic sampling on the original cost surface. Despite 6000 simulation steps at a reasonable step size $\Delta t_{sd} = 0.01$, only the basin of the starting minimum is sampled extensively, and no successful escape has yet taken place. Fig. 3 shows two hyperdynamics runs with parameters set for moderate boosting. Note the reduced emphasis on sampling in the core of the minimum — in fact the minimum is replaced by a set of higher energy ones — and the fact that the runs escape the initial basin. In the right hand plot there is a clear focusing of samples in the region corresponding to the saddle point linking the two adjacent minima M_1 and M_2. Finally, fig. 4 shows results for more aggressive bias potentials that cause the basins of all three minima to be visited, with strong focusing of samples on the inter-minimum transition

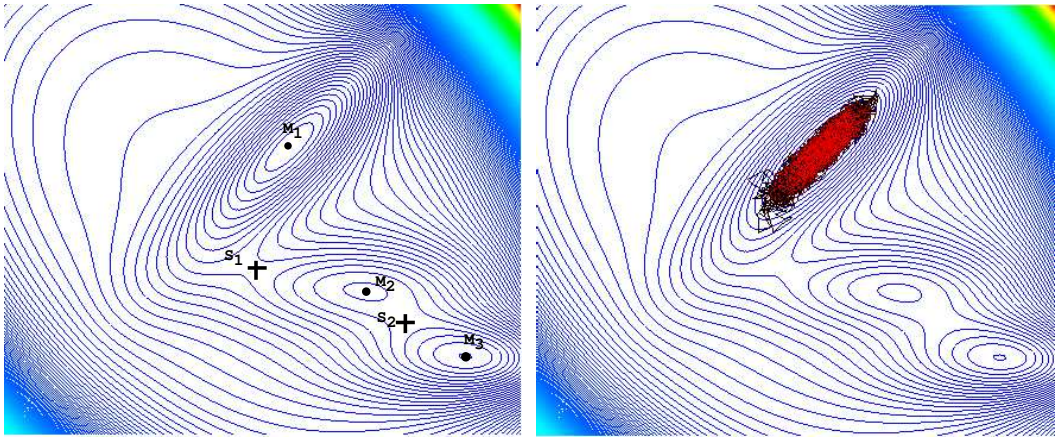


Figure 2: The Müller Potential (left) and a standard stochastic dynamics gradient sampling simulation (right) that gets trapped in the basin of the starting minimum.

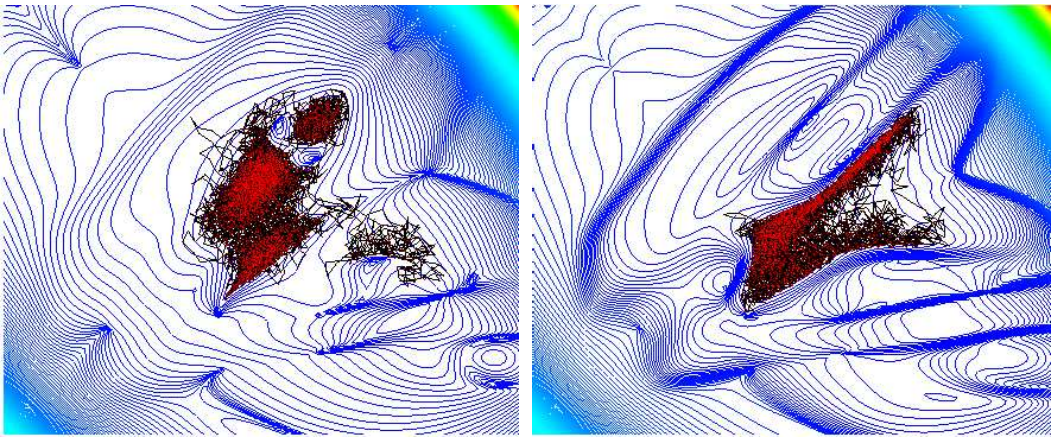


Figure 3: Hyperdynamic sampling with $h_b = 150, d = 0.1$ and $h_b = 200, d = 0.5$.

regions. The bias here turns the lowest positive curvature region of the initial minimum into a local maximum.

The plots also show that the Voter potential is somewhat 'untidy', with complicated local steps and ridges. Near the hypersurfaces where the first Hessian eigenvalue e_1 passes down through zero, the bias jumps from h_b to 0 with an abruptness that increases as the length scale d increases (sic) or the gradient projection g_{p1} decreases, owing to the $e_1/\sqrt{e_1^2 + g_{p1}^2/d^2}$ term in (13). A small d makes these $e_1 = 0$ transitions smoother, but increases the suddenness of ridges in the potential that occur on hypersurfaces where g_{1p} passes through zero.

Fig. 5 plots the simulation boosting time for two bias potentials. The left plot has a milder potential that simply encourages exploration of saddle points, while the right plot has a more aggressive one that is able to explore and jump between individual modes more rapidly. (Note the very large and very different sizes of the boosting time scales in these plots).

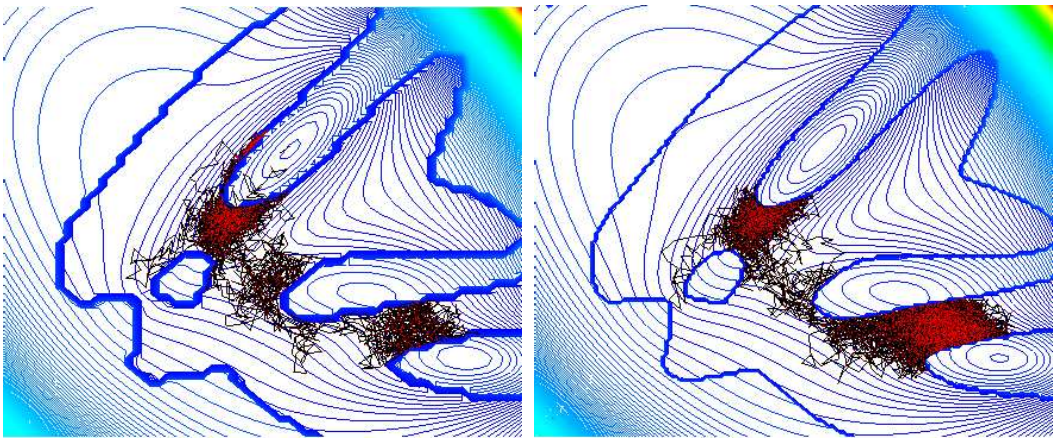


Figure 4: Hyperdynamic sampling with $h_b = 300, d = 10$ and $h_b = 400, d = 100$.

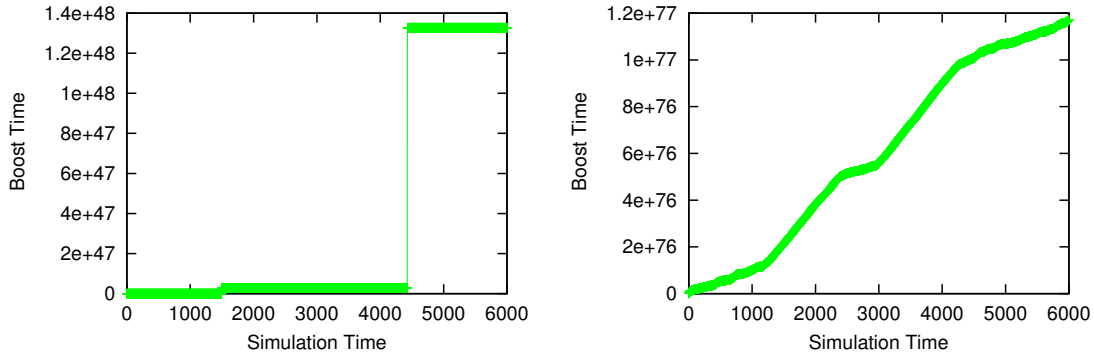


Figure 5: Effective boost times for mild (left) and more aggressive (right) bias potentials.

7.2 Monocular 3D Pose Estimation

Now we explore the potential of the hyperdynamics method for monocular 3D human pose estimation under model to image joint correspondences. This problem is well adapted to illustrating the algorithm, as its cost surface is highly multimodal. Of the 32 kinematic model d.o.f., about 10 are subject to 'reflective' kinematic ambiguities (forwards *vs*. backwards slant in depth), which potentially creates around $2^{10} = 1024$ local minima in the cost surface [16, 41, 37], although some of these are not physically feasible and are automatically pruned during the simulation (see below). Indeed, we find that it is very difficult to ensure initialization to the 'correct' pose with this kind of data.

The simulation enforces joint limit constraints using reflective boundary conditions, *i.e*. by reversing the sign of the particle's normal momentum when it hits a joint limit. We found that this gives an improved sampling acceptance rate compared to simply projecting the proposed configuration back into the constraint surface, as the latter leads to cascades of rejected moves until the momentum direction gradually swings around.

We ran the simulation for 8000 steps with $\Delta t_{sd} = 0.01$, both on the original cost surface (fig. 8) and on the boosted one (fig. 6). It is easy to see that the original sampler gets trapped in the starting mode, and wastes all of its samples exploring it repeatedly. Conversely, the boosted hyperdynamics

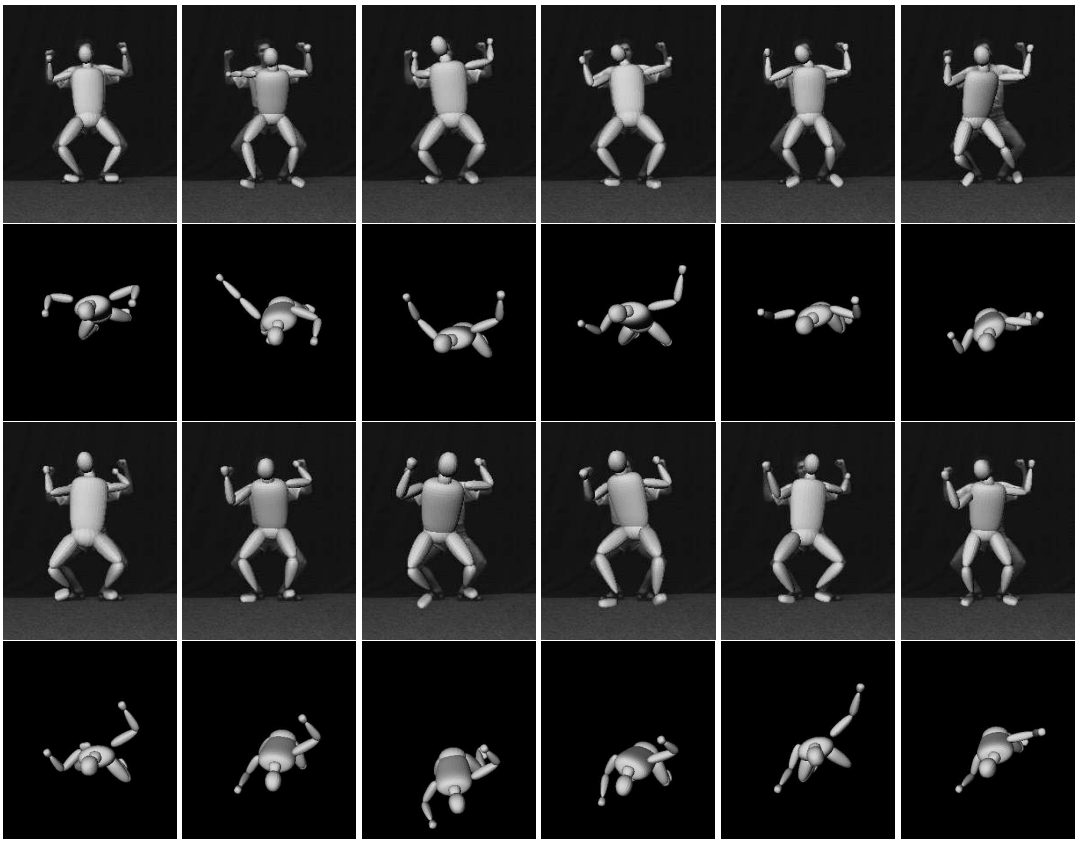


Figure 6: Human poses sampled using hyperdynamics on a cost surface based on given model-to-image joint correspondences, seen from the camera viewpoint and from above. Hyperdynamics finds a variety of different poses including well separated reflective ambiguities (which, as expected, all look similar from the camera viewpoint). In contrast, standard stochastic dynamics (on the same underlying cost surface with identical parameters) essentially remains trapped in the original starting mode even after 8000 simulation steps (fig. 8).

method escapes from the starting mode relatively quickly, and subsequently explores many of the minima resulting from the depth reflection ambiguities.

Fig. 7 plots the estimated boosting times for two different bias potentials, $h_b = 200, d = 2$, and $h_b = 400, d = 20$. The computed mean state variance of the original estimator was 4.10^{-6}, compared to 7.10^{-6} for the boosted one.

8 Conclusions and Open Research Directions

This paper has underlined the fact that for high dimensional multimodal cost functions, rather than focusing only on performing their target computation, importance samplers need to devote some of their samples to reducing trapping in local minima. With this in mind, we presented an MCMC sampler designed to accelerate the exploration of different minima, based on the 'hyperdynamics' method from computational chemistry. It uses local cost gradients and curvatures to construct a modified cost function that focuses samples towards regions with low gradient and at least one

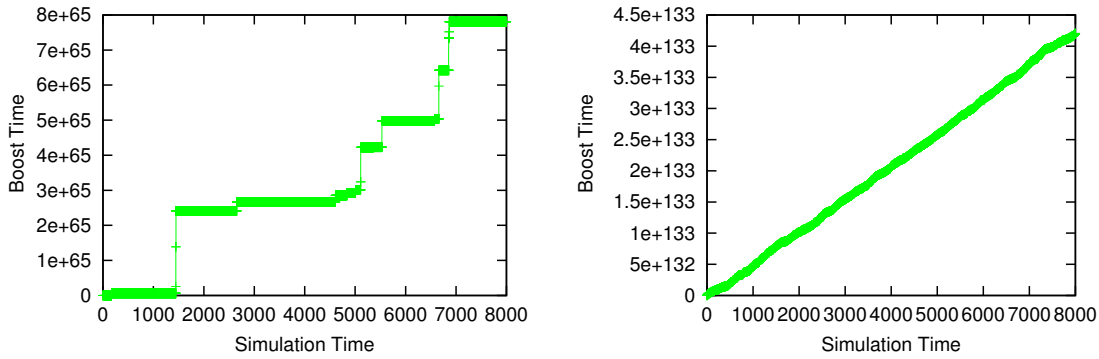


Figure 7: Boosting times for human pose experiments, with mild (left) and strong (right) bias.

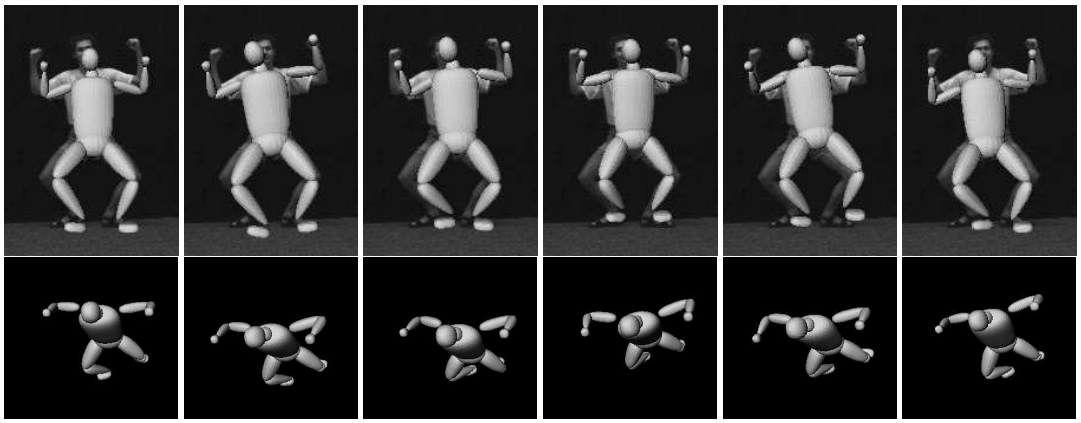


Figure 8: Stochastic dynamics on the original cost surface leads to "trapping" in the starting mode.

negative curvature, which are likely to contain the transition states (low cost saddle points with one negative curvature direction) of the original cost. Our experimental results demonstrate that the method significantly improves inter-minimum exploration behavior in the problem of monocular articulated 3D human pose estimation.

An interesting research direction would be the derivation and investigation of alternative, computationally more efficient biased sampling distributions.

References

[1] I. Andricioaiei, J. Straub, and A. Voter. Smart Darting MonteCarlo. *J. Chem. Phys.*, 114(16), 2001.

[2] A. Barr. Global and Local Deformations of Solid Primitives. *Computer Graphics*, 18, 1984.

[3] M. Black and A. Rangarajan. On the Unification of Line Processes, Outlier Rejection, and Robust Statistics with Applications in Early Vision. *International Journal of Computer Vision*, 19(1):57–92, July 1996.

[4] T. Cham and J. Rehg. A Multiple Hypothesis Approach to Figure Tracking. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 239–245, 1999.

[5] K. Choo and D. Fleet. People Tracking Using Hybrid Monte Carlo Filtering. In *IEEE International Conference on Computer Vision*, 2001.

[6] J. Deutscher, A. Blake, and I. Reid. Articulated Body Motion Capture by Annealed Particle Filtering. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2000.

[7] J. Deutscher, B. North, B. Bascle, and A. Blake. Tracking through Singularities and Discontinuities by Random Sampling. In *IEEE International Conference on Computer Vision*, pages 1144–1149, 1999.

[8] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.

[9] D. Forsyth, J. Haddon, and S. Ioffe. The Joy of Sampling. *International Journal of Computer Vision*, 41:109–134, 2001.

[10] D. Gavrila and L. Davis. 3-D Model Based Tracking of Humans in Action:A Multiview Approach. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 73–80, 1996.

[11] T. Heap and D. Hogg. Wormholes in Shape Space: Tracking Through Discontinuities Changes in Shape. In *IEEE International Conference on Computer Vision*, pages 334–349, 1998.

[12] N. Howe, M. Leventon, and W. Freeman. Bayesian Reconstruction of 3D Human Motion from Single-Camera Video. *Advances in Neural Information Processing Systems*, 1999.

[13] C. Jarzynski. Targeted Free Energy Perturbation. Technical Report LAUR-01-2157, Los Alamos National Laboratory, 2001.

[14] O. King and D. Forsyth. How does CONDENSATION Behave with a Finite Number of Samples? In *European Conference on Computer Vision*, pages 695–709, 2000.

[15] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by Simulated Annealing. *Science*, 1983.

[16] H. J. Lee and Z. Chen. Determination of 3D Human Body Postures from a Single View. *Computer Vision, Graphics and Image Processing*, 30:148–168, 1985.

[17] J. MacCormick and M. Isard. Partitioned Sampling, Articulated Objects, and Interface-Quality Hand Tracker. In *European Conference on Computer Vision*, volume 2, pages 3–19, 2000.

[18] E. Marinari and G. Parisi. Simulated Tampering: A New Monte Carlo Scheme. *Europhysics Letters*, 19(6), 1992.

[19] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *J. Chem. Phys.*, 21(6):1087–1092, 1953.

[20] M. Miller and W. Reinhardt. Efficient Free Energy Calculations by Variationally Optimized Metric Scaling. *J. Chem. Phys.*, 113(17), 2000.

[21] D. Morris and J. Rehg. Singularity Analysis for Articulated Object Tracking. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 289–296, 1998.

[22] R. Neal. Probabilistic Inference Using Markov Chain Monte Carlo. Technical Report CRG-TR-93-1, University of Toronto, 1993.

[23] R Neal. Annealed Importance Sampling. Technical Report 9805, Department of Statistics, University of Toronto, 1998.

[24] R. Neal. Annealed Importance Sampling. *Statistics and Computing*, 11:125–139, 2001.

[25] R. Plankers and P. Fua. Articulated Soft Objects for Video-Based Body Modeling. In *IEEE International Conference on Computer Vision*, pages 394–401, 2001.

[26] R. Rosales and S. Sclaroff. Inferring Body Pose without Tracking Body Parts. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 721–727, 2000.

[27] H. Senderowitz, F. Guarnieri, and W. Still. A Smart Monte Carlo Technique for Free Energy Simulations of Multiconformal Molecules. Direct Calculation of the Conformational Population of Organic Molecules. *J. Am. Chem. Society*, 117, 1995.

[28] E. M. Sevick, A. T. Bell, and D. N. Theodorou. A Chain of States Method for Investigating Infrequent Event Processes Occuring in Multistate, Multidimensional Systems. *J. Chem. Phys.*, 98(4), 1993.

[29] H. Sidenbladh, M. Black, and D. Fleet. Stochastic Tracking of 3D Human Figures Using 2D Image Motion. In *European Conference on Computer Vision*, 2000.

[30] C. Sminchisescu. Consistency and Coupling in Human Model Likelihoods. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 27–32, Washington D.C., 2002.

[31] C. Sminchisescu and A. Jepson. Generative Modeling for Continuous Non-Linearly Embedded Visual Inference. In *International Conference on Machine Learning*, pages 759–766, Banff, 2004.

[32] C. Sminchisescu and A. Jepson. Variational Mixture Smoothing for Non-Linear Dynamical Systems. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 608–615, Washington D.C., 2004.

[33] C. Sminchisescu and B. Triggs. Covariance-Scaled Sampling for Monocular 3D Body Tracking. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 447–454, Hawaii, 2001.

[34] C. Sminchisescu and B. Triggs. Building Roadmaps of Local Minima of Visual Models. In *European Conference on Computer Vision*, volume 1, pages 566–582, Copenhagen, 2002.

[35] C. Sminchisescu and B. Triggs. Hyperdynamics Importance Sampling. In *European Conference on Computer Vision*, volume 1, pages 769–783, Copenhagen, 2002.

[36] C. Sminchisescu and B. Triggs. Estimating Articulated Human Motion with Covariance Scaled Sampling. *International Journal of Robotics Research*, 22(6):371–393, 2003.

[37] C. Sminchisescu and B. Triggs. Kinematic Jump Processes for Monocular 3D Human Tracking. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 69–76, Madison, 2003.

[38] C. Sminchisescu and B. Triggs. Mapping Minima and Transitions in Visual Models. *International Journal of Computer Vision*, 2005.

[39] C. Sminchisescu, M. Welling, and G. Hinton. A Mode-Hopping MCMC Sampler. Technical Report CSRG-478, University of Toronto, submitted to *Machine Learning Journal*, September 2003.

[40] M. R. Sorensen and A. F. Voter. Temperature-Accelerated Dynamics for Simulation of Infrequent Events. *J. Chem. Phys.*, 112(21):9599–9606, 2000.

[41] C. J. Taylor. Reconstruction of Articulated Objects from Point Correspondences in a Single Uncalibrated Image. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 677–684, 2000.

[42] J. Vermaak, A. Doucet, and P. Perez. Maintaining Multimodality through Mixture Tracking. In *IEEE International Conference on Computer Vision*, 2003.

[43] G. H. Vineyard. Frequency factors and Isotope Effects in Solid State Rate Processes. *J. Phys. Chem. Solids*, 3:121–127, 1957.

[44] A. Voter. A Monte Carlo Method for Determining Free-Energy Differences and Transition State Theory Rate Constants. *J. Chem. Phys.*, 82(4), 1985.

[45] A. F. Voter. A Method for Accelerating the Molecular Dynamics Simulation of Infrequent Events. *J. Chem. Phys.*, 106(11):4665–4677, 1997.

[46] A. F. Voter. Hyperdynamics: Accelerated Molecular Dynamics of Infrequent Events. *Physical Review Letters*, 78(20):3908–3911, 1997.

Kinematic Jump Processes For Monocular 3D Human Tracking

Cristian Sminchisescu and **Bill Triggs**
GRAVIR-CNRS-INRIA, 655 avenue de l'Europe, 38330 Montbonnot, France
{*Cristian.Sminchisescu,Bill.Triggs*}*@inrialpes.fr, www.inrialpes.fr/movi/people/*{*Sminchisescu,Triggs*}

Abstract

A major difficulty for 3D human body tracking from monocular image sequences is the near non-observability of kinematic degrees of freedom that generate motion in depth. For known link (body segment) lengths, the strict non-observabilities reduce to twofold 'forwards/backwards flipping' ambiguities for each link. These imply $2^{\#\,links}$ formal inverse kinematics solutions for the full model, and hence linked groups of $\mathcal{O}(2^{\#\,links})$ local minima in the model-image matching cost function. Choosing the wrong minimum leads to rapid mistracking, so for reliable tracking, rapid methods of investigating alternative minima within a group are needed. Previous approaches to this have used generic search methods that do not exploit the specific problem structure. Here, we complement these by using simple kinematic reasoning to enumerate the tree of possible forwards/backwards flips, thus greatly speeding the search within each linked group of minima. Our methods can be used either deterministically, or within stochastic 'jump-diffusion' style search processes. We give experimental results on some challenging monocular human tracking sequences, showing how the new kinematic-flipping based sampling method improves and complements existing ones.

Keywords: *Monocular 3D human body tracking, kinematic ambiguity, Covariance Scaled Sampling, inverse kinematics, particle filtering, constrained optimization, high-dimensional search.*

1 Introduction

A major difficulty for 3D human body tracking from monocular image sequences is the quasi-unobservability of kinematic degrees of freedom that generate motion in depth. For unknown limb (link) lengths this leads to continuous nonrigid 'affine folding' ambiguities, but once lengths are known these reduce to twofold 'forwards/backwards flipping' ambiguities for each link. The full model thus has $2^{\#\,\text{links}}$ formal inverse kinematics solutions. Even with strong joint limits and no image correspondence ambiguities, the model-image matching cost function typically still has $\mathcal{O}(2^{\#\,\text{links}})$ local minima, so optimizing it is a difficult global search problem. But also a necessary one, as following the wrong local minimum rapidly leads to mistracking.

Several generic global search methods have already been applied to this problem [4, 11, 14, 16], but they tend to be somewhat inefficient as they make little use of the specific problem structure. Here, we develop a new method that speeds the search for local minima by using simple kinematic principles to construct 'interpretation trees' generating the possible 3D body configurations associated with a given set of projected joint centres (§3). We give simple closed-form inverse kinematics solutions for constructing these trees for human limbs, and show how the method can be used to produce an efficient deterministic 'kinematic jump' sampler for the different configurations. We use this sampler to construct a novel mixture density propagation based tracking algorithm (§4) that combines local covariance based diffusion, adaptive kinematic chain selection based on local uncertainties, quasi-global kinematic jumps and local continuous constrained optimization. We present quantitative results showing the effectiveness of the new samplers compared to existing methods (§5.1), and conclude with some challenging monocular experiments showing the final tracker's ability to follow rapid, complex human motions in clutter.

1.1 Related Research

There is a large literature on human motion tracking but relatively little work on developing search methods that exploit both the local and global structure present in the 3D monocular articulated problem. Sidenbladh *et al* use particle filtering with importance sampling based on either a learned walking model or a database of motion snippets, to focus search in the neighborhood of known trajectory pathways [11, 12]. Deutscher *et al* propose an annealing framework in a multi-camera setting [3]. During annealing, the search for parameters is driven by noise proportional with their individual variances [4]. Considered as an improved (implicit) search space decomposition mechanism, an early method of this type was proposed by Gavrila & Davis [6] to efficiently sample partial kinematic chains. Adaptively identifying and sampling parameters with high variance is useful, but kinematic parameters usually have quite strong interactions that make simple axis-aligned sampling questionable. It is important to realize that the principal axes of the covariance change drastically depending on the viewing direction, and that even if these are computed and used for sampling (as in [14]), they are only local measures that capture little information about the global minimum structure.

Published in 2003 IEEE Int. Conf. on Computer Vision& Pattern Recognition.

Sminchisescu & Triggs [14] argue that an effective random sampler must combine all three of cost-surface-aware covariance scaling, a sampling distribution with widened tails for deeper search, and local optimization (because deep samples usually have very high costs, and hence will not be resampled even if they lead to other minima). More recently, they have also constructed deterministic optimization methods [15] and cost-function-modifying MCMC samplers [16], for finding 'transition states' (saddle points) leading to nearby minima.

Skeletal reconstruction methods recover an interpretation tree of possible 3D joint positions, based on user-specified image joint positions [8, 17]. Lee & Chen [8] attempt to prune their perspective interpretation tree using physical reasoning, while Taylor [17] relies on additional user input to specify plausible relative joint-centre depths for his affine one. Although these methods do incorporate the forward-backward flipping ambiguity, they can not reconstruct skeletal joint angles, and this makes them inappropriate for tracking applications.

Our approach can be seen as a marriage of locally optimized covariance based random sampling with a domain-specific deterministic sampler based on skeletal reconstruction using inverse kinematics. The local covariance information obtained during optimization also provides a useful heuristic for which kinematic parameters to sample.

2 Modeling and Estimation

Representation The 3D body model used in our human tracking experiments consists of a kinematic 'skeleton' of articulated joints controlled by angular joint parameters, covered by a 'flesh' built from superquadric ellipsoids with additional global deformations [1]. A typical model has 30–35 joint parameters; 8 internal proportions encoding the positions of the hip, clavicle and skull tip joints; and 9 deformable shape parameters for each body part. The complete model is encoded in a single large parameter vector $\mathbf{x}$. During tracking and static pose estimation we usually estimate only joint parameters, but during initialization some length ratios are also estimated. In use, the superquadric surfaces are discretized into 2D meshes and the mesh nodes are mapped to 3D points using the kinematic body chain then projected to predicted image points $\mathbf{r}_i(\mathbf{x})$ using perspective image projection.

Observation Likelihood: During tracking robust model-to-image matching cost metrics are evaluated for each predicted image feature $\mathbf{r}_i$, and the results are summed over all observations to produce the image contribution to the parameter space cost function. Cost gradient and Hessian contributions $\mathbf{g}_i, \mathbf{H}_i$ are also computed and assembled. We use a robust combination of extracted-feature-based metrics and intensity-based matching ones (registering the model reprojected texture at previous tracking step with the current image) and robustified normalized edge energy. The feature-based terms associate the predictions $\mathbf{r}_i$ with nearby image features $\bar{\mathbf{r}}_i$, the cost being a robust function of the prediction errors $\mathbf{\Delta r}_i(\mathbf{x}) = \bar{\mathbf{r}}_i - \mathbf{r}_i(\mathbf{x})$. We also give results for a simpler likelihood designed for model initialization, based on squared distances between reprojected model joints and their specified image positions.

Priors and Constraints: Our model [14] incorporates both hard constraints (for joint angle limits) and soft priors (penalties for anthropometric model proportions, collision avoidance between body parts, and stabilization of useful but hard-to-estimate model parameters such as internal d.o.f. of the clavicle complex). The priors provide additional cost, gradient and Hessian contributions for the optimization.

Estimation: We apply Bayes rule and maximize the total posterior probability to give locally MAP parameter estimates:

$$\log p(\mathbf{x}|\bar{\mathbf{r}}) \propto \log p(\mathbf{x}) + \log p(\bar{\mathbf{r}}|\mathbf{x}) = \log p(\mathbf{x}) - \int e(\bar{\mathbf{r}}_i|\mathbf{x})\, di \tag{1}$$

Here, $p(\mathbf{x})$ is the prior on the model parameters, $e(\bar{\mathbf{r}}_i|\mathbf{x})$ is the cost density associated with observation i, and the integral is over all observations (assumed independent). Equation (1) gives the model likelihood in a single image, under the model priors but without initial state or temporal priors. During tracking, the temporal prior at time t is determined by the previous posterior $p(\mathbf{x}_{t-1}|\mathbf{R}_{t-1})$ and the system dynamics $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, where we have collected the observations at time t into vector $\mathbf{r}_t$ and defined $\mathbf{R}_t = \{\mathbf{r}_1, \ldots, \mathbf{r}_t\}$. The posterior at t becomes

$$p(\mathbf{x}_t|\mathbf{R}_t) \propto p(\bar{\mathbf{r}}_t|\mathbf{x}_t)\, p(\mathbf{x}_t) \int_{\mathbf{x}_{t-1}} p(\mathbf{x}_t|\mathbf{x}_{t-1})\, p(\mathbf{x}_{t-1}|\mathbf{R}_{t-1})$$

Together $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and $p(\mathbf{x}_{t-1}|\mathbf{R}_{t-1})$ form the time t prior $p(\mathbf{x}_t|\mathbf{R}_{t-1})$ for the image correspondence search (1).

3 Kinematic Jump Processes

Each configuration of the skeletal kinematic tree has an associated *interpretation tree* — the tree of all fully- or partially-assigned 3D skeletal configurations that can be obtained from the given one by forwards/backwards flips. The tree contains only, and generically all, configurations that are image-consistent in the sense that their joint centres have the same image projections as the given one. (Some of these may still be inconsistent with other constraints: joint limits, body self-intersection, occlusion...). The interpretation tree is constructed by traversing the kinematic tree from the root to the leaves. For each link, we construct the 3D sphere centred on the currently hypothesized position of the link's root, with radius equal to link length. This sphere is pierced by the camera ray of sight through the observed image position of the link's endpoint to give (in general) two possible 3D positions of the endpoint that are consistent with the image observation

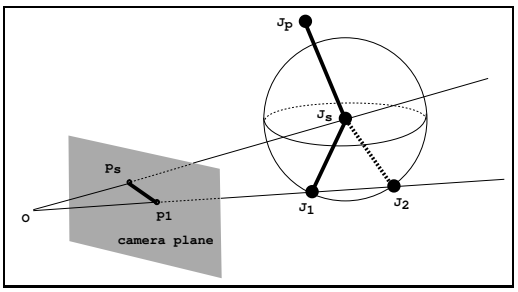


Figure 1: Forwards/backwards ambiguity for a kinematic link under monocular perspective projection. Given a standard joint configuration $...J_pJ_sJ_1$, one can build an alternative 'flipped' configuration $...J_pJ_sJ_2$ with the same joint-centre image projections. J_2 is found by intersecting the sphere centered at J_s with radius $|J_sJ_1|$ with the camera line of sight through the projection of J_1, OJ_1.

and the hypothesized parent position (see fig. 1). Joint angles are then recovered for each position using simple inverse kinematics (see below). If the ray misses the sphere, the parent hypothesis was inconsistent with the image data and the branch can be pruned.

More precisely, the above tree structure applies to non-branching kinematic chains such as limbs. When there is kinematic branching — *e.g.* for the four limbs attached to the trunk — each branch can be sampled independently, so the set of possible interpretations has a natural factored 'product of trees' structure. In such cases we build independent trees for each limb and take their product, *e.g.* each full-body configuration contains independently-sampled configurations for each of the four limbs.

Compared to current generic configuration space sampling methods, forwards/backwards flipping generates high-quality hypotheses very rapidly, and also provides unusually thorough coverage, at least within each kinematically-induced equivalence class of minima. Its quality stems from the fact that the hypotheses generated all have approximately-correct image projections (in particular, correct joint-centre projections). Its rapidity stems from the existence of simple closed form solutions for the inverse kinematics in this particular case (*i.e.* flexible kinematics constrained by observed joint-centre projections), and the fact that the accurate hypotheses generated do not need further 'polishing' by expensive non-linear optimization.

One could also generate 'flips' using classical closed-form or iterative techniques for solving the full inverse kinematics of the articulated skeleton, *e.g.* [10, 18]. However these methods are not well-adapted to this application in the sense that they solve a much more complicated problem (full redundant kinematics from a given end-effector pose) while ignoring much of the available image information (constrained projections of intermediate joint centres).

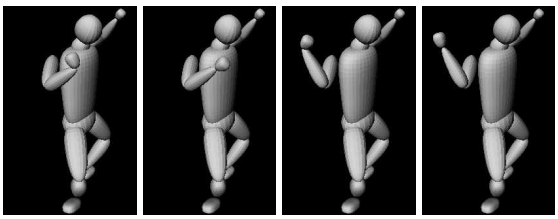


Figure 2: The 'flipping' ambiguities of the forearm and hand under monocular perspective. (The left-most configuration violates a wrist joint-angle limit and will be pruned away).

3.1 Direct Inverse Kinematics

As described above, flipping applies only to kinematic chains with fully spherical joints. Single d.o.f. joints such as hinges are usually too rigid to have a flipping ambiguity, as two d.o.f. are needed to move the link end to an arbitrary new position on the sphere. However, for human kinematics, flipping ambiguities apply even to hinge joints such as the elbow: although physically a hinge, the elbow effectively has spherical mobility once axial rotations of the upper arm about the shoulder are included. Here we give the inverse kinematics of this three link case as an example. We work in a reference coordinate system and know the 3D positions $\mathbf{P}_i$ of joints J_i, $i = 1..4$, as well as the rotational displacement $\mathbf{R}$ of J_1 with respect to the reference frame. The kinematic chain is represented in terms of Euler angles and pure translations along negative z axes. We use $\widehat{\mathbf{R}}$ to denote the z column of the rotation matrix $\mathbf{R}$. Suppose $\mathbf{p}_i = \frac{\mathbf{P}_i - \mathbf{P}_{i+1}}{\|\mathbf{P}_i - \mathbf{P}_{i+1}\|}$, with $i = 1..3$ unit vectors specifying the (known) z axes at each individual joint, after applying the rotation in that joint. There are 3 d.o.f. in J_1, 1 d.o.f. in J_2 and 2 d.o.f. in J_3 – these are represented by rotation matrices $\mathbf{R}^{1,2,3}_{x,y,z}$ as in fig. 3. To solve for rotations, we descend the kinematic chain and factor rotation angles $(x, y, z)_{1,2,3}$ by applying the constraints derived from the known positions of $\mathbf{P}_i$. The key observation is that, at any joint J_i, given the known previous rotational displacement, we have to factor out a rotation that aligns the z-axis with $\mathbf{p}_i$. For instance, at J_1, $\widehat{\mathbf{R}^1_x\mathbf{R}^1_y\mathbf{R}^1_z} = \mathbf{R}^T\mathbf{p}_i$ and we extract x_1, y_1 from:

$$\begin{pmatrix} -\sin(y_1) \\ \sin(x_1)\cos(y_1) \\ \cos(x_1)\cos(y_1) \end{pmatrix} = \mathbf{R}^T\mathbf{p}_1$$

In general this gives 4 solutions for x_1, y_1, but usually 2 do not satisfy all 3 equalities and are removed. z_1 is then recovered together with x_2 by solving $\widehat{\mathbf{R}^1_z\mathbf{R}^2_x} = (\mathbf{R}\mathbf{R}^1_x\mathbf{R}^1_y)^T\mathbf{p}_2$ for the next joint J_2:

$$\begin{pmatrix} \sin(z_1)\sin(x_2) \\ \cos(z_1)\sin(x_2) \\ \cos(x_2) \end{pmatrix} = (\mathbf{R}\mathbf{R}^1_x\mathbf{R}^1_y)^T\mathbf{p}_2$$

Again there are 4 possible solutions but 2 can be pruned. Finally, x_3, y_3 are obtained in the same way as x_1, y_1, given the

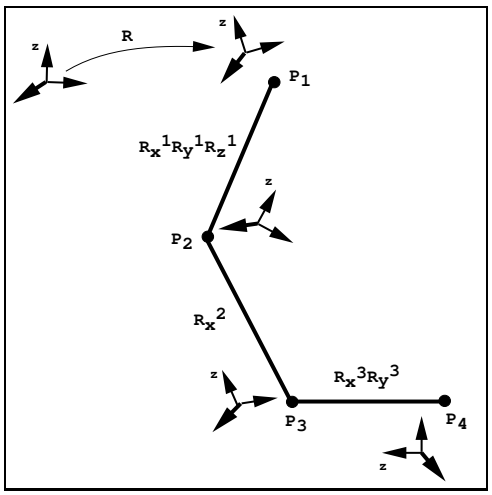


Figure 3: A three-joint link modeling anthropometric limbs. It has one spherical joint J_1, one hinge joint J_2 and a 2 d.o.f. end effector J_3. The representation is built in terms of Euler angles (with associated rotation matrices $\mathbf{R}_{x,y,z}^{1,2,3}$ with angles as sub-scripts and the joint rotation centers as superscripts) and pure translations to the next joint along the negative z axis. The inverse kinematics solution factors rotation angles using knowledge of successive z axes (computed from $\mathbf{P}_i - \mathbf{P}_{i+1}$) for limbs.

known x_1, y_1, z_1, x_2 values. As as special case, note that $\mathbf{R}_z^1$ remains unconstrained when $\mathbf{P}_1$, $\mathbf{P}_2$ and $\mathbf{P}_3$ are collinear. In this case, z_1 is either fixed to some default value or (for tracking) sampled within its range of variation.

3.2 Iterative Inverse Kinematics

In some situations, the simple closed form inverse kinematics given above does not suffice. This might happen for more general kinematic structures — for example the looped kinematic chains formed when the hands are joined or placed on the hips — or when the exact inverse kinematics either fails (a camera ray does not intersect its sphere) or is expected to be inaccurate for some reason (a joint limit or body non-self-intersection constraint is violated). In such cases, we can fall back on a more general approach that directly minimizes the sum of squared differences between the current and desired joint configurations, using nonlinear optimization in joint space. Our minimizer uses analytical gradients and Hessians in a second-order damped Newton trust-region framework, with both hard joint-angle limits and soft non-self-intersection and image correspondence constraints [14]. In practice, this method locates new flipped local minima fairly successfully, but is significantly more expensive than kinematics-based flipping as $\mathcal{O}(1)$ full local optimization runs are needed for each new minimum found. However this is still significantly more efficient than the random samplers we have tested — see §5.

4 The Algorithm

In normal use, we embed our kinematic jump sampler within a cost-sensitive mixture density propagation framework [14]. The jump sampler ensures rapid, consistent diffusion of samples across the kinematic minima associated with any given set of image joint positions, while the random sampler provides robustness against incorrect image correspondences. Here, we use a Covariance Scaled Sampling [14] tracker. This probabilistic method represents the posterior distribution of hypotheses in joint space as a mixture of long-tailed Gaussian-like distributions $m_i \in \mathcal{M}$, whose weights, centres and scale matrices ('covariances') $m_i = (c_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ are obtained as follows. Random samples are generated, and each is optimized (by nonlinear local optimization, respecting any joint constraints, *etc*.) to maximize the local posterior likelihood encoded by an image- and prior-knowledge based cost function. The optimized likelihood value and position give the weight and centre of a new component, and the inverse Hessian of the log-likelihood gives a scale matrix that is well adapted to the contours of the cost function, even for very ill-conditioned problems like monocular human tracking. However, when sampling, particles are deliberately scattered more widely than a Gaussian of this scale matrix (covariance) would predict, in order to probe more deeply for alternative minima.

Fig. 4 gives the general form of the algorithm, and fig. 5 describes the novel **KinematicDiffusionJumpSampling** routine that lies at its core. On entry, the user specifies a set $\mathcal{C}$ of kinematic sub-chains that may be sampled (this can be quite large, as the routine adaptively decides which to sample). At each time step, covariance scaled samples are generated from the prior. For each such sample an interpretation tree is created on-line by the **BuildInterpretationTree** routine, with kinematic solutions obtained using **InverseKinematics**. The chain to be sampled is chosen adaptively using a voting process based on the local covariance structure of that region of the parameter space, **SelectSamplingChain** in fig. 5. Local covariance scaled resampling is performed before the jump because we do not (yet) have the covariance information needed to perform it afterwards. Each element of the sampleable sub-chain set $\mathcal{C}$ is simply a list of parameter names to sample. For instance, for a sub-chain rooted at the left shoulder, this might include the rotational parameters $(x_s, y_s, z_s, x_e, x_h, y_h)$ where the (s, e, h) stand for (shoulder, elbow, hand) and x, y, z for the rotation axes.

The proposed sampling strategy provides a balance between local and global search effort since samples are generated around the prior modes, as well as around new peaks that are potentially emerging and have not yet been explored. Re-weighting based on closest prior modes as in fig. 4, step 5, ensures the tracker is not distracted by remote multi-modality when tracking the correct minima.

Kinematic Jump + CSS Diffusion Based Tracker

Input: The set $\mathcal{C}$ of permissible kinematic chain partitions to use for sampling, and the previous posterior $p(\mathbf{x}_{t-1}|\mathbf{R}_{t-1}) = \sum_{i=1}^{K} \pi_i^{t-1} \mathcal{N}(\mu_i^{t-1}, \Sigma_i^{t-1})$.

---

1. Build the covariance scaled proposal density $p^*(t-1) = \sum_{i=1}^{K} \pi_i^{t-1} \mathcal{N}(\mu_i^{t-1}, s\Sigma_i^{t-1})$. ($s \sim$ 4-6).

---

2. Generate a set of samples S using **KinematicDiffusionJumpSampling** on $p^*(t-1)$ and $\mathcal{C}$.

---

3. Optimize each sample $s_j \in S$ w.r.t. the time t observation likelihood (1), using local constrained optimization to get MAP estimates μ_j^t with covariances $\Sigma_j^t = \mathbf{H}(\mu_j^t)^{-1}$.

---

4. Construct the unpruned posterior $p_t^u(\mathbf{x}_t|\mathbf{R}_t) = \sum_{j=1}^{N} \pi_j^t \mathcal{N}(\mu_j^t, \Sigma_j^t)$, where $\pi_j^t = \frac{p(\mu_j^t|\bar{\mathbf{r}}_t)}{\sum_{j=1}^{N} p(\mu_j^t|\bar{\mathbf{r}}_t)}$, and prune it to keep the K components with highest probability: $p_t^p(\mathbf{x}_t|\mathbf{R}_t) = \sum_{k=1}^{K} \pi_k^t \mathcal{N}(\mu_k^t, \Sigma_k^t)$, with $\pi_k^t = \frac{p(\mu_k^t|\bar{\mathbf{r}}_t)}{\sum_{j=1}^{K} p(\mu_j^t|\bar{\mathbf{r}}_t)}$.

---

5. For each mixture component $j = 1..K$ in p_t^p, find the closest prior component i in $p(\mathbf{x}_{t-1}|\mathbf{R}_{t-1})$ in Bhattacharyya distance $\mathcal{B}_{ij}(\mu_i^{t-1}, \Sigma_i^{t-1}, \mu_j^t, \Sigma_j^t)$. Scale $\pi_j^t = \pi_j^t * \pi_i^{t-1}$ and discard component i of $p(\mathbf{x}_{t-1}|\mathbf{R}_{t-1})$.

---

6. Compute the final posterior mixture $p(\mathbf{x}_t|\mathbf{R}_t) = \sum_{k=1}^{K} \pi_k^t \mathcal{N}(\mu_k^t, \Sigma_k^t)$, with $\pi_k^t = \frac{\pi_k^t}{\sum_{j=1}^{K} \pi_j^t}$.

Figure 4: The steps of our mixture density propagation algorithm.

$S =$ **KinematicDiffusionJumpSampling**$(\{p^*\}, \mathcal{C})$
Generates a set of samples S based on Covariance Scaled Sampling diffusion and kinematic jump processes.

1. Use **SelectSamplingChain**$(\Sigma_i, \mathcal{C})$ to select a kinematic chain $C_i \in \mathcal{C}$ to sample for each mixture component p_i^*.
2. Generate N random samples as follows:
2.1. Choose a mixture component p_i^* with probability π_i.
2.2. CSS sample from p_i^* to obtain $\mathbf{s}_j$.
2.3. T_j = **BuildInterpretationTree**$(\mathbf{s}_j, C_i)$.
2.4. For each path (list of 3D joint positions) P in T_j, use **InverseKinematics**(P) to find joint angles c_P, and add c_P to the list of samples, $S = S \cup c_P$.

SelectSamplingChain$(\Sigma, \mathcal{C})$
Heuristic to select a chain $C \in \mathcal{C}$ to sample for a component with covariance Σ. $C = \cup_{i=1}^{M} C_i$. The function Idx(C_i) will provide the index of parameter C_i in the N-d skeleton joint state.

1. Diagonalize Σ to obtain $(\mathbf{v}_j, \sigma_j)_{j=1..N}$.
2. For each chain $C \in \mathcal{C}$, find
vote$_k = \sum_{i=1}^{M} \sum_{j=1}^{N} \sigma_j \mathbf{v}_j[\text{Idx}(C_i)]$
Intuitively this counts the cumulated uncertainty of C along the local covariance principal directions $\mathbf{v}_j$, weighted by their corresponding standard deviations σ_j.
3. Return the chain C with the highest vote. (Alternatively, the best k chains could be returned, or a vote-weighted random one).

BuildInterpretationTree$(\mathbf{s}, C)$
Builds the interpretation tree for $\mathbf{s}$ based on flipping the variables in chain C (§3).

InverseKinematics(P)
Uses either closed-form (§3.1) or iterative (§3.2) inverse kinematics to find the joint-space configuration associated with a list of 3D joint positions P.

Figure 5: The components of our CSS diffusion plus kinematic jump sampling algorithm.

5 Experiments

This section gives experiments showing the performance of our new Kinematic Jump Sampling (KJS) method relative to two established random sampling methods, cost-surface-sensitive Covariance Scaled Sampling (CSS) [14] and the traditional cost-insensitive Spherical Sampling (SS) method used implicitly, *e.g.* in CONDENSATION [7].

5.1 Quantitative Evaluation

Our first set of experiments studies the quantitative behavior of the different sampling methods, particularly their efficiency at locating minima or low-cost regions of parameter space. We study performance for different kinematic partitions of the joint space under deterministic Kinematic Jump Sampling (KJS), and also give results for the random Covariance Scaled (CSS) and Spherical (SS) samplers, showing how different core shapes (spherical vs. local covariance-based) and tail widths (scaled-Gaussian versus Cauchy) affect their efficiency. The study was based on the simple, but still highly multi-modal, model-joint to known-image-joint likelihood function that we use to initialize our 34 d.o.f. articulated model. The model started at approximately its true 3D con-

CONDENSATION samples 'spherically' in the sense that the source of randomness is Gaussian dynamical noise with a fixed prespecified covariance. We could choose coordinates in which this distribution was spherically symmetric. Whereas in CSS, the 'noise' adapts to the local cost surface at each time step.

Our full initialization procedure also estimates some body dimensions, but here these are held fixed.

METHOD	SCALE	NUMBER OF MINIMA	MEDIAN PARAMETER DISTANCE		MEDIAN STANDARD DEVIATION		MEDIAN COST	
			NO OPT	OPT	NO OPT	OPT	NO OPT	OPT
KJS1	-	1024	2.9345	2.8378	92.8345	93.9628	0.0998	0.0212
KJS2	-	1466	3.2568	2.2986	83.4798	82.5709	0.1045	0.0203
CSS	1	8	1.1481	2.5524	10.9351	47.6042	116.9512	8.4968
CSS	4	59	3.2123	2.9474	35.2918	55.3163	1995.1232	6.9810
CSS	8	180	4.9694	3.3466	75.1119	109.8131	16200.8134	7.0986
CSS	16	667	6.4242	6.7209	177.1111	465.8892	45444.1223	8.6958
CSS	1/HT	580	5.0536	6.9362	106.6311	517.3872	15247.7134	8.7242
SS	1	0	0.1993	-	24.5274	-	273.5091	-
SS	4	11	0.7673	2.0492	96.1519	39.0745	4291.1211	6.2801
SS	8	42	1.4726	2.5488	188.1571	56.8268	16856.1211	6.9648
SS	16	135	2.7195	2.8494	367.7461	87.8533	63591.4211	8.6958
SS	1/HT	232	2.1861	6.5474	178.6471	535.9991	18173.1121	17.8807

Table 1: Quantitative results on sample distribution for KJS, as well as CSS and SS with different types of tails (scaled-Gaussian vs. HT, with and without optimization NO OPT vs. OPT). KJS finds 1024 minima in 1024 samples for the first trial and 1466 minima in 1536 samples for the second round. The CSS/SS experiments used 2000 samples. Note that KJS finds many more minima than SS and CSS, and that its samples are already very close to the final minima in cost, whereas SS and CSS samples require a substantial amount of optimization to become plausible hypotheses. Also note that CSS has significantly better performance than SS, both in terms of numbers of minima found and median costs of raw samples.

figuration.

Table 1 summarizes the results, giving the number of minima found by each method, and also their median costs (likelihoods relative to the true configuration) and their distances from the starting configuration in both spherical parameter space units and covariance-scaled standard deviations. It gives statistics both for raw samples, and for samples after local continuous optimization subject to joint and body non-self-intersection constraints. Fig. 6 shows some histograms of numbers of samples and minima found versus parameter space and Mahalanobis distance.

Spherical and Covariance Scaled Sampling: CSS and SS were run with both Gaussian and heavy tailed (HT Cauchy) distributions, using 2000 samples per run. For a fairer comparison we kept the volume of the distribution cores constant: the volume of the unit covariance CSS ellipsoid is always equal to the volume of the corresponding sphere, *i.e.* the sphere's radius is taken to be $R = \sqrt[n]{\lambda_1 ... \lambda_n}$, where λ_i are the eigenvalues of the covariance ellipsoid. We ran the methods for Gaussian distributions with scaling 4,8,16 and Cauchy distributions with scaling 1. Samples that violated physical constraints were projected back onto the feasible constraint surface. This often leads to highly non-Gaussian features such as multi-peaked histograms, even though the raw sampling distribution is Gaussian.

In the results, note the significantly greater number of local minima found by CSS than by SS, and also that CSS samples on average have much lower cost than SS ones. One can also see the large cost difference between unoptimized (NO OPT) and optimized (OPT) samples. Although the table seems to show that SS generates slightly lower-cost optimized minima than CSS, this is illusory. SS is simply too myopic to find more than a few close-lying (and hence low cost) minima, whereas CSS reliably finds both these and also many more distant ones, some of which naturally have somewhat higher cost.

Kinematic Jump Sampling: We ran KJS for several different partitions of the skeleton into sampleable subchains. Experiment KJS1 sampled the left and right shoulder joints and the left calf, for a frontal view similar to the one in fig. 2. Each of the 1024 configurations generated lead to a distinct local minimum after optimization. The second experiment KJS2 sampled the left and right calf joints and the right shoulder joint for a total of 1536 samples leading to 1466 minima after optimization. In both cases the parameter space minima were hit quite accurately, so optimization is largely superfluous. The KJS samples also have far lower costs than raw SS or CSS samples. Thus, KJS sampling is also likely to be effective when used with optimization-free discrete density propagation methods such as CONDENSATION.

5.2 Tracking

Finally, we illustrate the full KJS + CSS method on a 4 s sequence involving full-body tracking of a subject performing agile and rapid dancing moves. This sequence contains both self-occlusion and significant relative motion in depth. It was shot at 25 frames (50 fields) per second against a cluttered,

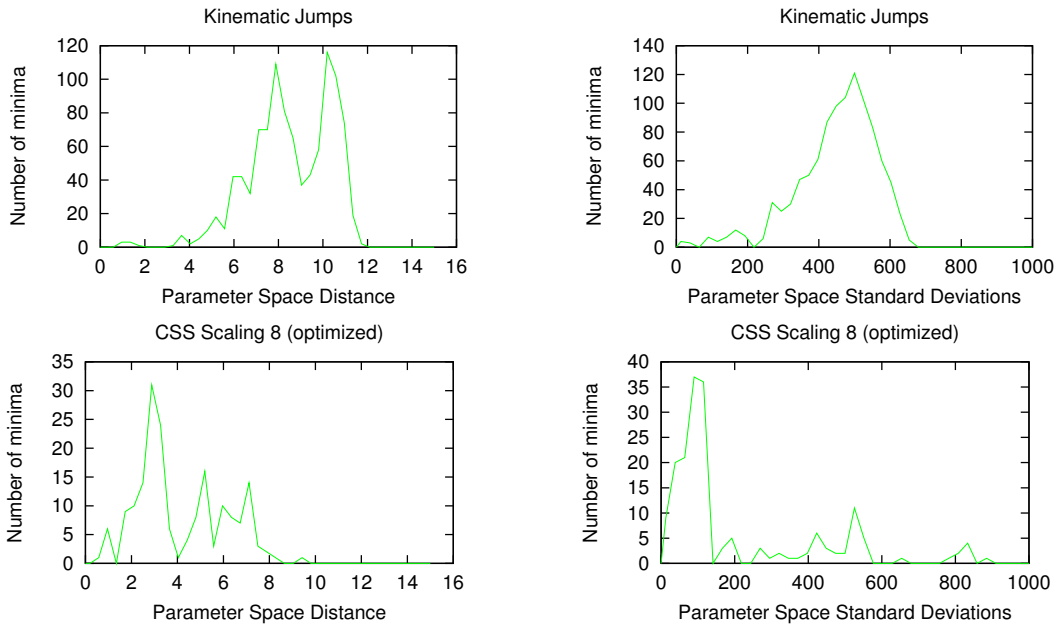


Figure 6: *Top:* Distribution of optimized parameter space distance and standard deviation for the KJS1 experiment. The samples are from the product of the interpretation trees for the left and right shoulder joints and the left calf, for a frontal view similar to fig. 2. *Bottom:* Analogous distributions for Covariance Scaled Sampling (CSS) with scaling factor 8.

unevenly illuminated background, without special clothing or markers. Fig. 7 shows some frames from the original sequence (first row), 2D tracking results showing the current-best model configuration reprojected into the original image (middle row), and the corresponding 3D model pose rendered from a downwards-looking synthetic camera (bottom row). The tracks were initialized by running a method similar to that in §5.1, then selecting an initial set of 8 hypotheses that gave plausible initial body poses. From then on, the full sequence was tracked automatically using an observation likelihood function based on edge and intensity measurements as explained in §2. The sampling procedure was based on CSS diffusion (with scaling 4-6) followed by kinematic jump sampling with closed-form inverse kinematics. The selection of which kinematic sub-chain to sample at a given mode and time was done automatically using the local-uncertainty based voting mechanism described in §5. In this experiment the list $\mathcal{C}$ of user supplied chains contained the short 3-link chains associated with the neck, and each shoulder and each hip. For tracking, one usually needs a search process that does not wander too far from the given prior modes, and these chains have the advantage of generating shallow interpretation trees representing relatively probable local jumps or ambiguities. Such behavior is important not only for efficient and reliabile tracking, but also for the coherence of the post-tracking smoothing process, if any. (No smoothing was done here). The above settings prove highly effective in the sequence analyzed here, as can be seen from the model reprojection both in the original image, and as seen from above.

6 Conclusions

We have presented a novel kinematic sampling framework for recovering 3D human body motion from monocular video sequences. The cost surface for monocular human tracking is structured and highly multi-modal. For any feasible set of image joint positions, there are exponentially many 3D body configurations projecting to it. All of these have similar image projections, and they tend to have similar image likelihoods as well. The different 3D configurations are linked by 'forwards/backwards flipping' moves, one for each kinematic link. Our method uses simple inverse kinematics to systematically generate the complete set of such configurations given any one of them, and hence to investigate the full set of associated cost minima. Our experiments show that kinematic sampling complements and substantially improves on conventional random sampling based trackers, and that it can be used very effectively in tandem with them. The combined system is able to track short sequences involving fast, complex dancing motions in cluttered backgrounds.

Ongoing work is studing whether adding further physical scene constraints can improve the pruning of inconsistent samples, and also investigating the possibility of applying jump-based strategies for non-kinematic ambiguities such as image matching (*e.g.* 'right limb but wrong edge' correspondence errors) and within other MCMC algorithms. We also plan to make a more quantitative evaluation of our voting heuristic, and we are interested in developing smoothing algorithms that are better adapted to long range inter-frame dynamic moves.

Figure 7: Jump kinematics in action! Tracking results for a 4 s agile dancing sequence. *First row:* original images. *Middle row:* 2D tracking results showing the model-image projection of the best candidate configuration at the given time step. *Bottom row:* the corresponding 3D model configuration rendered from above. Note the difficulty of the sequence, the good model image overlap, and the realistic quality of 3D reconstucted model poses.

Acknowledgement Work supported by EU project VIBES.

References

[1] A. Barr. Global and Local Deformations of Solid Primitives. *Computer Graphics*, 18:21–30, 1984.

[2] K. Choo and D. Fleet. People Tracking Using Hybrid Monte Carlo Filtering. In *ICCV*, 2001.

[3] J. Deutscher, A. Blake, and I. Reid. Articulated Body Motion Capture by Annealed Particle Filtering. In *CVPR*, 2000.

[4] J. Deutscher, A. Davidson, and I. Reid. Articulated Partitioning of High Dimensional Search Spacs associated with Articulated Body Motion Capture. In *CVPR*, 2001.

[5] R. Fletcher. Practical Methods of Optimization. In *John Wiley*, 1987.

[6] D. Gavrila and L. Davis. 3-D Model Based Tracking of Humans in Action:A Multiview Approach. In *CVPR*, pages 73–80, 1996.

[7] M. Isard and A. Blake. CONDENSATION – Conditional Density Propagation for Visual Tracking. *IJCV*, 1998.

[8] H. J. Lee and Z. Chen. Determination of 3D Human Body Postures from a Single View. *CVGIP*, 30:148–168, 1985.

[9] J. MacCormick and M. Isard. Partitioned Sampling, Articulated Objects, and Interface-Quality Hand Tracker. In *ECCV*, volume 2, pages 3–19, 2000.

[10] C. Samson, M. Borgne, and B. Espiau. *Robot Control. The Task Function Approach*. Oxford Science Publications, 1991.

[11] H. Sidenbladh, M. Black, and D. Fleet. Stochastic Tracking of 3D Human Figures Using 2D Image Motion. In *ECCV*, 2000.

[12] H. Sidenbladh, M. Black, and L. Sigal. Implicit Probabilistic Models of Human Motion for Synthesis and Tracking. In *ECCV*, 2002.

[13] C. Sminchisescu. Consistency and Coupling in Human Model Likelihoods. In *FGR*, pages 27–32, Washington D.C., 2002.

[14] C. Sminchisescu and B. Triggs. Covariance-Scaled Sampling for Monocular 3D Body Tracking. In *CVPR*, volume 1, pages 447–454, Hawaii, 2001.

[15] C. Sminchisescu and B. Triggs. Building Roadmaps of Local Minima of Visual Models. In *ECCV*, volume 1, pages 566–582, Copenhagen, 2002.

[16] C. Sminchisescu and B. Triggs. Hyperdynamics Importance Sampling. In *ECCV*, volume 1, pages 769–783, Copenhagen, 2002.

[17] C. J. Taylor. Reconstruction of Articulated Objects from Point Correspondences in a Single Uncalibrated Image. In *CVPR*, pages 677–684, 2000.

[18] D. Tolani, A. Goswami, and N. Badler. Real-Time Inverse Kinematics Techniques for Anthropometric Limbs. *Graphical Models*, 62:353–388, 2000.

Chapter 2

2D Model Based Human Detection and Motion Capture

This chapter contains two papers that approach the problem of extracting useful information from images of humans from a purely 2D, image-based perspective. Both methods use "scaled prismatic model" style articulated 2D body models [JBY96, CR99].

Summary of paper 5, "Learning to Parse Pictures of People"

This paper, published in the 2002 European Conference on Computer Vision [RST02], describes a method for detecting humans in static images and labelling their major body sections. The algorithm is based the dynamic programming approach to assembling possible human parts into a coherent whole [FE73, FH00, IF01]. It adds individual part detectors based on Support Vector Machines working over derivative energy based image descriptors to this approach, to allow the detector to handle a wider range of images. The method is trained and tested on images from the MIT pedestrian image database.

Summary of paper 6, "Tracking Articulated Motion with Piecewise Learned Dynamical Models"

This paper, published in the 2004 European Conference on Computer Vision [AT04c], develops a 2D human tracker based on a 2D "scaled prismatic model" body representation [CR99]. The main contribution is the use of a learned dynamical model to help stabilize tracking, particularly through rapid motions and changes of model aspect. The model is a mixture of second order linear autoregressors, with each autoregressor learned as follows. K-means is used to cluster the training data in state space, then for each patch PCA based dimensionality reduction is applied to the state vectors to stabilize the ensuing estimation, an autoregressive model is learned, and the resulting models are lifted back to the original dimensionality. Finally, an EM-like re-clustering iteration is run to reassign training states to the models that fit them best, and readjust these models to fit the assigned points.

Learning to Parse Pictures of People

Remi Ronfard, Cordelia Schmid and Bill Triggs*
INRIA, 655 avenue de l'Europe, 38330, Montbonnot, France

Abstract

Detecting people in images is a key problem for video indexing, browsing and retrieval. The main difficulties are the large appearance variations caused by action, clothing, illumination, viewpoint and scale. Our goal is to find people in static video frames using learned models of both the appearance of body parts (head, limbs, hands), and of the geometry of their assemblies. We build on Forsyth & Fleck's general 'body plan' methodology and Felzenszwalb & Huttenlocher's dynamic programming approach for efficiently assembling candidate parts into 'pictorial structures'. However we replace the rather simple part detectors used in these works with dedicated detectors learned for each body part using Support Vector Machines (SVMs) or Relevance Vector Machines (RVMs). We are not aware of any previous work using SVMs to learn articulated body plans, however they have been used to detect both whole pedestrians and combinations of rigidly positioned subimages (typically, upper body, arms, and legs) in street scenes, under a wide range of illumination, pose and clothing variations. RVMs are SVM-like classifiers that offer a well-founded probabilistic interpretation and improved sparsity for reduced computation. We demonstrate their benefits experimentally in a series of results showing great promise for learning detectors in more general situations.

Keywords: object recognition, image and video indexing, grouping and segmentation, statistical pattern recognition, kernel methods.

1 Introduction

Detecting people in images is an important practical challenge for content-based image and video processing. It is difficult owing to the wide range of appearances that people can have. There is a need for methods that can detect people in general everyday situations. For instance, actors in typical feature films are shown in a great variety of activities, scales, viewpoints and lightings. We can not rely on frequently-made simplifying assumptions such as non-occlusion, perfect background subtraction, *etc.*

To address this issue, Forsyth & Fleck introduced the general methodology of *body plans* [8] for finding people in images. However, they relied on simplistic body part detectors based on generalized cylinders. This is problematic, especially in the case of loose clothing. Similarly, Felzenszwalb & Huttenlocher [6] showed how dynamic

*Appeared in 2002 European Conf. on Computer Vision. Work supported by European Union FET-Open research project VIBES.

programming could be used to efficiently group body plans cast as 'pictorial structures' [7], but they relied on simplistic colour-based part detectors. Both of these works make strong photometric assumptions about the body parts. We retain their ideas for composing parts into assemblies by building tree-structured models of people, but propose a more general approach to learning the body part detectors and the underlying geometric model, based on Support Vector Machines (SVM) [24, 4] or Relevance Vector Machines (RVM) [22, 23]. In the past, SVM classifiers have been learned for entire humans [18] and also for rigidly connected assemblies of subimages (typically, upper body, arms, and legs) [16], but not for flexibly articulated body models.

We present a series of experiments showing the promise of learning the articulated structure of people from training examples with hand-labelled body parts, using SVMs or RVMs. Our contribution is three-fold. Firstly, our feature set and training method builds reasonably reliable part detectors from as few as 100 hand-labelled training images, and the final RVM detectors are very efficient, often involving comparison with only 2–3 positive and 2–3 negative exemplars. Secondly, we sketch a method for learning a body joint model using the recently proposed Adaptive Combination of Classifiers (ACC) framework [16]. Thirdly, we describe an efficient decoder for the learned models, that combines kernel based detection with dynamic programming. Our initial experiments demonstrate that body part detectors learned with only 100 images from the MIT pedestrian database can give reliable detection with as few as 4 false alarms per image on this data set. This is remarkable as even humans often find it difficult to classify the isolated part subimages correctly. The detected parts can be efficiently assembled into correct body plans in 70% of cases.

The paper is structured as follows. We introduce our body plan model in §2, then discuss body part detectors learned by two competing algorithms, SVM and RVM, in §3. §4 presents our approach for learning and decoding body plans. Finally, §5 presents some results and discusses future work.

2 The Pictorial Structure of People

In the work of Marr & Nishihara [15] and others [10, 19], people are described as hierarchical 3D assemblies of generalized cylinders and components. The position of a part C relative to its parent P is parametrized by C's position (p, r, θ) and angular orientation (ψ, ϕ, χ) in P's cylindrical coordinate system. Each joint is thus represented as a 6-vector, with discrete toleranced values for each parameter. They note that perspective projection makes many parameters unobservable and that the image signature of a joint is a pair of axes, but still emphasize, and attempt to recover, 3D structure.

Recovering articulated 3D models from single images is difficult. Felzenszwalb & Huttenlocher recently reconsidered Fischler & Elschlager's notion of *pictorial structure* [7] and demonstrated its usefulness for detecting people in indoor scenes [6]. Pictorial structures are collections of image parts arranged in deformable configurations. They are directly adapted to monocular observations. Similarly, Morris & Rehg argued that 3D tracking singularities can be removed using image based 'scaled prismatic models' [17] — essentially, pictorial structure models. Other 2D part-based models use image edges [25] or motion models derived from dense optical flow [14] as features for

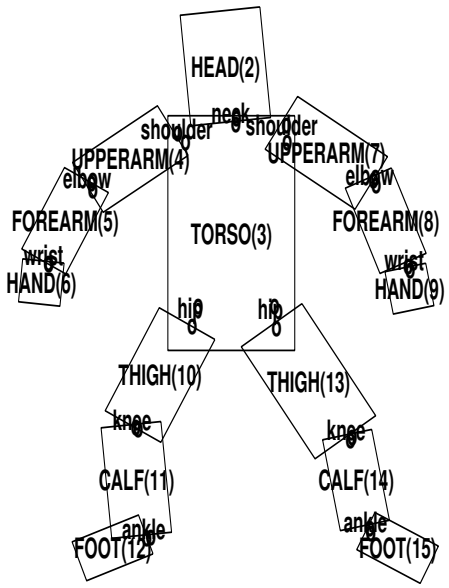


Figure 1: Our articulated body model with its 14 joints and 15 body parts.

detection and/or tracking.

Following this line of research, we represent people using a 2D articulated appearance model composed of 15 part-aligned image rectangles surrounding the projections of body parts: the complete body, the head, the torso, and the left and right upper arms, forearms, hands, thighs, calves and feet, numbered from 1 to 15 as in Figure 1. Each body part P_i is a rectangle parametrized in image coordinates by its centre $[x_i, y_i]$, its length or size s_i and its orientation θ_i. A coarse resolution whole-body image is also included in case 'the whole is greater than the sum of the parts'. During training and detection, we discretize the admissible range of sizes and orientations. As discussed later, we use a range of 8 scales, and 36 orientations equally spaced every 10 degrees. 14 body joints connect the parts: the plexus between body and torso, the neck between head and torso, the hips between torso and thighs, the knees between thighs and calves, the ankles between calves and feet, the shoulders between torso and upper arms, the elbows between upper arms and forearms and the wrists between forearms and hands. Figure 1 shows the body model in average position, using a single aspect ratio of 16:9 for all body parts.

Expressed in terms of the probabilistic formulation of pictorial structure, the posterior likelihood of there being a body with parts P_i at image locations l_i ($i \in \{1...15\}$) is the product of the *data likelihoods for the 15 parts* (*i.e.* the classification probabilities for the observed subimages at the given part locations to be images of the required parts) and the *prior likelihoods for the 14 joints* (*i.e.* the probabilities for a coherent body to generate an image with the given relative geometric positionings between each part and its parent in the body tree). The negative log likelihood for the whole body

assembly $A = \{l_1, \ldots, l_{15}\}$ can thus be written as follows, where E is the list of body joints ('edges' of the body tree):

$$L(A) \;=\; -\sum_i \log p_i(l_i) - \sum_{(ij)\in E} d_{ij}(l_i, l_j)$$

Felzenszwalb & Huttenlocher model body parts as constant color regions of known shapes and body joints as rotational joints. In this paper, we machine-learn the 29 distributions $p_i(l_i)$ and $d_{ij}(l_i, l_j)$ from sets of positive and negative examples. We model the part and articulation likelihoods using linear Support Vector or Relevance Vector Machines. Our work can be viewed as an extension of Mohan's recent work on *combined classifiers* [16], where 'component' classifiers are trained separately for the limbs, torso and head based on image pixel values, and 'combination' classifiers are trained for the assemblies based on the scores of the component classifiers in fixed image regions. However, we learn part-aligned, rather than image-aligned, classifiers for each body part, and we extend the 'combination' classifier to include deformable, articulated structures rather than rigid assemblies.

3 Detecting Body Parts

In our model, learning each body part amounts to estimating its probability given the observed image distribution at its location. Detecting and labelling body parts is a central problem in all component-based approaches. Clearly the image must be scanned at all relevant locations and scales, but there is also a question of how to handle different part orientations, especially for small, mobile, highly articulated parts such as arms and hands. One can work either in the image frame, trying to build a general detector that is capable of finding the part whatever its orientation, or in a part-aligned frame, building a detector that works for just one orientation and scanning this over all relevant orientations. The part-aligned approach has the potential to produce simpler detectors from less (but better labelled) training data, and the advantage that it also recovers the part orientation. Which approach is faster or better must depend on the relative complexity and reliability of all-orientation and one-orientation detectors, but in general it is difficult to build good transformation invariance into general-purpose detectors. The image-frame approach is well adapted to pedestrian detection applications such as Mohan's [16], where one wants a relatively coarse whole person detector for distant people with similar poses (mainly standing or walking). But our ultimate goal is to detect people and label them with detailed part locations, in applications where the person may be in any pose and partly occluded. For this we believe that the part-based body plan approach is preferable.

Our detector works with a generalized feature pyramid spanning 8 scales and 36 orientations $0^\circ \ldots 350^\circ$. During training, the articular structure of each training image is clicked, and for each designated part a 14×24 subimage aligned with its axes and scaled to its size is extracted as shown in Figure 2. We learn 15 Support Vector or Relevance Vector Machines for the individual parts and the whole body, and during detection run each of them over the scale-orientation-position feature pyramid, then assemble the results as discussed in the next section.

Figure 2: A hand-labelled training image from the MIT database and its extracted body part subimages. Reading vertically from left to right: left upper arm, forearm, hand; left thigh, calf and foot; head, torso and whole body; right thigh, calf, foot; right upper arm, forearm and hand.

3.1 Feature Sets

The problem of choosing features for object recognition has received a lot of interest in recent years and numerous feature sets have been suggested, including image pixel values, wavelet coefficients and Gaussian derivatives. Wavelets are currently popular, but as a general representation for human body parts it is unclear whether standard (rectangular) or non-standard (square) wavelet constructions are most suitable [9, 16]. Heisele *et al* obtained better results for their SVM face detector using gray levels rather than Haar wavelets [9]. Some authors also feel that wavelets are unsuitable as a general image representation because they represent point events rather than line or curve ones, and instead propose ridgelets and curvelets [2, 5]. These might prove useful for detecting human limbs.

Here we leave such issues for future work and use a feature set consisting of the Gaussian filtered image and its first and second derivatives. Although simple, these features seem to represent the variations of body part detail effectively over a range of scales and orientations. The feature vector for an image rectangle at location-scale-orientation $[x_i, y_i, s_i, \theta_i]$ contains the absolute values of the responses of the six Gaussian $\sigma = 1$ filters $\{G, \nabla_x G, \nabla_y G, \nabla_{xx} G, \nabla_{xy} G, \nabla_{yy} G\}$ in the rectangle's (rescaled and reoriented) 14×24 window. There are thus $14 \times 24 \times 6 = 2016$ features per window. For color images we use only the luminance values Y. The absolute values of the filter responses are normalized across each image. The extracted features are not

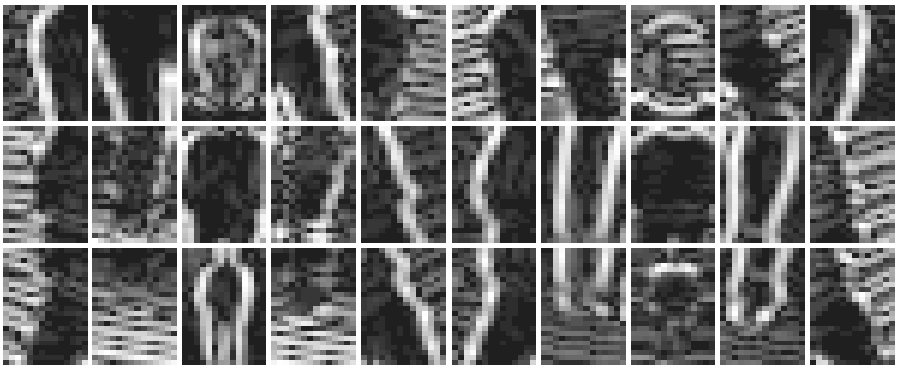


Figure 3: The $\nabla_x G$ and $\nabla_y G$ feature images for the example in Figure 2.

required to be scale- or orientation-invariant. On the contrary, we seek features that are tuned to the characteristic scales and orientations of the detail in the aligned body-part images. Some examples of the feature vectors are shown in Figure 3.

To implement this, the Gaussian filters are computed using 9 rotated images from 0 to 80 degrees and 8 scales. We resample according to scale in each window, so the standard deviation of the filters in their resampled 14×24 windows is always 1. For any given size and orientation, we select the feature vector that best approximates the part-aligned region as an axis-aligned rectangle of height 24. This choice of primitives makes reasonably few assumptions about the nature of the features to be learned, which can be arbitrary combinations of shape, luminance and texture.

3.2 Training

Using the 2016-dimensional feature vectors for all body parts in the training set, we trained two linear classifiers for each part, one using a Support Vector Machine and the other using a Relevance Vector Machine. SVMs and RVMs are grounded on statistical learning results that suggest that they should give good classification performance even when there are relatively few training examples. Here we decided to put this claim to a severe test by training on the smallest sets of examples that give reasonable results — in our case, about 100.

We trained the 15 part classifiers individually against a common 'background' data set consisting of random pieces of the training images that do not contain people. Note that we are not attempting to learn isolated part detectors or multi-class part-type classifiers, but *reliable filters* for rejecting non-parts within an articulated body plan framework. We expect the overlap in appearance between different parts to be significant, but we do not want this to cause missed detections in ambiguous cases.

Support Vector Machines: SVMs are discriminant classifiers that give a yes/no decision, not a probability. However in our experiments we treat the SVM scores (scalar products in feature space) as if they were log likelihoods for the body parts given the

image values[1].

Relevance Vector Machines: RVMs [22, 23] are Bayesian kernel methods that choose sparse basis sets using an 'Automatic Relevance Determination' [1] style prior that pushes non-essential weights to zero. They do not usually give significantly better error rates than the corresponding SVMs, but they do often give similar results with many fewer kernels. The functional form of the final classifier is the same as that of an SVM — only the fitted weights are different. Here we use logistic linear discriminant RVMs, whose output directly models the log-odds for a part versus a non-part at the given point. In this paper, we use RVMs mainly to reduce the number of kernels ('relevance vectors') and hence the computational complexity. The trained RVM classifiers typically use only 2–3 positive and 2–3 negative relevance vectors each, as compared to 100–200 support vectors for a comparable SVM classifier.

Currently we train the linear RVMs to make sparse use of *examples*, but they could also be trained to make sparse use of *features*. This would potentially mean that fewer image features would have to be extracted, and hence that the method would run faster. We plan to investigate this in future work.

3.3 Detection

We detect all of the body parts at once, in a single scan over the orientation-scale pyramid. The detection score for each part reduces to a simple convolution product against a mask containing the discriminant sum of weighted support or relevance vectors. Conceptually, this amounts to generalized template matching over images of local feature vectors, with weighted sums of training examples as templates. The nonlinearity of the process is hidden in the rectified, normalized local feature vectors. For efficiency in the assembly stage, we currently retain only the 50 best candidates for each part. The observed detection rates suggest that this strategy suffices for simple images, but it is not ideal for robustness against occlusions and we ultimately plan to use a more sophisticated strategy based on adaptive thresholds.

4 Parsing the body tree

In a non-articulated, image-aligned method such as that of Mohan [16], assembling the part detections is relatively straightforward: decompose the search window into subwindows, keep the highest score for the appropriate part in each subwindow, and compose the scores into a single, low-dimensional feature vector. Given these second-stage feature vectors, a single linear SVM can be learned for the overall body detection.

In our articulated, part-aligned method, the composition of part-models is only slightly more difficult, and can be cast as a combinatorial search: from all detected parts, search for the assemblies looking most like people. Since assemblies are natu-

[1] A more principled approach to converting the scores of a discriminant classifier to probabilities is as follows: run the detector over a validation set and fit density models to its positive-example and negative-example output scores. At any given score, the ratio of the positive-example density to the negative-example one is an estimate of the positive-to-negative odds ratio for detections at that score.

rally described as trees, efficient dynamic programming algorithms can be used to build the second-stage classifier, as we now describe.

4.1 Parsing/decoding algorithm

Given N candidate body part locations l_{kn} detected by each body part classifier C_k, we are looking for a 'parse' of the scene into one or more 'body trees'. One important subproblem is to assign a 'valid detection' or 'false alarm' label to each candidate, based not only on the candidate's scores, but on the local configuration between the candidates and its neighbours. Our approach relies on an extension of the Viterbi decoding algorithm, as described by Ioffe & Forsyth [13] and Felzenszwalb & Huttenlocher [6], which we sketch only briefly here. Given the detection scores $D_k(l_{kn})$ for all candidates $n = 1...N$, we search for the best candidate as a function of their direct parents $pa(n)$ in the body tree. For the leaves (i.e. hands, feet and head), this is computed by algorithm 1:

Algorithm 1 leaf location

$$B_k(l_{jm}) = \min_{\{n=1...N\}} -D_k(l_{kn}) + d_{kj}(l_{kn}, l_{jm})$$
$$l_k^*(l_{jm}) = \arg\min_{\{n=1...N\}} -D_k(l_{kn}) + d_{kj}(l_{kn}, l_{jm})$$

Based on this computation, we can score candidates from the bottom up, using the recursion algorithm 2:

Algorithm 2 bottom up

$$B_k(l_{jm}) = \min_{\{n=1...N\}} -D_k(l_{kn}) + d_{kj}(l_{kn}, l_{jm}) + \Sigma_{\{c|k=pa(c)\}} B_c(l_{kn})$$
$$l_k^*(l_{jm}) = \arg\min_{\{n=1...N\}} -D_k(l_{kn}) + d_{kj}(l_{kn}, l_{jm}) + \Sigma_{\{c|k=pa(c)\}} B_c(l_{kn})$$

At the root node we obtain the simple formula 3 for scoring the high level hypotheses.

Algorithm 3 root location

$$B_r = \min_{\{n=1...N\}} -D_r(l_{rn}) + \Sigma_{\{c|r=pa(c)\}} B_c(l_{rn})$$
$$L_r^* = \arg\min_{\{n=1...N\}} -D_r(l_{rn}) + \Sigma_{\{c|r=pa(c)\}} B_c(l_{rn})$$

Choosing the most probable root node, we can then assign the other nodes in a top down fashion by choosing $L_k^* = l_k^*(L_{pa(k)})$ for each node given its parent. Note that this algorithm has a complexity $O(MN^2)$ with M the number body parts and N the number of candidates per body part. As an example of the detection results obtained with this method, Figure 6 shows the three most probable parses for four test images, ranked in order of decreasing likelihood.

4.2 Learning the body tree

The cost functions used in our body tree model are based on geometric constraints on the relative positions of parts at a body articulation, as in Felzenszwalb & Huttenlocher

[6]. Essentially, the articulation model is a linear combination of the differences between two joint locations, as predicted separately by the two body parts meeting at the articulation.

Algorithm 4 joint distance(l_i, l_j)

Compute joint location x_{ij}, y_{ij} given first body part location l_i
Compute joint location x_{ji}, y_{ji} given second body part location l_j
Return $d_{ij} = w_{ij}^x|x_{ij} - x_{ji}| + w_{ij}^y|y_{ij} - y_{ji}| + w_{ij}^\theta|\theta_i - \theta_j - \theta_{ij}| + w_{ij}^s|\log \frac{s_i}{s_j} - \log s_{ij}|$

Each body joint is parametrized by the relative sizes s_{ij} and angles θ_{ij} between its parts, and the four rigidity parameters $w_{ij}^x, w_{ij}^y, w_{ij}^\theta, w_{ij}^s$ governing the admissible range of apparent deformations of the articulation in position, size and orientation. We learned the relative sizes s_{ij} and angles θ_{ij} of each articulation by simply taking the average relative positions of all pairs of body parts over the training set.

To learn the rigidity parameters, we again used a Support Vector Machine. For each articulation A_{ij} between parts P_i and P_j, we learned a 'combination classifier' based on a five-dimensional feature vector $F_i^0 = D_i + D_j$, $F_i^x = |x_{ij} - x_{ji}|$, $F_i^y = |y_{ij} - y_{ji}|$, $F_i^\theta = |\theta_i - \theta_j - \theta_{ij}|$, $F_i^s = |\log \frac{s_i}{s_j} - \log s_{ij}|$.

Using positive and negative examples from our training set, we used a linear SVM classifier to learn a set of weights $w_{ij}^0, w_{ij}^x, w_{ij}^y, w_{ij}^\theta, w_{ij}^s$ such that the score is positive for all positive example, and negative for all negative examples. We experimentally verified that the learned weights have the expected signs, $w_{ij}^0 > 0$ and $w_{ij}^x < 0, w_{ij}^y < 0, w_{ij}^\theta < 0, w_{ij}^s < 0$, so that the learned model can indeed be related to the log-likelihood of the articulation

$$L(A_{ij}) = F_i^0 - \frac{|w_{ij}^x|}{w_{ij}^0}F_i^x - \frac{|w_{ij}^y|}{w_{ij}^0}F_i^y - \frac{|w_{ij}^\theta|}{w_{ij}^0}F_i^\theta - \frac{|w_{ij}^s|}{w_{ij}^0}F_i^s$$

In our experiments with the MIT pedestrian database, the learned models performed slightly better than the naive approach of assigning equal weights to all parameters and all articulations, and we expect the method to be of even greater benefit for dealing with the more complicated cases of people in action such as running or jumping.

5 Implementation and results

We implemented and tested our method in Matlab. The system consists of several components. There is an interactive program for hand-labelling examples and storing the locations of the body joints and parts. Another function computes image pyramids and extracts image signatures at all locations x, y, s, θ . These are used both to generate feature vectors for SVM/RVM training, and to perform detection against the learned models. Finally, a parser based on the above dynamic programming approach reads candidate locations from the 15 body part detectors and produces a ranked list of candidate assemblies.

We used MIT's public domain program SvmFu-3.0 to train the SVM classifiers. We trained the RVM classifiers in Matlab using a new algorithm that will be described in detail elsewhere.

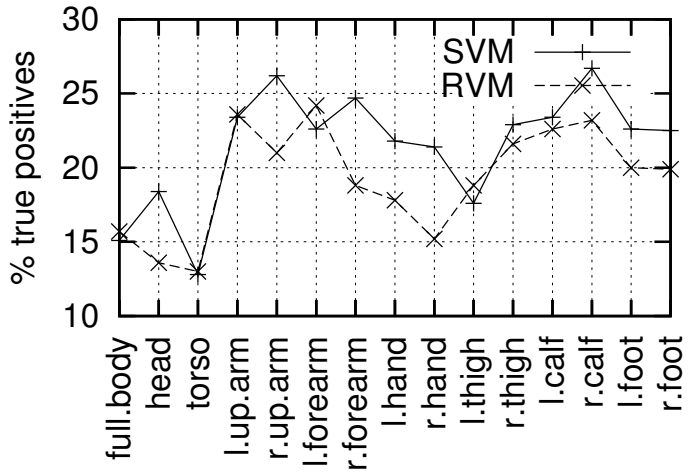


Figure 4: True positive rates for SVM and RVM body part detectors.

5.1 Experimental setup

We selected 100 frontal images from the MIT pedestrian database and labelled their 15 parts, as shown in Figure 2. Each example is labelled by clicking 14 body joints. Occluded parts are clicked at their most likely (hidden) location, but flagged as occluded. Only visible parts are used to train the image part models, but the hidden parts can be included when training the geometric models. We also picked 5 background regions in each image, for use as negative examples. As a result, each body part classifier was trained with slightly less than 100 positive examples, and 500 negative examples.

Separate examples are needed for training and testing, so we selected and labelled another 100 images from the MIT pedestrian database to serve as a test set. This was used to evaluate the body part and assembly detectors.

5.2 Detection of body parts.

Detectors are traditionally compared by tracing ROC curves, i.e. true detection rates (recall) as a function of false alarm rates ($1-$precision). In our case the detectors must be tuned to function as filters, so most important parameter is the false alarm rate needed to achieve 'total recall'. Hence, we compared the two detectors by measuring the false detection rates required to detect all visible body parts in our test set. The resulting true positive rates for each part detector are shown in Figure 4.

As can be seen, individual part images are not very discriminative, so the absolute false alarm rates remain quite high. In fact, they become still higher (up to 15:1) once confusions between parts are included. Even so, the linking stage manages to resolve most of the ambiguity, and the number of candidates that have to be examined remains

quite tractable, at most about 75 candidates per part for these images. Ignoring spatial contiguity, the worst-case number of body joint hypotheses is therefore $14 \times 75^2 = 78750$. In practice, we observed an average number closer to $14 \times 20^2 = 5600$ and used 50 candidates as a safe bet in all of our experiments. The RVM classifiers perform only slightly worse than their SVM counterparts, with mean false detection rates of 80.1% and 78.5% respectively. This is remarkable given the very small number of relevance vectors used by the RVM detectors. For the purpose of rapid filtering, the advantages of the RVM clearly outweigh their inconvenience.

Also note that the worst results are obtained for the torso (3) and head (2) models. The torso is probably the hardest body part to detect as it is almost entirely shapeless. It is probably best detected indirectly from geometric clues. In contrast, the head is known to contain highly discriminant features, but the training images contain a wide range of poses and significantly more training data (and perhaps some bootstrapping on false alarms) is probably needed to build a good detector.

5.3 Detection of body trees

We evaluated the final body detector by visually comparing the best (highest probability) three configurations returned with the correct interpretation in each of the 100 test set images. Thus, the task was purely that of detecting humans using the 50 best candidates for each body part and the body tree model. Our first experiment used 100 training exemples. We obtained correct detections rates of 72 % using RVM scores and 83 % using SVM scores, while using a naive geometric model with uniform rigidity parameters for all of the body joints. We then learned a geometric model using labelled body joints from the 100 training images. We used the correct assemblies as positive examples, and circular permutations of the body parts as negative ones. Using the learned model, the correct detection rates improved to 74 % and 85 %. We should note that detection is a relatively easy task with this data set, and our method should be evaluated also with regards to the pose estimates. We plan to investigate this area quantitatively in later work. Qualitatively, we noted that a majority of the body parts were correctly positioned in only 36 % of the test images for RVM and 55 % for SVM.

In a second experiment, we increased the size of the training set to 200 examples. This resulted in a slight increase of the detection rates, to 76 % for SVM and 88 % for RVM, and a much vaster improvement of the pose estimates, resulting in qualitatively correct poses in 54 % of the test examples for RVM and 75 % for SVM.

6 Discussion and Future Work

The good detection rates achieved by the method make a convincing case that the body-plan strategy is applicable to real problems in image and video indexing. We plan to extend this work to video, where we hope to improve the detection rates even further by making use of temporal and kinematic constraints. But the construction of the image pyramid is computationally expensive, and we plan to move to a more efficient implementation, which could rely on a more thorough selection of the feature vectors. One way to do this will be to use RVM classifiers that learn relevant features rather

Figure 5: Part detection results from test collection.

than relevant examples. As a complement, Sidenbladh & Black's [20, 21] approach for learning the image statistics of people vs. background could prove useful for learning better models by selecting better features. In the assembly phase, the complexity of the dynamic programming algorithm is quadratic n the number of candidate parts which need to be stored, which in turn depends on the precision of the individual body part detectors. By fine-tuning the body part detectors, we expect to achieve significant improvements also in the overall performance of the global detector.

Further work will be needed for assessing the correctness of the detection and pose estimation results in a more systematic way and for 'bootstrapping' the learned models (adding examples on which our current model fails, and retraining). Even without bootstrapping, we have verified experimentally that the quality of the body part classifiers

Figure 6: Ranked detections and their energies, using the learned body model and SVM scores.

is improved significantly by increasing the size of the training data. We will need to quantify this observation in future work.

We also plan to extend the method to handle multiple persons in a greater variety of backgrounds and poses, by explicitly representing occlusions in the decoding process as in the work of Coughlan et al. [3] or by introducing mixtures of partial body trees, as in the recent proposal made by Ioffe and Forsyth [11, 12]. The cost functions used to evaluate the assembly of the body plans could also benefit from a richer geometric model and additional photometric constraints (e.g. similarity of color and texture between the body parts for the same person). There are cases where we would like to move even further away from the human anatomic model, and replace it with a small set of 'clothing models', which could be learned in much the same way and provide additional flexibility. Those are avenues for further experimental work.

7 Conclusion

Detecting humans is a challenging problem in computer vision, with considerable practical implications for content-based indexing. We believe we have reached three useful concusions with the work reported in this paper. Firstly, it is possible to learn appearance models for human body parts from examples and to use them as input to a body plan parser, at least for a modest-size problem such as pedestrian detection. Secondly, we have been able to learn geometric models for the combination of the detected parts, allowing us to robustly estimate the likelihood of a body part assembly, without recourse to sampling or HMM distributions, which require thousands of examples to be learned efficiently. Thirdly, the learned models lead to an efficient decoding algorithm that combines kernel based learning and dynamic programming techniques, and is simple enough to be extended to video sequences.

References

[1] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[2] E. Candes and D. Donoho. Curvelets – a surprisingly effective nonadaptive representation for objects with edges. In L. L. Schumaker et al., editor, *Curves and Surfaces*. Vanderbilt University Press, 1999.

[3] J. Coughlan, D. Snow, C. English, and A. Yuille. Efficient optimization of a deformable template using dynamic programming. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2000.

[4] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other Kernel Based Learning Methods*. Cambridge University Press, 2000.

[5] M. Do and M. Vetterli. Orthonormal finite ridgelet transform for image compression. In *Int. Conference on Image Processing*, volume 2, pages 367–370, 2000.

[6] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient matching of pictorial structures. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2000.

[7] M. Fischler and R. Elschlager. The representation and matching of pictorial structures. *IEEE Trans. Computer*, C-22:67–92, 1973.

[8] D. Forsyth and M. Fleck. Body plans. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1997.

[9] B. Heisele, T. Poggio, and M. Pontil. Face detection in still gray images. Technical report, AI Memo 1687, Massachusetts Institute of Technology, 2000.

[10] D. Hogg. Model-based vision: A program to see a walking person. *Image and Vision Computing*, 1(1):5–20, 1983.

[11] Sergey Ioffe and David Forsyth. Human tracking with mixtures of trees. In *Proc. Int. Conf. Computer Vision*, 2001.

[12] Sergey Ioffe and David Forsyth. Mixtures of trees for object recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2001.

[13] Sergey Ioffe and David Forsyth. Probabilistic methods for finding people. *Int. J. of Computer Vision*, 43(1), 2001.

[14] S. Ju, M. Black, and Y. Yacoob. Cardboard people: a parameterized model of articulated image motion. In *Int. Conference on Automatic Face and Gesture Recognition*, 1996.

[15] D. Marr and H.K. Nishihara. Representation and recognition of the spatial organization of three dimensional structure. *Proceedings of the Royal Society of London B*, 200:269–294, 1978.

[16] Anuj Mohan, Constantine Papageorgiou, and Tomaso Poggio. Example-based object detection in images by components. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(4), 2001.

[17] D. Morris and J. Rehg. Singularity analysis for articulated object tracking. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1998.

[18] C. Papageorgiou. Object and pattern detection in video sequences. Technical report, Master's thesis, Massachusetts Institute of Technology, 1997.

[19] K. Rohr. Incremental recognition of pedestrians from image sequences. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 8–13, 1993.

[20] H. Sidenbladh and M. Black. Learning the statistics of people in images and video. *Int. Journal of Computer Vision*, 2001.

[21] H. Sidenbladh, F. Torre, and M. Black. A framework for modeling the appearance of 3d articulated figures. In *Int. Conference on Automatic Face and Gesture Recognition*, 2000.

[22] M. Tipping. The relevance vector machine. In *Advances in Neural Information Processing Systems*. Morgan Kaufmann, 2000.

[23] M. E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

[24] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[25] Liang Zhao and Chuck Thorpe. Recursive context reasoning for human detection and parts identification. In *IEEE Workshop on Human Modeling, Analysis and Synthesis*, 2000.

Tracking Articulated Motion with Piecewise Learned Dynamical Models

Ankur Agarwal and Bill Triggs
GRAVIR-INRIA-CNRS, 655 Avenue de l'Europe, Montbonnot 38330, France
{*Ankur.Agarwal,Bill.Triggs*}*@inrialpes.fr*
http://www.inrialpes.fr/lear/{*agarwal,triggs*}

December 25, 2004

Abstract

We present a novel approach to modelling the non-linear and time-varying dynamics of human motion, using statistical methods to capture the characteristic motion patterns that exist in typical human activities. Our method is based on automatically clustering the body pose space into connected regions exhibiting similar dynamical characteristics, modelling the dynamics in each region as a Gaussian autoregressive process. Activities that would require large numbers of exemplars in example based methods are covered by comparatively few motion models. Different regions correspond roughly to different action-fragments and our class inference scheme allows for smooth transitions between these, thus making it useful for activity recognition tasks. The method is used to track activities including walking, running, *etc.*, using a planar 2D body model. Its effectiveness is demonstrated by its success in tracking complicated motions like turns, without any key frames or 3D information.

1. Introduction

Tracking and analyzing human motion in video sequences is a key requirement in several applications. There are two main levels of analysis: *(i)* detecting people and tracking their image locations; and *(ii)* estimating their detailed body pose, *e.g.* for motion capture, action recognition or human-machine-interaction. The two levels interact, as accurate detection and tracking requires prior knowledge of pose and appearance, and pose estimation requires reliable tracking. Using an explicit body model allows the state of the tracker to be represented as a vector of interpretable pose parameters, but the problem is non-trivial owing to the great flexibility of the human body, which requires the modelling of many degrees of freedom, and the frequent non-observability of many of these degrees of freedom in monocular sequences owing to self-occlusions and depth ambiguities. In fact, if full 3D pose is required from monocular images, there are potentially thousands of local minima owing to kinematic flipping ambiguities [18]. Even without this, pervasive image ambiguities, shadows and loose clothing add to the difficulties.

Previous work: Human body motion work divides roughly into *tracking based approaches*, which involve propagating the pose estimate from one time step to another, and *detection based approaches*, which estimate pose from the current image(s) alone. The latter have become popular recently in the form of 'exemplars' [21] and 'key frames' [19]. These methods allow the direct use of image data, which eliminates the need for predefined parametric models. But the interpretability of parametric models is lost, and large numbers of exemplars are needed to cover high dimensional example spaces such as those of human poses. (Tree-based structures have recently been explored for organizing these datasets [20], but they rely on the existence of accurate distance metrics in the appearance space).

Within the tracking framework, many methods are based on computing optical flow [9, 3, 2], while others optimize over static images (*e.g.* [18]). On the representation side, a variety of 2D and 3D parametric models

Published in 2004 European Conf. on Computer Vision.
This research was supported by European Union research projects VIBES and LAVA.

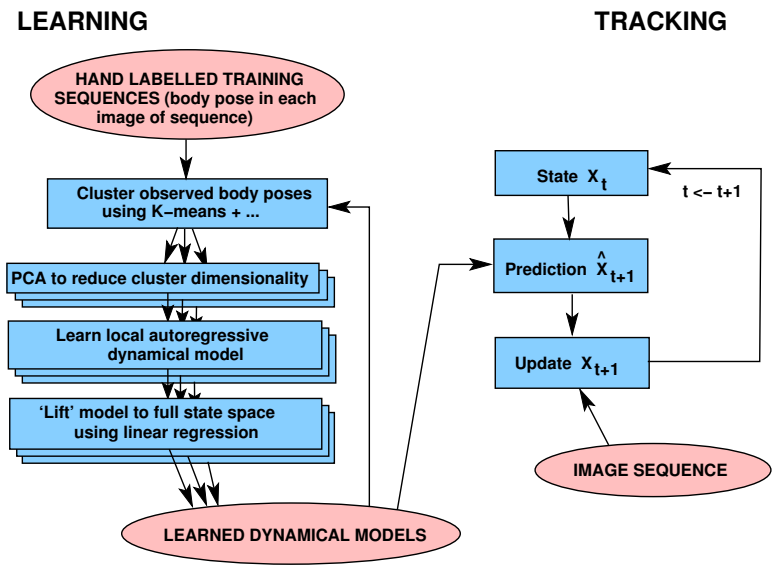


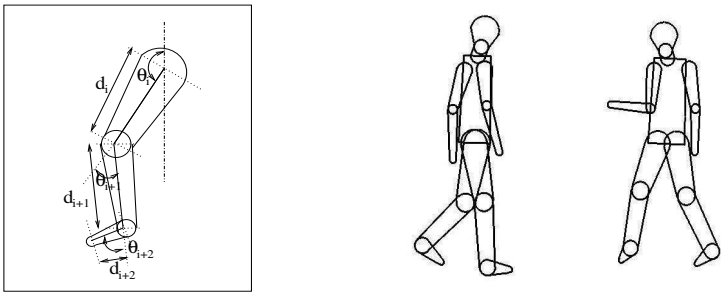
Figure 1: Overview of the learning and tracking components of our algorithm (see text).

have been used [9, 3, 16, 18], as well as non-parametric representations based on motion [4] or appearance [15, 11, 21]. A few learning based methods have modelled dynamics [8, 17, 14], motion patterns from motion capture data (*e.g.* [1]), and image features [16, 7, 6]. To track body pose, Howe *et al* [8] and Sidenbladh *et al* [17] propose plausible next states by recovering similar training examples, while Pavlovic *et al* [14] learn a weak dynamical model over a simplified 8-parameter body for fronto-parallel motions. We extend the learning based approach by modelling complex high dimensional motions within reduced manifolds in an unsupervised setting. In the past, nonlinear motion models have been created by combining Hidden Markov Models and Linear Dynamical Systems in the multi-class dynamics framework, *e.g.* in [13, 14]. However, this approach artificially decouples the switching dynamics from the continuous dynamics. We propose a simpler alternative that avoids this decoupling, discussing our philosophy in section 3.4.

Problem formulation: We use a tracking based approach, representing human motions in terms of a fixed parametric body model controlled by pose-related parameters, and focusing on flexible methods for learning the human dynamics. We specialize to monocular sequences using a 2D (image based) body model, but our methods extend immediately to the 3D and multicamera cases. Our main aim is to study how relationships and constraints in parameter space can be learned automatically from sample trajectories, and how this information can be exploited for tracking. Issues to be handled include the 'curse of dimensionality', complex nonlinear motions, and transitions between different parts of the space.

Overview of approach: Our approach is based on learning dynamical models from sample trajectories. We learn a collection of local motion models (Gaussian autoregressive processes) by automatically partitioning the parameter space into regions with similar dynamical characteristics. The piecewise dynamical model is built from a set of hand-labelled training sequences as follows: *(i)* the state vectors are clustered using K-means and projected to a lower dimensional space using PCA to stabilize the subsequent estimation process; *(ii)* a local linear autoregression for the state given the p previous reduced states is learned for each cluster (p = 1,2 in practice); *(iii)* the data is reclustered using a criterion that takes into account the accuracy of the local model for the given point, as well as the spatial contiguity of points in each model; *(iv)* the models are refitted to the new clusters, and the process is iterated to convergence.

We sidestep the difficult depth estimation problem by using a purely 2D approach, so our dynamical models are view dependent. Our tracking framework is similar to Covariance Scaled Sampling [18]: well-shaped random sampling followed by local optimization of image likelihood. Figure 1 illustrates the basic scheme of dividing the problem into learning and tracking stages.



(a) (b)

Figure 2: *(a)* Human pose parametrization in the Scaled Prismatic Model. *(b)* Examples of different poses of the complete SPM. Each limb segment is overlayed with its corresponding template shape.

2. Body representation

We choose a simple representation for the human body: a modified Scaled Prismatic Model [12] that encodes the body as a set of 2D chains of articulated limb segments. This avoids 3D ambiguities while still capturing the natural degrees of freedom. Body parts are represented by rounded trapezoidal image templates defined by their end widths, and body poses are parametrized by their joint angles and apparent (projected) limb lengths. Including limb lengths, joint angles and hip and shoulder positions, our model contains 33 parameters, giving 33-D state vectors $\mathbf{x} = (\theta_1, d_1, \theta_2, d_2, \ldots \theta_n, d_n)$. Figure 2 illustrates the parametrization and shows some sample poses.

Three additional parameters are used during tracking, two for the image location of the body centre and one for overall scale. We learn scale and translation independently of limb movements, so these parameters are not part of the learned body model. The template for each body part contains texture information used for model-image matching. Its width parameters depend on the subject's clothing and physique. They are defined during initialization and afterwards remain fixed relative to the overall body scale, which is actively tracked.

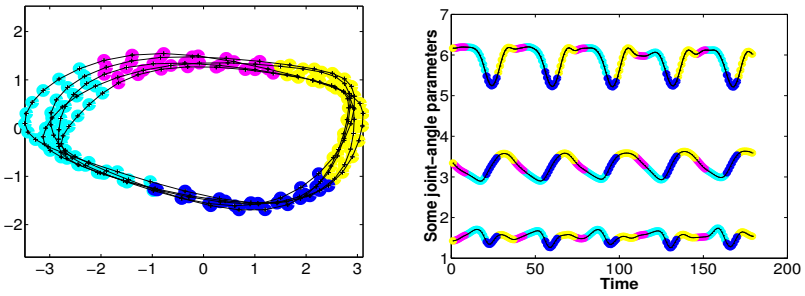
3. Dynamical Model Formulation

Human motion is both complex and time-varying. It is not tractable to build an exact analytical model for it, but approximate models based on statistical methods are a potential substitute. Such models involve learning characteristic motions from example trajectories in parameter space. Our model learns the nonlinear dynamics by partitioning the parameter space into distinct regions or motion classes, and learning a linear autoregressive process covering each region.

3.1 Partitioning of State Space

In cases where the dynamics of a time series changes with time, a single model is often inadequate to describe the evolution in state space. To get around this, we partition the state space into regions containing separate models that describe distinct motion patterns. The partitions must satisfy two main criteria: *(i)* different motion patterns must belong to different regions; and *(ii)* regions should be contiguous in state space. *I.e.*, we need to break the state space into *contiguous regions* with *coherent dynamics*. Coherency means that the chosen dynamical model is locally accurate, contiguity that it can be reliably deduced from the current state space position. Different walking or running styles, viewpoints, *etc.*, tend to use separate regions of state space and hence separate sets of partitions, allowing us to infer pose or action from class information.

We perform an initial partitioning on unstructured input points in state space by using K-means on Mahalanobis distances (see fig. 3). The clusters are found to cut the state trajectories into short sections, all sections in a given partition having similar dynamics. The partition is then refined to improve the accuracies of the nearby dynamical models. The local model estimation and dynamics based partition refinement are iterated in an EM-like loop, details of which are given in section 3.3.



(a) (b)

Figure 3: *(a)* The initial partition of the state space of a walking motion (5 cycles), projected to 2-D using PCA (see text). *(b)* The clusters correspond to different *phases* of the walking cycle, here illustrated using the variations of individual joint angles with time. (The cluster labels are coded by colour). These figures illustrate the optimal clustering obtained for a p=1 ARP. For p=2, a single class suffices for modelling unidirectional walking dynamics.

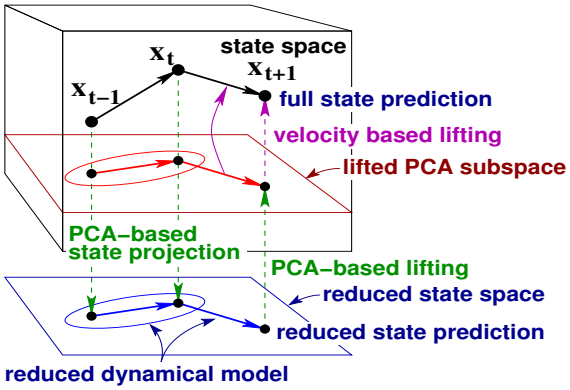


Figure 4: Using a reduced dynamical model to predict states in a high-dimensional space. A given state is projected onto a low-dimensional space using PCA, within which a linear autoregressive progress is used to predict a current (reduced) state. This is then lifted back into full state space to estimate a noise model in the high-dimensional space. To prevent the state from being continually squashed into the PCA subspace, we lift the velocity prediction and not the state prediction.

3.2 Modelling the Local Dynamics

Despite the complexity of human dynamics and the use of unphysical image-based models, we find that the local dynamics within each region is usually well described by a linear Auto-Regressive Process (ARP):

$$\mathbf{x}_t = \sum_{i=1}^{p} \mathbf{A}_i \, \mathbf{x}_{t-i} + \mathbf{w}_t + \mathbf{v}_t \tag{1}$$

Here, $\mathbf{x}_t \in \mathbb{R}^m$ is the pose at time t (joint angles and link lengths), p is the model order (number of previous states used), $\mathbf{A}_i$ are $m \times m$ matrices giving the influence of $\mathbf{x}_{t-i}$ on $\mathbf{x}_t$, $\mathbf{w}_t \in \mathbb{R}^m$ is a drift/offset term, and $\mathbf{v}_t$ is a random noise vector (here assumed white and Gaussian, $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$).

The choice of ARP order is strongly dependent on the nature of the motions exhibited by the system. In practice, experiments on different kinds of motion showed that a second order ARP usually suffices for human tracking:

$$\mathbf{x}_t = \mathbf{A}_1 \, \mathbf{x}_{t-1} + \mathbf{A}_2 \, \mathbf{x}_{t-2} + \mathbf{v}_t \tag{2}$$

This models the local motion as a mass-spring system (set of coupled damped harmonic oscillators). It can also be written in differential form: $\ddot{\mathbf{x}}_t = \mathbf{B}_1 \, \dot{\mathbf{x}}_t + \mathbf{B}_2 \, \mathbf{x}_t + \mathbf{v}_t$.

3.3 Model Parameter Estimation

The parameters to be estimated are the state-space partitioning, here encoded by the class centers $\mathbf{c}^k$, and the ARP parameters $\{\mathbf{A}_1^k, \mathbf{A}_2^k, \ldots \mathbf{A}_p^k, \mathbf{Q}^k\}$ within each class ($k = 1 \ldots K$). There are standard ways of learning ARP models from training data [10]. We computed maximum likelihood parameter estimates. We also wanted to take advantage of the well-structured nature of human motion. People rarely move their limbs completely independently of one another, although the actual degree of correlation depends on the activity being performed. This can be exploited by learning the dynamics with respect to a *reduced set of degrees of freedom* within each class, *i.e.* locally projecting the system trajectories into a lower dimensional subspace. Thus, within each partition, we:

1. reduce the dimensionality using linear PCA (in practice to about 5);
2. learn an ARP model in the reduced space;
3. "lift" this model to the full state space using the PCA injection;
4. cross-validate the resulting model to choose the PCA dimension.

The basic scheme is illustrated in figure 4, and the complete algorithm is given below. Before applying PCA, the state-space dimensions need to be statistically normalized. This is done by dividing each dimension by its observed variance over the complete set of training data.

Algorithm for estimation of maximum-likelihood parameters:

1. Initialize the state-space partitions by K-means clustering based on scaled (diagonal Mahalanobis) distance.
2. Learn an autoregressive model within each partition.
3. Re-partition the input points to minimize the dynamical model prediction error. If the class assignments have converged, stop. Otherwise go to step 2.

Step 2 above is performed as follows:

1. Reduce the vectors in the class to a lower dimensional space by:
   (a) Centering them and assembling them into a matrix (by columns): $\mathbf{X} = [\ (\mathbf{x}_{p_1} - \mathbf{c}) \quad (\mathbf{x}_{p_2} - \mathbf{c}) \quad \cdots \quad (\mathbf{x}_{p_m} - \mathbf{c})\ ]$, where $p_1 \ldots p_m$ are the indices of the points in the class and $\mathbf{c}$ is the class mean.
   (b) Performing a Singular Value Decomposition of the matrix to project out the dominant directions: $\mathbf{X} = \mathbf{U}\,\mathbf{D}\,\mathbf{V}^T$.
   (c) Projecting each vector into the dominant subspace: each $\mathbf{x}_i \in \mathbb{R}^m$ is represented as a reduced vector $\mathbf{q}_i = \tilde{\mathbf{U}}^T(\mathbf{x}_i - \mathbf{c})$ in $\mathbb{R}^{m'}$ ($m' < m$), where $\tilde{\mathbf{U}}$ is the matrix consisting of the first m' columns of $\mathbf{U}$.
2. Build an autoregressive model, $\hat{\mathbf{q}} = \sum_{i=1}^{p} \mathbf{A}_i\, \mathbf{q}_{t-i}$, and estimate $\mathbf{A}_i$ by writing this in the form of a linear regression:
$$\mathbf{q}_t = \tilde{\mathbf{A}}\, \tilde{\mathbf{q}}_{t-1}, \quad t = t_{p_1}, t_{p_2}, \ldots t_{p_n} \tag{3}$$
   where
$$\tilde{\mathbf{A}} = (\ \mathbf{A}_1 \quad \mathbf{A}_2 \quad \cdots \quad \mathbf{A}_p\ ), \quad \tilde{\mathbf{q}}_{t-1} = \begin{pmatrix} \mathbf{q}_{t-1} \\ \mathbf{q}_{t-2} \\ \vdots \\ \mathbf{q}_{t-p} \end{pmatrix}$$
3. Estimate the error covariance $\mathbf{Q}$ from the residual between $\{\hat{\mathbf{x}}_i\}$ and $\{\mathbf{x}_i\}$ by "lifting" $\hat{\mathbf{q}}_t$ back into m dimensions:
$$\hat{\mathbf{x}}_t = \mathbf{c} + \tilde{\mathbf{U}}\hat{\mathbf{q}}_t \tag{4}$$

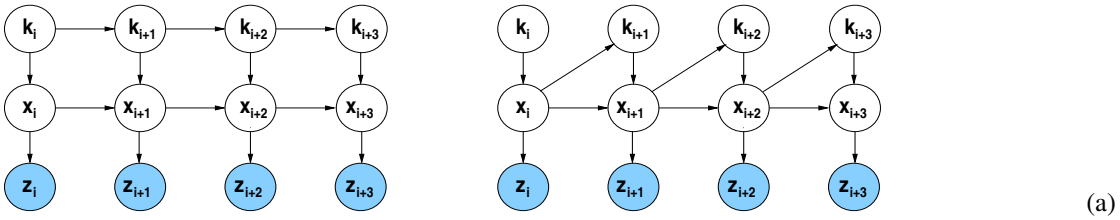


Figure 5: Graphical models for inter-class transitions of a system. *(a)* An HMM-like mixed-state model, and *(b)* our inter-class transition model ($\mathbf{z}_i$: observation, $\mathbf{x}_i$: continuous state, k_i: discrete class). Transitions in an HMM are learned as a fixed transition probability matrix, while our model allows location-sensitive estimation of the class label by exploiting continuous state information.

Step 3 above is performed as follows: The K-means based partitions are revised by assigning training points to the dynamical model that predicts their true motion best, and the dynamical models are then re-learned over their new training points. This EM / relaxation procedure is iterated to convergence. In practice, using dynamical prediction error as the sole fitting criterion gives erratic results, as models sometimes "capture" quite distant points. So we include a spatial smoothing term by minimizing:

$$\sum_{\text{training points}} (\text{prediction error}) + \lambda \cdot (\text{number of inter-class neighbors})$$

where λ is a relative weighting term, and the number of inter-class neighbors is the number of edges in a neighborhood graph that have their two vertices in different classes (*i.e.*, a measure of the lack of contiguity of a partition).

3.4 Inter-Class Transitions

Many example-based trackers use discrete state HMMs (transition probability matrices) to model inter-cluster transitions [21, 20]. This is unavoidable when there is no state space model at all (*e.g.* in exemplars [21]), and it is also effective when modelling time series that are known to be well approximated by a set of piecewise linear regimes [5]. Its use has been extended to multi-class linear dynamical systems exhibiting continuous behavior [14], but we believe that this is unwise, as the discrete transitions ignore the location-within-partition information encoded by the continuous state, which strongly influences inter-class transition probabilities. To work around this, quite small regions have to be used, which breaks up the natural structure of the dynamics and greatly inflates the number of parameters to be learned. In fact, in modelling human motion, the current continuous state already contains a great deal of information about the likely future evolution, and we have found that this alone is rich enough to characterize human motion classes, without the need for the separate hidden discrete state labels of HMM based models.

We thus prefer the simpler approach of using a piecewise linear dynamical model over an explicit spatial partition, where the 'class' label is just the current partition cell. More precisely, we use soft partition assignments obtained from a Gaussian mixture model based at the class centres, so the dynamics for each point is a weighted random mixture over the models of nearby partitions. Our classes cover relatively large regions of state space, but transitions typically only occur at certain (boundary) areas within them. Constant transition probabilities given the current class label would thus be inappropriate in our case.

Figure 5 compares the two schemes in graphical form. By modelling the class-label to be conditional on continuous state, we ensure a smooth flow from one model to the next, avoiding erratic jumps between classes, and we obviate the need for complex inference over a hidden class-label variable.

4. Image Matching Likelihood

At present, for the model-image matching likelihood we simply use the weighted sum-of-squares error of the backwards-warped image against body-part reference templates fixed during initialization. Occlusions are handled using support maps. Each body part P has an associated support map whose j^{th} entry gives the probability that image pixel j currently 'sees' this part. Currently, we use hard assignments, $p(j \text{ sees } P) \in \{0, 1\}$. To resolve the visibility ambiguity when two limbs overlap spatially, each pose has an associated

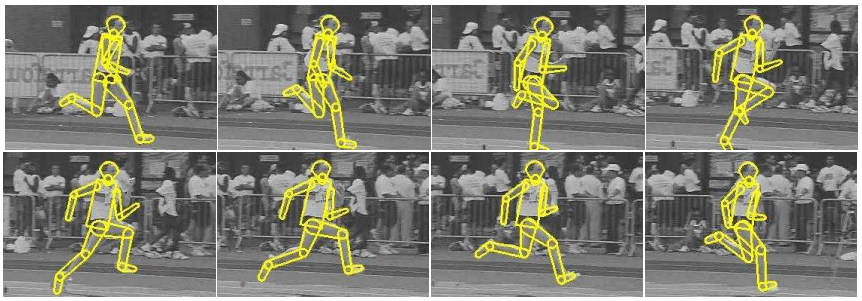


Figure 6: Results from tracking athletic motion (frames 0,4,8,12,16,20,24). The tracker was trained on a different athlete performing a similar motion. Strong priors from the dynamical model allow individual limbs to be tracked in the presence of a confusing background. Note that the left arm is not tracked accurately. This is due to the fact that it was occluded in the initial image and hence no information about its appearance was captured in the template. However, the dynamics continue to give a good estimate of its position.

limb-ordering, which is known a priori for different regions in the pose space from the training data. This information is used to identify occluded pixels that do not contribute to the image matching likelihood for the pose. We charge a fixed penalty for each such pixel, equal to the mean per-pixel error of the visible points in that segment. Some sample support maps are shown in figure 8(b).

5. Tracking Framework

Our tracking framework is similar to Covariance Scaled Sampling [18]. For each mode of $\mathbf{x}_{t-1}$, the distribution $\mathcal{N}(\hat{\mathbf{x}}_t, \mathbf{Q})$ estimated by the dynamical model (1,5) is sampled, and the image likelihood is locally optimized at each mode. State probabilities are propagated over time using Bayes' rule. The probability of the tracker being in state (pose) $\mathbf{x}_t$ at time t given the sequence of observations $\mathcal{Z}_t = \{\mathbf{z}_t, \mathbf{z}_{t-1} \ldots \mathbf{z}_0\}$ is:

$$p(\mathbf{x}_t \,|\, \mathcal{Z}_t) \;=\; p(\mathbf{x}_t \,|\, \mathbf{z}_t, \mathcal{Z}_{t-1}) \;\propto\; p(\mathbf{z}_t \,|\, \mathbf{x}_t)\, p(\mathbf{x}_t \,|\, \mathcal{Z}_{t-1})$$

where $\mathcal{X}_t$ is the sequence of poses $\{\mathbf{x}_i\}$ up to time t and

$$p(\mathbf{x}_t \,|\, \mathcal{Z}_{t-1}) \;=\; \int p(\mathbf{x}_t \,|\, \mathcal{X}_{t-1})\, p(\mathcal{X}_{t-1} \,|\, \mathcal{Z}_{t-1})\, d\mathcal{X}_{t-1} \tag{5}$$

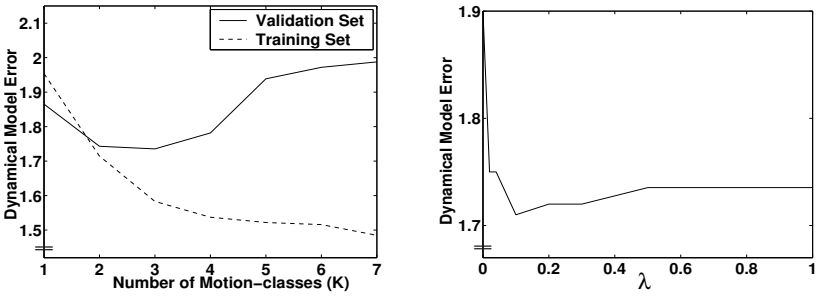
The likelihood $p(\mathbf{z}_t \,|\, \mathbf{x}_t)$ of observing image $\mathbf{z}_t$ given model pose $\mathbf{x}_t$ is computed based on the image-model matching error. The temporal prior $P(\mathbf{x}_t \,|\, \mathcal{X}_{t-1})$ is computed from the learned dynamics. In our piecewise model, the choice of discrete class label k_t is determined by the current region in state space, which in our current implementation depends only on the previous pose $\mathbf{x}_{t-1}$, enabling us to express the probability as

$$p(\mathbf{x}_t \,|\, \mathcal{X}_{t-1}) \;=\; p(\mathbf{x}_t \,|\, \mathcal{X}_{t-1}, k_t)\, p(k_t \,|\, \mathbf{x}_{t-1}) \tag{6}$$

The size and contiguity of our dynamical regions implies that $p(k_t \,|\, \mathbf{x}_{t-1})$ is usually highly unimodal. The number of modes increases when the state lies close to the boundary between two or more regions, but in this case, the spatial coherence inherited from the training dynamics usually ensures that any of the corresponding models can be used successfully, so the number of distinct modes being tracked does not tend to increase exponentially with time. For each model $k = 1 \ldots K$, we use a Gaussian posterior for $p(k|\mathbf{x}_t)$: $p(k \,|\, \mathbf{x}_t) \propto e^{-((\mathbf{x}_t - \mathbf{c}_k)\Sigma^{-1}(\mathbf{x}_t - \mathbf{c}_k))/2}$ where $\mathbf{c}_k$ is the center of the k^{th} class. Note that with a second order ARP model, $p(\mathbf{x}_t \,|\, \mathcal{X}_{t-1}) = p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}, \mathbf{x}_{t-2})$.

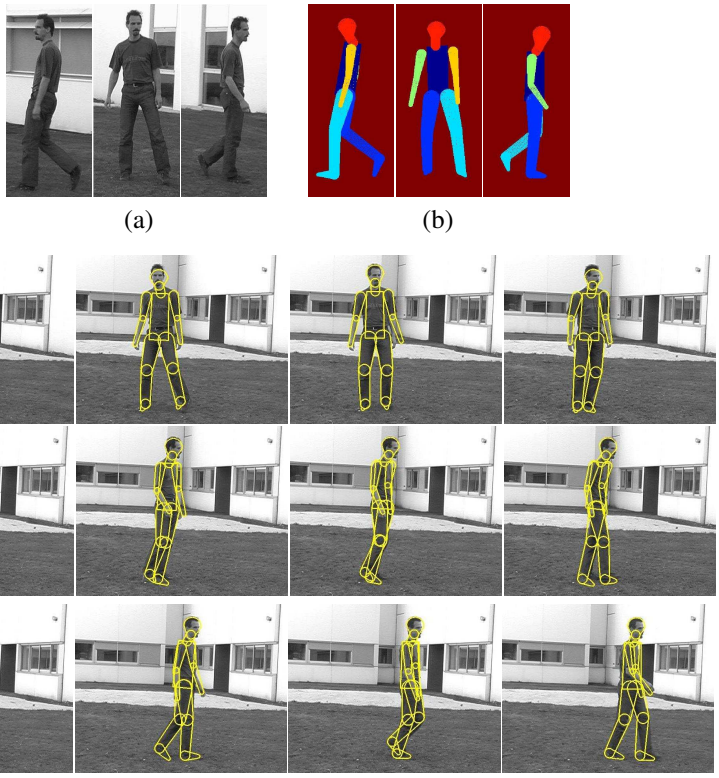
6. Results

We demonstrate our technique by learning models for different classes of human motion and using them to track complete body movements in unseen video sequences. Here, we present results from two challenging sequences.



(a) (b)

Figure 7: *(a)* Dynamical model prediction error w.r.t. number of motion-classes in the turning experiment. Minimizing the validation error selected 3 classes, corresponding to the two walking directions and turning between them. *(b)* The influence of spatial regularization when re-partitioning the state space. A weak regularization $\lambda \sim 0.1$ gives the optimal dynamical estimates. A larger λ causes the partition to remain too close to the suboptimal initial K-means estimate.



(a) (b)

(c)

Figure 8: Examples from our turning experiment. *(a)* Poses characterizing the 3 motion classes learned. *(b)* Support maps illustrating occlusion information for the 3 classes (color coded by body part). *(c)* Tracking results (every 6th frame from 0–66). The corresponding state vectors show a smooth transition between the turning and walking models.

1. Fast athletic motion: This is a case where traditional methods typically fail due to high motion blur. A hand-labelled sequence covering a few running cycles is used to train a model and this is used to track a different person performing a similar motion. For a given viewing direction, we find that a single 2nd order autoregressive process in 5 dimensions suffices to capture the dynamics of such running motions. A tracking example is shown in figure 6.

2. Switching between turning and walking: This experiment illustrates the effectiveness of our inter-class transition model. A 300-frame sequence consisting of walking in different directions and turning motion is used as training data. Our learning algorithm correctly identifies 3 motion patterns (see figure 7(a)), corresponding to two different walking directions and turning between them. The frames corresponding to the centers of these 3 classes are shown in figure 8(a). While tracking a new sequence, the model correctly shifts between different classes enabling smooth switching between activities. Figure 8(c) shows complete tracking results on an unseen test sequence.

In both cases, the models were initialized manually (we are currently working on automatic initialization), after which only the learned dynamics and appearance information were used for tracking. Position and scale changes were modelled respectively as first and zeroth order random walks and learned online during tracking. This allows us to track sequences without assuming either static or fixating cameras, as is done in several other works. The dynamical model alone gives fairly accurate pose predictions for at least a few frames, but the absence of clear observations for any longer than this may cause mistracking.

Figure 7(b) shows how repartitioning (step 3 of our parameter estimation algorithm) improves on the initial K-means based model, provided that a weak smoothing term is included.

7. Conclusion

We have discussed a novel approach to modelling dynamics of high degree-of-freedom systems such as the human body. Our approach is a step towards describing dynamical behavior of high-dimensional parametric model spaces without having to store extremely large amounts of training data. It takes advantage of local correlations between motion parameters by partitioning the space into contiguous regions and learning individual local dynamical behavior within reduced dimensional manifolds. The approach was tested on several different human motion sequences with good results, and allows the tracking of complex unseen motions in the presence of image ambiguities. The piecewise learning scheme developed here is practically effective, and scalable in the sense that it allows models for different actions to be built independently and then stitched together to cover the complete 'activity space'. The learning process can also be made interactive to allow annotation of different classes for activity recognition purposes.

In terms of future work, the appearance model needs to be improved. Adding detectors for characteristic human features and allowing the appearance to evolve with time would help to make the tracker more robust and more general. Including a wider range of training data would allow the tracker to cover more types of human motions.

An open question is whether non-parametric models could usefully be incorporated to aid tracking. Joint angles are a useful output, and are probably also the most appropriate representation for dynamical modelling. But it might be more robust to use comparison with real images, rather than comparison with an idealized model, to compute likelihoods for joint-based pose tracking.

Acknowledgements

This work was supported by the European Union FET-Open Project VIBES.

References

[1] Matthew Brand and Aaron Hertzmann. Style Machines. In *Siggraph 2000, Computer Graphics Proceedings*, pages 183–192, 2000.

[2] C. Bregler and J. Malik. Tracking People with Twists and Exponential Maps. In *International Conference on Computer Vision and Pattern Recognition*, pages 8–15, 1998.

[3] T. Cham and J. Rehg. A Multiple Hypothesis Approach to Figure Tracking. In *International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 239–245, 1999.

[4] A. Efros, A. Berg, G. Mori, and J. Malik. Recognizing Action at a Distance. In *International Conference on Computer Vision*, 2003. To appear.

[5] Z. Ghahramani and G. Hinton. Switching State-Space Models. Technical report, Department of Computer Science, University of Toronto, Canada, 1998.

[6] Tony Heap and David Hogg. Nonlinear Manifold Learning for Visual Speech Recognition. In *International Conference on Computer Vision*, pages 494–499, 1995.

[7] Tony Heap and David Hogg. Wormholes in Shape Space: Tracking Through Discontinuous Changes in Shape. In *International Conference on Computer Vision*, pages 344–349, 1998.

[8] N. Howe, M. Leventon, and W. Freeman. Bayesian Reconstruction of 3D Human Motion from Single-Camera Video. In *Neural Information Processing Systems*, 1999.

[9] S. Ju, M. Black, and Y. Yacoob. Cardboard People: A Parameterized Model of Articulated Motion. In *Int. Conf. on Automatic Face and Gesture Recognition*, pages 38–44, 1996.

[10] H. Lütkepohl. *Introduction to Multiple Time Series Analysis*. Springer-Verlag, Berlin, Germany, second edition, 1993.

[11] G. Mori and J. Malik. Estimating Human Body Configurations Using Shape Context Matching. In *European Conference on Computer Vision*, volume 3, pages 666–680, 2002.

[12] D. Morris and J. Rehg. Singularity Analysis for Articulated Object Tracking. In *International Conference on Computer Vision and Pattern Recognition*, pages 289–296, 1998.

[13] B. North, A. Blake, M. Isard, and J. Rittscher. Learning and Classification of Complex Dynamics. *Pattern Analysis and Machine Intelligence*, 22(9):1016–1034, 2000.

[14] V. Pavlovic, J. Rehg, and J. MacCormick. Learning Switching Linear Models of Human Motion. In *Neural Information Processing Systems*, pages 981–987, 2000.

[15] D. Ramanan and D. Forsyth. Finding and Tracking People from the Bottom Up. In *International Conference on Computer Vision and Pattern Recognition*, 2003.

[16] H. Sidenbladh and M. Black. Learning Image Statistics for Bayesian Tracking. In *International Conference on Computer Vision*, volume 2, pages 709–716, 2001.

[17] H. Sidenbladh, M. Black, and L. Sigal. Implicit Probabilistic Models of Human Motion for Synthesis and Tracking. In *European Conference on Computer Vision*, volume 1, 2002.

[18] C. Sminchisescu and B. Triggs. Covariance Scaled Sampling for Monocular 3D Body Tracking. In *International Conference on Computer Vision and Pattern Recognition*, 2001.

[19] J. Sullivan and S. Carlsson. Recognizing and Tracking Human Action. In *European Conference on Computer Vision*, 2002.

[20] A. Thayananthan, B. Stenger, P. Torr, and R. Cipolla. Learning a kinematic prior for tree-based filtering. In *Proc. British Machine Vision Conference*, volume 2, pages 589–598, 2003.

[21] K. Toyama and A. Blake. Probabilistic Tracking in a Metric Space. In *International Conference on Computer Vision*, pages 50–59, 2001.

Chapter 3

3D Learning Based Motion Capture

This chapter presents a recent paper written with my PhD student Ankur Agarwal. It describes a "model free" approach to human motion capture, based on directly regressing ('learning') a non-linear association between image descriptors and 3D pose vectors. The paper is currently under review for PAMI. It represents an extended journal version of two conference papers, one presented at the 2004 International Conference on Computer Vision and Pattern Recognition (CVPR) [AT04a] representing the static version of the method, and the other presented at the 2004 International Conference on Machine Learning (ICML) [AT04b], representing the dynamic version.

Summary of paper 7, "Recovering 3D Human Pose from Monocular Images"

The paper starts by describing a direct-learning-based method for reconstructing 3D human pose from a single image [AT04a]. The human subject's image silhouette is assumed to have been extracted, *e.g*. by background subtraction. The process begins by extracting a 100-D descriptor vector that robustly encodes the silhouette shape, as follows. Shape context vectors (log-polar histograms of silhouette-edge locations) are calculated for each point on the image silhouette, giving a distribution of points in a 60-D descriptor space. These points are then vector quantized using 100 code vectors learned by k-means clustering over the training data set, and the resulting histogram is normalized to give the silhouette's final 100-D descriptor vector. This is done for all examples in a large training set of 3D pose vectors (joint angles) and their corresponding silhouettes. Finally a nonlinear regressor that predicts 3D pose vectors from 100-D silhouette vectors is learned, and using this, the poses of previously unseen silhouettes can be predicted. Several regression methods were tested. The finally adopted method used a Relevance Vector Machine (a sparse Bayesian method) over a Gaussian kernel, but several other methods gave very similar results. The training data was obtained from a conventional motion capture rig[1], so the final system (only) encodes natural human poses. The silhouette-based image description is intrinsically ambiguous in the sense that there are often several qualitatively different poses that project to a particular image silhouette. In our tests, the method worked well for about 85% of images, but returned incorrect results for the remaining 15%, largely because it chose the wrong solution under this ambiguity.

[1] Unfortunately, the corresponding silhouettes had to be synthesized, as the motion capture system could not supply these.

To correct this ambiguity, we then extend the method to provide 3D human motion capture from monocular image sequences [AT04b]. The main point here is to maintain a purely learning based direct estimation framework, but to use the incoming state estimate to disambiguate between the possible pose-from-silhouette solutions, thus removing the ambiguity of the static image case. State (pose) predictions are made in the obvious way, using a linear autoregressive dynamical model, here learned from the training data. However, rather than following the familiar filtering-based structure where separate pose estimates from the data (silhouette) and the dynamics are merged to make the final posterior state estimate, we use a joint regressor that predicts the state using both the silhouette descriptor and the dynamical prediction. To allow the dynamical prediction to "select" the correct inverse solution for the silhouette-based components to reconstruct (or if you prefer, the appropriate inverse Jacobian for them to use to correct the dynamical prediction), it is essential to include nonlinear interactions between the dynamics-based and the state-based components of this regressor. In our case, joint Gaussian kernels were used for this regression. In testing, the final method worked very well, successfully removing most of the visible 'glitchiness' of the one-image-at-a-time approach and producing realistic-looking resynthesized videos. However it should be noted that it is limited by its training data — poses that lie far from the span of those seen during training can not be reconstructed accurately.

Finally, we briefly study multiple-hypothesis versions of these two methods, based on replacing the individual regressor with a mixture of linear regressors model. In the static case, this simply offers multiple possible solutions, while in the dynamic one, it is integrated into a particle-based probabilistic tracker.

Recovering 3D Human Pose from Monocular Images

Ankur Agarwal and Bill Triggs

Abstract— **We describe a learning based method for recovering 3D human body pose from single images and monocular image sequences. Our approach requires neither an explicit body model nor prior labelling of body parts in the image. Instead, it recovers pose by direct nonlinear regression against shape descriptor vectors extracted automatically from image silhouettes. For robustness against local silhouette segmentation errors, silhouette shape is encoded by histogram-of-shape-contexts descriptors. We evaluate several different regression methods: ridge regression, Relevance Vector Machine (RVM) regression and Support Vector Machine (SVM) regression over both linear and kernel bases. The RVMs provide much sparser regressors without compromising performance, and kernel bases give a small but worthwhile improvement in performance. Loss of depth and limb labelling information often makes the recovery of 3D pose from single silhouettes ambiguous. We propose two solutions to this: the first embeds the method in a tracking framework, using dynamics from the previous state estimate to disambiguate the pose; the second uses a mixture of regressors framework to return multiple solutions for each silhouette. We show that the resulting system tracks long sequences stably, and is also capable of accurately reconstructing 3D human pose from single images, giving multiple possible solutions in ambiguous cases. For realism and good generalization over a wide range of viewpoints, we train the regressors on images resynthesized from real human motion capture data. The method is demonstrated on a 54-parameter full body pose model, both quantitatively on independent but similar test data, and qualitatively on real image sequences. Mean angular errors of 4–5 degrees are obtained — a factor of 3 better than the current state of the art for the much simpler upper body problem.**

Index Terms— **Computer vision, human motion estimation, machine learning, multivariate regression, Relevance Vector Machine**

I. Introduction

We consider the problem of estimating and tracking 3D configurations of complex articulated objects from monocular images, *e.g.* for applications requiring 3D human body pose and hand gesture analysis. There are two main schools of thought on this. *Model-based approaches* presuppose an explicitly known parametric body model, and estimate the pose either by directly inverting the kinematics (which has many possible solutions and requires known image positions for each body part) [28], or by numerically optimizing some form of model-image correspondence metric over the pose variables, using a forward rendering model to predict the images (which is expensive and requires a good initialization, and the problem always has many local minima [25]). An important sub-case is *model-based tracking*, which focuses on tracking the pose estimate from one time step to the next starting from a known initialization, based on an approximate dynamical model [9,23]. In contrast, *learning based approaches* try to avoid the need for explicit initialization and accurate 3D modelling and rendering, and to capitalize on the fact that the set of *typical* human poses is far smaller than the set of kinematically possible ones, by estimating (learning) a model that directly recovers pose estimates from observable image quantities. In particular, *example based methods* explicitly store a set of training examples whose 3D poses are known, and estimate pose by searching for training image(s) similar to the given input image, and interpolating from their poses [5,18,22,27].

In this paper we take a learning based approach, but instead of explicitly storing and searching for similar training examples, we use sparse Bayesian nonlinear regression to distill a large training database into a single compact model that has good generalization to unseen examples. Given the high dimensionality and intrinsic ambiguity of the monocular pose estimation problem, the selection of appropriate image features and good control of overfitting is critical for success. We are not aware of previous work on pose estimation that directly addresses these issues. Our strategy is based on the sparsification and generalization properties of our nonlinear regression algorithm, which is a form of the *Relevance Vector Machine (RVM)* [29]. RVMs have been used earlier, *e.g.* to build kernel regressors for 2D displacement updates in correlation-based patch tracking [33]. Human pose recovery is significantly harder — more ill-conditioned and nonlinear and much higher dimensional — but by selecting a sufficiently rich set of image descriptors, it turns out that we can still obtain enough information for successful regression. Loss of

Manuscript submitted 11 September 2004. The authors are with GRAVIR-CNRS-INRIA, 655 avenue de l'Europe, 38330 Montbonnot, France. *Ankur.Agarwal@inrialpes.fr, Bill.Triggs@inrialpes.fr*, *http://lear.inrialpes.fr/people/{agarwal,triggs}*.

depth and limb labelling information often makes the recovery of 3D pose from single silhouettes ambiguous. We propose two solutions to this. The first embeds the method in a tracking framework, using dynamics from the previous state estimate to disambiguate the pose. The second uses a mixture of regressors framework to return multiple possible solutions for each silhouette, allowing accurate pose reconstructions from single images. When working with a sequence of images, these solutions are fed as input to a multiple hypothesis tracker to give the most likely estimate for each time step.

Previous work: There is a good deal of prior work on human pose analysis, but relatively little on directly learning 3D pose from image measurements. Brand [8] models a dynamical manifold of human body configurations with a Hidden Markov Model and learns using entropy minimization, Athitsos and Sclaroff [4] learn a perceptron mapping between the appearance and parameter spaces, and Shakhnarovich *et al* [22] use an interpolated-k-nearest-neighbor learning method. Human pose is hard to ground truth, so most papers in this area [4,8,18] use only heuristic visual inspection to judge their results. However Shakhnarovich *et al* [22] used a human model rendering package (POSER from Curious Labs) to synthesize ground-truthed training and test images of 13 d.o.f. upper body poses with a limited ($\pm 40^\circ$) set of random torso movements and view points. In comparison, our regression algorithm estimates full body pose and orientation (54 d.o.f.) — a problem whose high dimensionality would really stretch the capacity of an example based method such as [22]. Like [11,22], we used POSER to synthesize a large set of training and test images from different viewpoints, but rather than using random synthetic poses, we used poses taken from real human motion capture sequences. Our results thus relate to real data.

Several publications have used the image locations of the centre of each body joint as an intermediate representation, first estimating these centre locations in the image, then recovering the 3D pose from them. Howe *et al* [12] develop a Bayesian learning framework to recover 3D pose from known centres, based on a training set of pose-centre pairs obtained from resynthesized motion capture data. Mori & Malik [18] estimate the centres using shape context image matching against a set of training images with pre-labelled centres, then reconstruct 3D pose using the algorithm of [28]. These approaches show that using 2D joint centres as an intermediate representation can be an effective strategy, but we have preferred to estimate pose directly from the underlying local image descriptors as we feel that this is likely to prove both more accurate and more robust, also providing a generic framework for directly estimating and tracking any prespecified set of parameters from image observations.

As regards tracking, some approaches have learned dynamical models for specific human motions [19,20]. Particle filters and MCMC methods have been widely used in probabilistic tracking frameworks *e.g.* [23,31]. Most of these methods use an explicit generative model to compute observation likelihoods. We propose a discriminatively motivated framework in which dynamical state predictions are directly fused with descriptors computed from the observed image. Our algorithm is related to Bayesian tracking, but we eliminate the need for both an explicit body model that is projected to predict image observations, and a corresponding error model that is used to evaluate the likelihood of the observed image given this projection. A brief description of our regression based scheme is given is [1] and its first extension to resolve ambiguities using dynamics within the regression is described in [2].

Overview of the approach: We represent 3D body pose by 55-D vectors $\mathbf{x}$ including 3 joint angles for each of the 18 major body joints. This choice corresponds to the motion capture data that we use to train the system (details in section II-B). The input images are reduced to 100-D observation vectors $\mathbf{z}$ that robustly encode the shape of a human image silhouette. Given a set of labelled training examples $\{(\mathbf{z}_i, \mathbf{x}_i) \,|\, i = 1 \ldots n\}$, the RVM learns a smooth reconstruction function $\mathbf{x} = \mathbf{r}(\mathbf{z})$, valid over the region spanned by the training points. The function is a weighted linear combination $\mathbf{r}(\mathbf{z}) \equiv \sum_k \mathbf{a}_k \, \phi_k(\mathbf{z})$ of a prespecifed set of scalar basis functions $\{\phi_k(\mathbf{z}) \,|\, k = 1 \ldots p\}$. In our tracking framework, to help to disambiguate pose in cases where there are several possible reconstructions, the functional form is extended to include an approximate preliminary pose estimate $\check{\mathbf{x}}$, $\mathbf{x} = \mathbf{r}(\check{\mathbf{x}}, \mathbf{z})$. (See section V.) At each time step, a state estimate $\check{\mathbf{x}}_t$ is obtained from the previous two pose vectors using an autoregressive dynamical model, and this is used to compute the basis functions, which now take the form $\{\phi_k(\check{\mathbf{x}}, \mathbf{z}) \,|\, k = 1 \ldots p\}$. Section VI gives an alternative method for handling ambiguities by returning multiple possible 3D configurations corresponding to a silhouette. The functional form is extended to a probabilistic mixture $p(\mathbf{x}) \sim \sum_k \pi_k \delta(\mathbf{x}, \mathbf{r}_k)$ allowing each reconstruction $\mathbf{r}_k$ to output a different solution.

Our solutions are well-regularized in the sense that the weight vectors $\mathbf{a}_k$ are damped to control over-fitting, and sparse in the sense that many of them are zero. Sparsity occurs because the RVM actively selects only the 'most relevant' basis functions — the ones that really need to

Fig. 1. Different 3D poses can have very similar image observations, causing the regression from image silhouettes to 3D pose to be inherently multi-valued. The legs are the arms are reversed in the first two images, for example.

have nonzero coefficients to complete the regression successfully. A sparse solution obtained by the RVM allows the system to select relevant input *features* (components) in case of a linear basis ($\phi_k(\mathbf{z}) = k^{th}$ component of $\mathbf{z}$). For a kernel basis — $\phi_k(\mathbf{z}) \equiv K(\mathbf{z}, \mathbf{z}_k)$ for some kernel function $K(\mathbf{z}_i, \mathbf{z}_j)$ and centres $\mathbf{z}_k$ — relevant training examples are selected, allowing us to prune a large training dataset and retain only a minimal subset.

Organization: §II describes our image descriptors and body pose representation. §III gives an outline of our regression methods. §IV details the recovery of 3D pose from single images using this regression, discussing the RVM's feature selection properties but showing that ambiguities in estimating 3D pose from single images cause occasional 'glitches' in the results. §V describes our first solution to this problem: a tracking based regression framework capable of resolving these ambiguities, with results from our novel tracker in §V-B. §VI describes an alternative solution: a mixture of regressors based approach incorporated in a multiple hypothesis tracker. Finally, §VII concludes with some discussions and directions for future work.

II. Representing Images and Body Poses

Directly regressing pose on input images requires a robust, compact and well-behaved representation of the observed image information and a suitable parametrization of the body poses that we wish to recover. To encode the observed images we use robust descriptors of the shape of the subject's image silhouette, and to describe our body pose, we use vectors of joint angles.

A. Images as Shape Descriptors

Silhouettes: Of the many different image descriptors that could be used for human pose estimation, and in line with [4,8], we have chosen to base our system on image silhouettes.

Silhouettes have three main advantages: *(i)* They can be extracted moderately reliably from images, at least when robust background- or motion-based segmentation is available and problems with shadows are avoided; *(ii)* they are insensitive to irrelevant surface attributes like clothing colour and texture; *(iii)* they encode a great deal of useful information about 3D pose without the need of any labelling information [1].

Two factors limit the performance attainable from silhouettes: *(i)* Artifacts such as shadow attachment and poor background segmentation tend to distort their local form. This often causes problems when global descriptors such as shape moments are used (as in [4,8]), as every local error pollutes each component of the descriptor: to be robust, shape descriptors need to have good *locality*. *(ii)* Silhouettes make several discrete and continuous degrees of freedom invisible or poorly visible (see fig. 1). It is difficult to tell frontal views from back ones, whether a person seen from the side is stepping with the left leg or the right one, and what are the exact poses of arms or hands that fall within (are "occluded" by) the torso's silhouette. Including interior edge information within the silhouette [22] is likely to provide a useful degree of disambiguation in such cases, but is difficult to disambiguate from, *e.g.* markings on clothing.

Shape Context Distributions: To improve resistance to segmentation errors and occlusions, we need a robust silhouette representation. The first requirement for robustness is *locality*. Histogramming edge information is a good way to encode local shape robustly [17,6], so we begin by computing local descriptors at regularly spaced points on the edge of the silhouette. We use shape contexts (histograms of local edge pixels into log-polar bins [6]) to encode silhouette shape quasi-locally over a range of scales, computing the contexts in local regions defined by diameter roughly equal to the size of a limb. In our application we assume that the vertical is preserved, so to improve discrimination, we do not normalize contexts with respect to their dominant local orientations as originally proposed in [6]. The silhouette shape is thus encoded as a distribution

[1] We do not believe that any representation (Fourier coefficients, *etc.*) based on treating the silhouette shape as a continuous parametrized curve is appropriate for this application: silhouettes frequently change topology (*e.g.* when a hand's silhouette touches the torso's one), so parametric curve-based encodings necessarily have discontinuities w.r.t. shape.

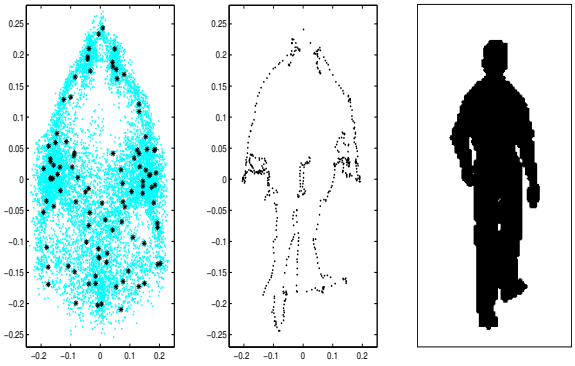


Fig. 2. (Left) The first two principal components of the distribution of all shape context vectors from a training data sequence, with the k-means centres superimposed. The average-over-human-silhouettes like form arises because (besides finer distinctions) the context vectors encode approximate spatial position on the silhouette: a context at the bottom left of the silhouette receives votes only in its upper right bins, *etc*. (Centre) The same projection for the edge-points of a single silhouette (shown on the right).

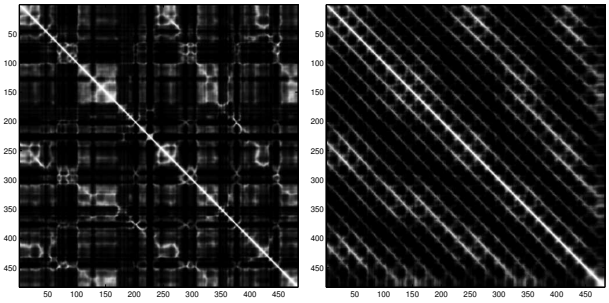


Fig. 3. Pairwise similarity matrices for (left) image silhouette descriptors and (right) true 3D poses, for a 483-frame sequence of a person walking in a decreasing spiral. The light off-diagonal bands that are visible in both matrices denote regions of comparative similarity linking corresponding poses on different cycles of the spiral. This indicates that our silhouette descriptors do indeed capture a significant amount of pose information. (The light SW-NE ripples in the 3D pose matrix just indicate that the standing-like poses at the middle of each stride have mid-range joint values, and hence are closer on average to other poses than the 'stepping' ones at the end of strides).

(in fact, as a noisy multibranched curve, but we treat it as a distribution) in the 60-D shape context space. (In our implementation, shape contexts contain 12 angular $\times$ 5 radial bins, giving rise to 60 dimensional histograms.) Matching silhouettes is therefore reduced to matching these distributions in shape context space. To implement this, a second level of histogramming is performed: we reduce the distributions of all points on each silhouette to 100-D histograms by vector quantizing the shape context space. Silhouette comparison is thus finally reduced to a comparison of 100-D histograms. The 100 centre codebook is learned once and for all by running k-means on the combined set of context vectors of all of the training silhouettes. See fig. 2. (Other centre selection methods give similar results.) For a given silhouette, a 100-D histogram $\mathbf{z}$ is built by allowing each of its context vectors to vote softly into the few centre-classes nearest to it, and accumulating scores of all context vectors. This *soft* voting reduces the effects of spatial quantization, allowing us to compare histograms using simple Euclidean distance, rather than, say, Earth Movers Distance [21]. (We have also tested the normalized cellwise distance $\|\sqrt{\mathbf{p}_1}-\sqrt{\mathbf{p}_2}\|^2$, with very similar results.) The histogram-of-shape-contexts scheme gives us a reasonable degree of robustness to occlusions and local silhouette segmentation failures, and indeed captures a significant amount of pose information (see fig. 3).

B. Body Pose as Joint Angles

We recover 3D body pose (including orientation w.r.t. the camera) as a real 55-D vector $\mathbf{x}$, including 3 joint angles for each of the 18 major body joints. The subject's overall azimuth (compass heading angle) θ can wrap around through 360°. To maintain continuity, we actually regress $(a, b) = (\cos\theta, \sin\theta)$ rather than θ, using $\mathrm{atan2}(b, a)$ to recover θ from the not-necessarily-normalized vector returned by regression. So we have $3\times18+1 = 55$ parameters.

We stress that our framework is inherently 'model-free' and is independent of the choice of this pose representation. The system itself has no explicit body model or rendering model, and no knowledge of the 'meaning' of the motion capture parameters that it is regressing — it simply learns to predict these from silhouette data. Similarly, we have not sought to learn a minimal representation of the true human pose degrees of freedom, but simply to regress the original motion capture based training format, and our regression methods handle such redundant output representations without problems.

The motion capture data was taken from the public website www.ict.usc.edu/graphics/animWeb/ humanoid. Although we use real motion capture data for joint angles, we do not have access to the corresponding image silhouettes, so we currently use a graphics package, POSER from Curious Labs, to synthesize suitable training images, and also to visualize the final reconstruction. This does unfortunately involve the use of a synthetic body model, but we stress that this model is not part of our system and would not be needed if real motion capture data with silhouettes were available.

III. Regression Methods

This section describes the regression methods that we have evaluated for recovering 3D human body pose from the above image descriptors. Here we follow standard regression notation, representing the output pose by real vectors $\mathbf{y} \in \mathbb{R}^m$ and the input shape as vectors $\mathbf{x} \in \mathbb{R}^d$. [2]

For most of the paper, we assume that the relationship between $\mathbf{x}$ and $\mathbf{y}$ — which a priori, given the ambiguities of pose recovery, might be multi-valued and hence relational rather than functional — can be approximated functionally as a linear combination as a prespecified set of basis functions:

$$\mathbf{y} \;=\; \sum_{k=1}^{p} \mathbf{a}_k \, \phi_k(\mathbf{x}) + \boldsymbol{\epsilon} \;\equiv\; \mathbf{A}\,\mathbf{f}(\mathbf{x}) + \boldsymbol{\epsilon} \qquad (1)$$

Here, $\{\phi_k(\mathbf{x}) \,|\, k = 1 \ldots p\}$ are the basis functions, $\mathbf{a}_k$ are $\mathbb{R}^m$-valued weight vectors, and $\boldsymbol{\epsilon}$ is a residual error vector. For compactness, we gather the weight vectors into an $m \times p$ weight matrix $\mathbf{A} \equiv (\mathbf{a}_1 \;\; \mathbf{a}_2 \;\; \cdots \;\; \mathbf{a}_p)$ and the basis functions into a $\mathbb{R}^p$-valued function $\mathbf{f}(\mathbf{x}) = (\phi_1(\mathbf{x}) \;\; \phi_2(\mathbf{x}) \;\; \cdots \;\; \phi_p(\mathbf{x}))^\top$. To allow for a constant offset $\mathbf{A}\mathbf{f}+\mathbf{b}$, we can include $\phi(\mathbf{x}) \equiv 1$ in $\mathbf{f}$.

To train the model (estimate $\mathbf{A}$), we are given a set of training pairs $\{(\mathbf{y}_i, \mathbf{x}_i) \,|\, i = 1 \ldots n\}$. In this paper we will usually use the Euclidean norm to measure $\mathbf{y}$-space prediction errors, so the estimation problem is of the form:

$$\mathbf{A} \;:=\; \arg\min_{\mathbf{A}} \left\{ \sum_{i=1}^{n} \|\mathbf{A}\,\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i\|^2 \;+\; R(\mathbf{A}) \right\} \qquad (2)$$

where $R(-)$ is a regularizer on $\mathbf{A}$. Gathering the training points into an $m \times n$ output matrix $\mathbf{Y} \equiv (\mathbf{y}_1 \;\; \mathbf{y}_2 \;\; \cdots \;\; \mathbf{y}_n)$ and a $p \times n$ feature matrix $\mathbf{F} \equiv (\mathbf{f}(\mathbf{x}_1) \;\; \mathbf{f}(\mathbf{x}_2) \;\; \cdots \;\; \mathbf{f}(\mathbf{x}_n))$, the estimation problem takes the form:

$$\mathbf{A} \;:=\; \arg\min_{\mathbf{A}} \left\{ \; \|\mathbf{A}\,\mathbf{F} - \mathbf{Y}\|^2 + R(\mathbf{A}) \; \right\} \qquad (3)$$

Note that the dependence on $\{\phi_k(-)\}$ and $\{\mathbf{x}_i\}$ is encoded entirely in the numerical matrix $\mathbf{F}$.

A. Ridge Regression

Pose estimation is a high dimensional and intrinsically ill-conditioned problem, so simple least squares estimation — setting $R(\mathbf{A}) \equiv \mathbf{0}$ and solving for $\mathbf{A}$ in least squares — typically produces severe overfitting and hence poor generalization. To reduce this, we need to add a smoothness constraint on the learned mapping, for example by including a damping or regularization term $R(\mathbf{A})$ that penalizes large values in the coefficient matrix $\mathbf{A}$. Consider the simplest choice, $R(\mathbf{A}) \equiv \lambda \|\mathbf{A}\|^2$, where λ is a regularization parameter. This gives the *ridge* regressor, or *damped least squares* regressor, which minimizes

$$\|\mathbf{A}\,\tilde{\mathbf{F}} - \tilde{\mathbf{Y}}\|^2 \;:=\; \|\mathbf{A}\,\mathbf{F} - \mathbf{Y}\|^2 + \lambda \|\mathbf{A}\|^2 \qquad (4)$$

where $\tilde{\mathbf{F}} \equiv (\mathbf{F} \;\; \lambda\mathbf{I})$ and $\tilde{\mathbf{Y}} \equiv (\mathbf{Y} \;\; \mathbf{0})$. The solution can be obtained by solving the linear system $\mathbf{A}\,\tilde{\mathbf{F}} = \tilde{\mathbf{Y}}$ (*i.e.* $\tilde{\mathbf{F}}^\top \mathbf{A}^\top = \tilde{\mathbf{Y}}^\top$) for $\mathbf{A}$ in least squares[3], using QR decomposition or the normal equations. Ridge solutions are not equivariant under scaling of inputs, so we usually standardize the inputs (*i.e.* scale them to have unit variance) before solving.

λ must be set large enough to control ill-conditioning and overfitting, but not so large as to cause overdamping (forcing $\mathbf{A}$ towards $\mathbf{0}$ so that the regressor systematically underestimates the solution).

RVM Training Algorithm

1) Initialize $\mathbf{A}$ with ridge regression. Initialize the running scale estimates $a_{\text{scale}} = \|\mathbf{a}\|$ for the components or vectors $\mathbf{a}$.
2) Approximate the $\nu \log \|\mathbf{a}\|$ penalty terms with "quadratic bridges", the gradients of which match at a_{scale}. *I.e.* the penalty terms take the form $\frac{\nu}{2}\,(\mathbf{a}/a_{\text{scale}})^2 + \text{const}$.
(One can set const $= \nu(\log\|a_{\text{scale}}\| - \frac{1}{2})$ to match the function values at a_{scale}, but this value is irrelevant for the least squares minimization.)
3) Solve the resulting linear least squares problem in $\mathbf{A}$.
4) Remove any components $\mathbf{a}$ that have become zero, update the scale estimates $a_{\text{scale}} = \|\mathbf{a}\|$, and continue from 2 until convergence.

Fig. 4. An outline of our RVM training algorithm.

B. Relevance Vector Regression

Relevance Vector Machines (RVMs) [29,30] are a sparse Bayesian approach to classification and regression. They introduce Gaussian priors on each parameter or group of parameters, each prior being controlled by its own individual scale hyperparameter. Integrating out the hyperpriors (which can be done analytically)

[2] However note that in subsequent sections, outputs (3D-pose vectors) will be denoted by $\mathbf{x} \in \mathbb{R}^{55}$ and inputs will be instances from either the observation space, $\mathbf{z} \in \mathbb{R}^{100}$, or the joint (predicted) state + observation space, $(\mathbf{x}^\top, \mathbf{z}^\top)^\top \in \mathbb{R}^{155}$.

[3] In case a constant offset $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$ is included, this vector $\mathbf{b}$ must not be *damped* and hence the system takes the form $(\mathbf{A} \;\; \mathbf{b})\,\tilde{\mathbf{F}} = \tilde{\mathbf{Y}}$ where $\tilde{\mathbf{F}} \equiv \begin{pmatrix} \mathbf{F} & \lambda\mathbf{I} \\ \mathbf{1} & \mathbf{0} \end{pmatrix}$ and $\tilde{\mathbf{Y}} \equiv (\mathbf{Y} \quad \mathbf{0})$.

gives singular, highly nonconvex total priors of the form $p(a) \sim \|a\|^{-\nu}$ for each parameter or parameter group a, where ν is a hyperprior parameter. Taking log likelihoods gives an equivalent regularization penalty of the form $R(a) = \nu \log \|a\|$. Note the effect of this penalty. If $\|a\|$ is large, the 'regularizing force' $dR/da \sim \nu/\|a\|$ is small so the prior has little effect on a. But the smaller $\|a\|$ becomes, the greater the regularizing force becomes. At a certain point, the data term no longer suffices to hold the parameter at a nonzero value against this force, and the parameter rapidly converges to zero. Hence, the fitted model is sparse — the RVM automatically selects a subset of 'relevant' basis functions that suffices to describe the problem. The regularizing effect is invariant to rescalings of $\mathbf{f}()$ or $\mathbf{Y}$. (*E.g.* scaling $\mathbf{f} \to \alpha\mathbf{f}$ forces a rescaling $\mathbf{A} \to \mathbf{A}/\alpha$ with no change in residual error, so the regularization forces $1/\|a\| \propto \alpha$ track the data-term gradient $\mathbf{A}\,\mathbf{F}\,\mathbf{F}^\top \propto \alpha$ correctly). ν serves both as a sparsity parameter and as a scale-free regularization parameter. The complete RVM model is highly non-convex with many local minima and optimizing it can be problematic because relevant parameters can easily become accidentally 'trapped' in the singularity at zero. However, in practice this does not prevent RVMs from giving useful results. Setting ν to optimize the estimation error on a validation set, one typically finds that RVMs give sparse regressors with performance very similar to the much denser ones from analogous methods with milder priors.

To train our RVMs, we do not use Tipping's algorithm [29], but rather a continuation method based on successively approximating the $\nu \log \|a\|$ regularizers with quadratic "bridges" $\nu\,(\|a\|/a_{\text{scale}})^2$ chosen to match the prior gradient at a_{scale}, a running scale estimate for a (see fig. 5). The bridging changes the apparent curvature if the cost surfaces, allowing parameters to pass through zero if they need to, with less risk of premature trapping. The algorithm is sketched in figure 4.

We have tested both *componentwise* priors, $R(\mathbf{A}) = \nu \sum_{jk} \log |\mathbf{A}_{jk}|$, which effectively allow a different set of relevant basis functions to be selected for each dimension of $\mathbf{y}$, and *columnwise* ones, $R(\mathbf{A}) = \nu \sum_k \log \|\mathbf{a}_k\|$ where $\mathbf{a}_k$ is the k^{th} column of $\mathbf{A}$, which select a common set of relevant basis functions for all components of $\mathbf{y}$. Both priors give similar results, but one of the main advantages of sparsity is in reducing the number of basis functions (support features or examples) that need to be evaluated, so in the experiments shown we use columnwise priors. Hence, we minimize

$$\|\mathbf{A}\,\mathbf{F} - \mathbf{Y}\|^2 + \nu \sum_k \log \|\mathbf{a}_k\| \qquad (5)$$

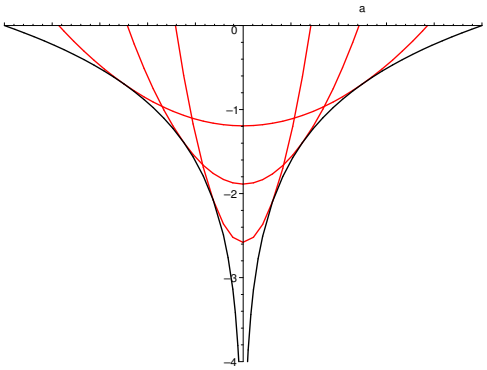


Fig. 5. "Quadratic bridge" approximations to the $\nu \log \|a\|$ regularizers. These are introduced to prevent parameters from prematurely becoming *trapped* at zero. (See text.)

C. Choice of Basis

We tested two kinds of regression bases $\mathbf{f}(\mathbf{x})$. *(i) Linear bases*, $\mathbf{f}(\mathbf{x}) \equiv \mathbf{x}$, simply return the input vector, so the regressor is linear in $\mathbf{x}$ and the RVM selects relevant *features* (components of $\mathbf{x}$). *(ii) Kernel bases*, $\mathbf{f}(\mathbf{x}) = (K(\mathbf{x}, \mathbf{x}_1) \;\cdots\; K(\mathbf{x}, \mathbf{x}_n))^\top$, are based on a kernel function $K(\mathbf{x}, \mathbf{x}_i)$ instantiated at training examples $\mathbf{x}_i$, so the RVM effectively selects relevant *examples*. Our experiments with various kernels and combinations of kernels and linear functions show that kernelization (of our already highly non linear features) gives a small but useful improvement in performance — about $0.8°$ per body angle, out of a total mean error of around $7°$. The form and parameters of the kernel have remarkably little influence. The experiments shown use a Gaussian kernel $K(\mathbf{x}, \mathbf{x}_i) = e^{-\beta\|\mathbf{x}-\mathbf{x}_i\|^2}$ with β estimated from the scatter matrix of the training data, but other β values within a factor of 2 from this value give very similar results.

IV. Pose from Static Images

We conducted experiments using a database of motion capture data for a 54 d.o.f. body model (3 angles for each of 18 joints, including body orientation w.r.t the camera). We report mean (over all 54 angles) RMS absolute difference errors between the true and estimated joint angle vectors, in degrees:

$$D(\mathbf{x}, \mathbf{x}') = \frac{1}{m} \sum_{i=1}^{m} |(x_i - x'_i) \bmod \pm 180°| \qquad (6)$$

The training silhouettes were created by using Poser to render the motion captured poses, and reduced to 100-D histograms by vector quantizing their shape context distributions using centres selected by k-means.

We compare here results of regressing body pose $\mathbf{x}$ (after transforming from 54-D to 55-D as described

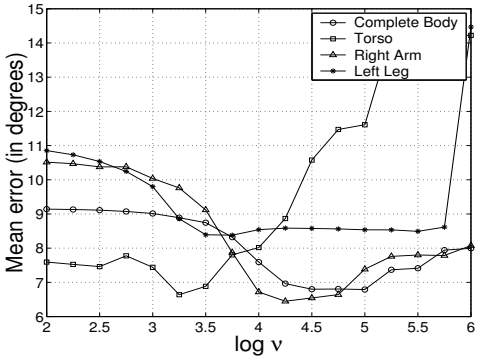


Fig. 6. Mean test-set fitting error for different combinations of body parts, versus the linear RVM spareseness parameter ν. The minima indicate the optimal sparsity / regularization settings for each body part. Limb regressors are sparser than body or torso ones: the whole body regressor retains 23 features; torso, 31; right arm, 10; and the left leg, 7.

(a) (b) (c) (d) (e) (f)

Fig. 7. Silhouette points whose shape context classes are retained by the RVM for regression on (a) left arm angles, (b) right leg angles, shown on a sample silhouette. (c-f): Silhouette points encoding torso & neck parameter values over different view points and poses. On average, about 10 features covering about 10% of the silhouette suffice to estimate the pose of each body part.

in section II-B) on the silhouette descriptors $\mathbf{z}$ using ridge, RVM and SVM [32] based regression methods on linear and kernel bases with the functional form given in section III:

$$\mathbf{x} \;=\; \mathbf{A}\,\mathbf{f}(\mathbf{z}) + \boldsymbol{\epsilon} \;\equiv\; \sum_{k=1}^{p} \mathbf{a}_k\,\phi_k(\mathbf{z}) + \boldsymbol{\epsilon} \qquad (7)$$

Ridge regression and RVM regression use quadratic loss functions to measure $\mathbf{x}$-space prediction errors, as described in section III, while SVM regression uses the ϵ-insensitive loss function [26] and a linear programming method for training. The results shown here use the SVM-*light* [15] for implementation.

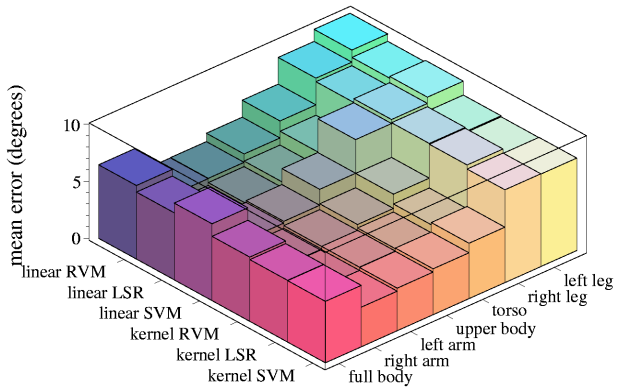
A. Implicit Feature Selection

Kernel based RVM regression gives reliable pose estimates while retaining only about 6% of the training examples, but working in kernel space hides information associated with individual input features (components of $\mathbf{z}$-vectors). Conversely, linear-basis RVM regression ($\mathbf{f}(\mathbf{z}) = \mathbf{z}$) provides less flexible modelling of the relationship between $\mathbf{x}$ and $\mathbf{z}$, but reveals which of the original input features encode useful pose information, as the RVM directly selects relevant components of $\mathbf{z}$.

One might expect that, *e.g.* the pose of the arms was mainly encoded by (shape-context classes receiving contributions from) features on the arms, and so forth, so that the arms could be regressed from fewer features than the whole body, and could be regressed robustly even if the legs were occluded. To test this, we divided the body joints into five subsets — torso & neck, the two arms, and the two legs — and trained separate linear RVM regressors for each subset. Fig. 6 shows that similar validation-set errors are attained for each part, but the optimal regularization level is significantly smaller (there is less sparsity) for the torso than for the other parts. Fig. 7 shows the silhouette points whose contexts contribute to the features (histogram classes) that were selected as relevant, for several parts and poses. The two main observations are that the regressors are indeed sparse — only about 10 of the 100 histogram bins were classed as relevant on average, and the points contributing to these tend to be well localized in important-looking regions of the silhouette — but that there is a good deal of non-locality between the points selected for making observations and the parts of the body being estimated. This nonlocality is somewhat surprising. It is perhaps only due to the extent to which the motions of different body segments are synchronized during natural walking motion, but if it turns out to be true for larger training sets containing less orchestrated motions, it may suggest that the localized calculations of model-based pose recovery actually miss a good deal of the information most relevant for pose.

B. Performance Analysis

Fig. 8 summarizes the test-set performance of the various regression methods studied — kernelized and linear basis versions of damped least squares regression (LSR), RVM and SVM regression, for the full body model and various subsets of it — at optimal regularizer settings computed using 2-fold cross validation. All output parameters are normalized to have unit variance before regression and the tube width ϵ in the SVM is set to correspond to an error of 1° for each joint angle. Kernelization brings only a small advantage (0.8° on an average) over purely linear regression against our (highly nonlinear) descriptor set. The regressors are all found to give their best results at similar optimal kernel parameters, which are more or less independent of the regularization prior strengths. The RVM regression gives



	LSR	RVM	SVM
Average error (in degrees)	5.95	6.02	5.91
% of support vectors retained	100	6	53

Fig. 8. (*Top*) A summary of our various regressors' performance on different combinations of body parts for the spiral walking test sequence. (*Bottom*) Error measures for the full body using Gaussian kernel bases with the corresponding number of support vectors retained.

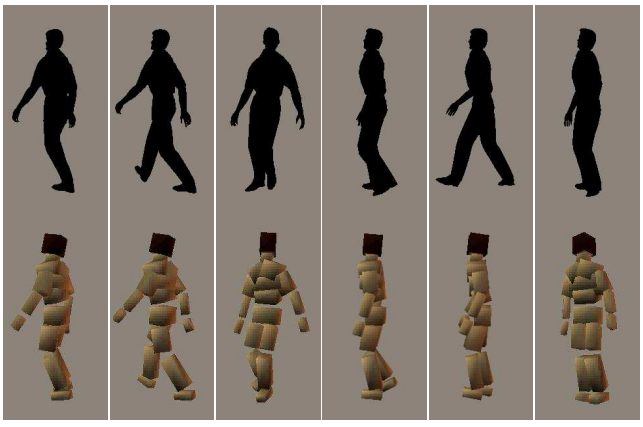


Fig. 9. Some sample pose reconstructions for a spiral walking sequence not included in the training data. The reconstructions were computed with a Gaussian kernel RVM, using only 156 of the 2636 training examples. The mean angular error per d.o.f. over the whole sequence is $6.0°$. While (a-c) show accurate reconstructions, (d-f) are examples of misestimation: (d) illustrates a label confusion (the left and right legs have been interchanged), (e,f) are examples of compromised solutions where the regressor has averaged between two or more distinct possibilities. Using single images alone, we find $\sim 15\%$ of our results are misestimated.

very slightly higher errors than the other two regressors, but much more sparsity. For example, in our whole-body method, the final RVM selects just 156 (about 6%) of the 2636 training points as basis kernels, to give a mean test-set error of $6.0°$. We attribute the slightly better performance of the SVM to the different form of its loss function. The overall similarity of the results obtained from the 3 different regressors confirms that our representation and framework are insensitive to the exact method of regression used.

Fig. 9 shows some sample pose estimation results, on silhouettes from a spiral-walking motion capture sequence that was not included in the training set. The mean estimation error over all joints for the Gaussian RVM in this test is $6.0°$, but the error for individual joints varies depending on the range and discernibility of each joint angle. The RMS errors obtained for some key body angles are as follows (the ranges of variation of these angles in the test set are given in parentheses): body heading angle: $17°$ ($360°$), left shoulder angle: $7.5°$ ($50.8°$), and right hip angle: $4.2°$ ($47.4°$). Fig. 10 (top) plots the estimated and actual values of the overall body heading angle θ during the test sequence, showing that much of the error is due to occasional large errors that we will refer to as "glitches". These are associated with ambiguous cases where the silhouette might easily arise from any of several possible poses. As one diagnostic for this, recall that to allow for the $360°$ wrap around of the heading angle θ, we actually regress $(a, b) = (\cos\theta, \sin\theta)$ rather than θ. In ambiguous cases, the regressor tends to compromise between several possible solutions, and hence returns an (a, b) vector whose norm is significantly less than one. These events are strongly correlated with large estimation errors in θ, as illustrated in fig. 10.

Fig. 11 shows reconstruction results on some real images. The reconstruction quality demonstrates the method's robustness to imperfect visual features, as a quite naive background subtraction method was used to extract somewhat imperfect body silhouettes from these images. The last example demonstrates the problem of silhouette ambiguity: the method returns a pose with the left knee bent instead of the right one as the silhouette looks the same in the two cases, causing a glitch in the output pose.

Although numerically our results are already significantly better than others presented in the literature ($6°$ as compared to RMS errors of about $20°$ per d.o.f. reported in [22]), our pose reconstructions do still contain a significant amount of temporal jitter, and also occasional glitches. The jitter is to be expected given that each image is processed independently. It can be reduced by temporal filtering (simple smoothing or Kalman filtering), and also by adding a temporal dimension to the regressor. The glitches occur when more than one solution is possible, causing the regressor to either 'select' the wrong solution, or to output a compromised solution, different from each. One possible way to reduce such errors would be to incorporate stronger features such as internal body edges within the silhouette, however the

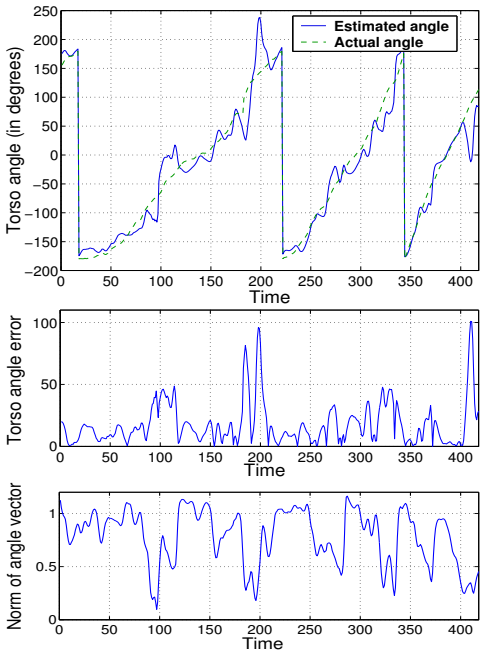


Fig. 10. (Top): The estimated body heading (azimuth θ) over 418 frames of the spiral walking test sequence, compared with its actual value from motion capture. (Middle, Bottom): Episodes of high estimation error are strongly correlated with periods when the norm of the $(\cos\theta, \sin\theta)$ vector that was regressed to estimate θ becomes small. These occur when similar silhouettes arise from very different poses, so that the regressor is forced into outputing a compromise solution.

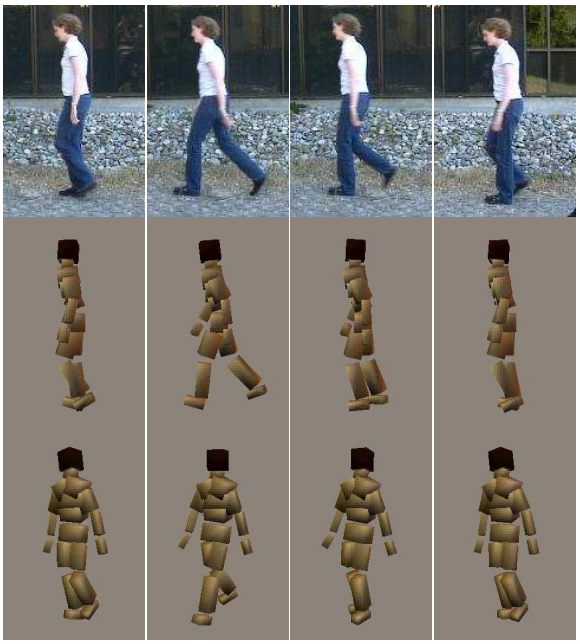


Fig. 11. 3D poses reconstructed from some real test images using a single image for each reconstruction (the images are part of a sequence from www.nada.kth.se/∼hedvig/data.html). The middle and lower rows respectively show the estimates from the original viewpoint and from a new one. The first two columns show accurate reconstructions. In the third column, a noisy silhouette causes slight misestimation of the lower right leg, while the final column demonstrates a case of left-right ambiguity in the silhouette.

problem is bound to persist as important internal body edges are often not visible and useful body edges have to be distinguished from irrelevant clothing texture edges. Furthermore, even without these limb labelling ambiguities, depth related ambiguities continue to remain an issue. By relying on experimentally observed poses, our single image method has already reduced this ambiguity significantly, but human beings often rely on very subtle cues to disambiguate multiple solutions.

In the absence of multiple simultaneous views, temporal continuity is an important supplementary source of information for resolving these ambiguities. In the following two sections, we describe two different approaches that exploit continuity within our regression model.

V. Tracking and Regression

This section describes a novel 'discriminative' tracking framework that fuses pose predictions from a learned dynamical model into our single image regression framework, to correctly reconstruct the most likely 3D pose at each time step. The 3D pose can only be observed indirectly via ambiguous and noisy image measurements, so it is appropriate to start by considering the Bayesian tracking framework in which our knowledge about the state (pose) $\mathbf{x}_t$ given the observations up to time t is represented by a probability distribution, the posterior state density $p(\mathbf{x}_t \mid \mathbf{z}_t, \mathbf{z}_{t-1}, \ldots, \mathbf{z}_0)$.

Given an image observation $\mathbf{z}_t$ and a prior $p(\mathbf{x}_t)$ on the corresponding pose $\mathbf{x}_t$, the posterior likelihood for $\mathbf{x}_t$ is usually evaluated using Bayes' rule, $p(\mathbf{x}_t|\mathbf{z}_t) \propto p(\mathbf{z}_t|\mathbf{x}_t)\, p(\mathbf{x}_t)$, where $p(\mathbf{z}_t|\mathbf{x}_t)$ is an explicit 'generative' observation model that predicts $\mathbf{z}_t$ and its uncertainty given $\mathbf{x}_t$. Unfortunately, when tracking objects as complicated as the human body, the observations depend on a great many factors that are difficult to control, ranging from lighting and background to body shape and clothing style and texture, so any hand-built observation model is necessarily a gross oversimplification. One way around this would be to learn the generative model $p(\mathbf{z}|\mathbf{x})$ from examples, then to work backwards via its Jacobian to get a linearized state update, as in the extended Kalman filter. However, this approach is somewhat indirect, and it may waste a considerable amount of effort modelling appearance details that are irrelevant for predicting pose. Instead, we prefer to learn a 'diagnostic' (discriminative or regressive) model $p(\mathbf{x}|\mathbf{z})$ for the pose

$\mathbf{x}$ given the observations $\mathbf{z}$ — *c.f.* the difference between generative and discriminative classifiers, and the regression based trackers of [16,33]. Similarly, in the context of maximum likelihood pose estimation, we prefer to learn a diagnostic regressor $\mathbf{x} = \mathbf{x}(\mathbf{z})$, *i.e.* a point estimator for the most likely state $\mathbf{x}$ given the observations $\mathbf{z}$, not a generative predictor $\mathbf{z} = \mathbf{z}(\mathbf{x})$. Unfortunately, this brings up a second problem. As we have seen in the previous section, image projection suppresses most of the depth (camera-object distance) information and using silhouettes as image observations induces further ambiguities owing to the lack of limb labelling. So the state-to-observation mapping is always many-to-one. These ambiguities make learning to regress $\mathbf{x}$ from $\mathbf{z}$ difficult because the true mapping is actually multi-valued. A single-valued least squares regressor tends to either zig-zag erratically between different training poses, or (if highly damped) to reproduce their arithmetic mean [7], neither of which is desirable.

To reduce the ambiguity, we work incrementally from the previous few states[4] $\mathbf{x}_{t-1}, \ldots$ (*e.g.* as was done in [10]). We adopt the working hypothesis that given a dynamics based estimate $\mathbf{x}_t(\mathbf{x}_{t-1}, \ldots)$ — or any other rough initial estimate $\check{\mathbf{x}}_t$ for $\mathbf{x}_t$ — it will usually be the case that only one of the observation-based estimates is at all likely a posteriori. Thus, we can use the $\check{\mathbf{x}}_t$ value to "select the correct solution" for the observation-based reconstruction $\mathbf{x}_t(\mathbf{z}_t)$. Formally this gives a regressor $\mathbf{x}_t = \mathbf{x}_t(\mathbf{z}_t, \check{\mathbf{x}}_t)$, where $\check{\mathbf{x}}_t$ serves mainly as a key to select which branch of the pose-from-observation space to use, not as a useful prediction of $\mathbf{x}_t$ in its own right. To work like this, the regressor must be well-localized in $\check{\mathbf{x}}_t$, and hence nonlinear. Taking this one step further, if $\check{\mathbf{x}}_t$ is actually a useful estimate of $\mathbf{x}_t$ (*e.g.* from a dynamical model), we can use a single regressor of the same form, $\mathbf{x}_t = \mathbf{x}_t(\mathbf{z}_t, \check{\mathbf{x}}_t)$, but now with a stronger dependence on $\check{\mathbf{x}}_t$, to capture the net effect of implicitly reconstructing an observation-estimate $\mathbf{x}_t(\mathbf{z}_t)$ and then fusing it with $\check{\mathbf{x}}_t$ to get a better estimate of $\mathbf{x}_t$.

A. Learning the Regression Models

Our discriminative tracking framework now has two levels of regression. We formulate the models as follows and continue to use the methods described in section III:

1) Dynamical (Prediction) Model: Human body dynamics can be modelled fairly accurately with a second order linear autoregressive process, $\mathbf{x}_t = \check{\mathbf{x}}_t + \boldsymbol{\epsilon}$, where $\check{\mathbf{x}}_t \equiv \tilde{\mathbf{A}}\,\mathbf{x}_{t-1} + \tilde{\mathbf{B}}\,\mathbf{x}_{t-2}$ is the second order dynamical estimate of $\mathbf{x}_t$ and $\boldsymbol{\epsilon}$ is a residual error vector (*c.f.* *e.g.* [3]). To ensure dynamical stability and avoid overfitting, we actually learn the autoregression for $\check{\mathbf{x}}_t$ in the following form:

$$\check{\mathbf{x}}_t \;\equiv\; (\mathbf{I} + \mathbf{A})(2\mathbf{x}_{t-1} - \mathbf{x}_{t-2}) + \mathbf{B}\,\mathbf{x}_{t-1} \qquad (8)$$

where $\mathbf{I}$ is the $m \times m$ identity matrix. This form helps to maintain stability by converging towards a default linear prediction if $\mathbf{A}$ and $\mathbf{B}$ are overdamped. We estimate $\mathbf{A}$ and $\mathbf{B}$ by regularized least squares regression against $\mathbf{x}_t$, minimizing $\|\boldsymbol{\epsilon}\|_2^2 + \lambda(\|\mathbf{A}\|_{\text{Frob}}^2 + \|\mathbf{B}\|_{\text{Frob}}^2)$ over the training set, with the regularization parameter λ set by cross-validation to give a well-damped solution with good generalization.

2) Likelihood (Correction) Model: Now consider the observation model. As discussed above, the underlying density $p(\mathbf{x}_t \,|\, \mathbf{z}_t)$ is highly multimodal owing to the pervasive ambiguities in reconstructing 3D pose from monocular images, so no single-valued regression function $\mathbf{x}_t = \mathbf{x}_t(\mathbf{z}_t)$ can give acceptable point estimates for $\mathbf{x}_t$. However much of the 'glitchiness' and jitter observed in the static reconstructions of section IV-B can be removed by feeding $\check{\mathbf{x}}_t$ into the regression model. The combined regressor can be formulated in several different ways. The simplest is to linearly combine $\check{\mathbf{x}}_t$ with the estimate $\mathbf{x}_t$ given by equation (7), but this only smooths the results, reducing jitter, while still continuing to give wrong solutions when (7) returns a wrong estimate. We thus include a non-linear dependence on $\check{\mathbf{x}}_t$ with $\mathbf{z}_t$ in the observation-based regressor, giving a state sensitive observation update. Our full regression model also includes an explicit linear $\check{\mathbf{x}}_t$ term to represent the direct contribution of the dynamics to the overall state estimate, so the final model becomes $\mathbf{x}_t \equiv \hat{\mathbf{x}}_t + \boldsymbol{\epsilon}'$ where $\boldsymbol{\epsilon}'$ is a residual error to be minimized, and:

$$\hat{\mathbf{x}}_t = \mathbf{C}\,\check{\mathbf{x}}_t + \sum_{k=1}^{p} \mathbf{d}_k\,\phi_k(\check{\mathbf{x}}_t, \mathbf{z}_t) \equiv \begin{pmatrix}\mathbf{C} & \mathbf{D}\end{pmatrix} \begin{pmatrix}\check{\mathbf{x}}_t \\ \mathbf{f}(\check{\mathbf{x}}_t, \mathbf{z}_t)\end{pmatrix} \qquad (9)$$

Here, $\{\phi_k(\mathbf{x}, \mathbf{z}) \,|\, k = 1 \ldots p\}$ is a set of scalar-valued nonlinear basis functions for the regression, and $\mathbf{d}_k$ are the corresponding $\mathbb{R}^m$-valued weight vectors. For compactness, we gather these into an $\mathbb{R}^p$-valued feature vector $\mathbf{f}(\mathbf{x}, \mathbf{z}) \equiv (\phi_1(\mathbf{x}, \mathbf{z}), \ldots, \phi_p(\mathbf{x}, \mathbf{z}))^\top$ and an $m \times p$ weight matrix $\mathbf{D} \equiv (\mathbf{d}_1, \ldots, \mathbf{d}_p)$. In the experiments reported here, we used instantiated-kernel bases of the form

$$\phi_k(\mathbf{x}, \mathbf{z}) = K_x(\mathbf{x}, \mathbf{x}_k) \cdot K_z(\mathbf{z}, \mathbf{z}_k) \qquad (10)$$

where $(\mathbf{x}_k, \mathbf{z}_k)$ is a training example and K_x, K_z are (here, independent Gaussian) kernels on $\mathbf{x}$-space and $\mathbf{z}$-space, $K_x(\mathbf{x}, \mathbf{x}_k) = e^{-\beta_x \|\mathbf{x} - \mathbf{x}_k\|^2}$ and $K_z(\mathbf{z}, \mathbf{z}_k) = e^{-\beta_z \|\mathbf{z} - \mathbf{z}_k\|^2}$. Building the basis from Gaussians based at training examples in joint $(\mathbf{x}, \mathbf{z})$ space makes examples

[4] As an alternative we tried regressing the pose $\mathbf{x}_t$ against a sequence of the last few silhouettes $(\mathbf{z}_t, \mathbf{z}_{t-1}, \ldots)$, but the ambiguities are found to persist for several frames.

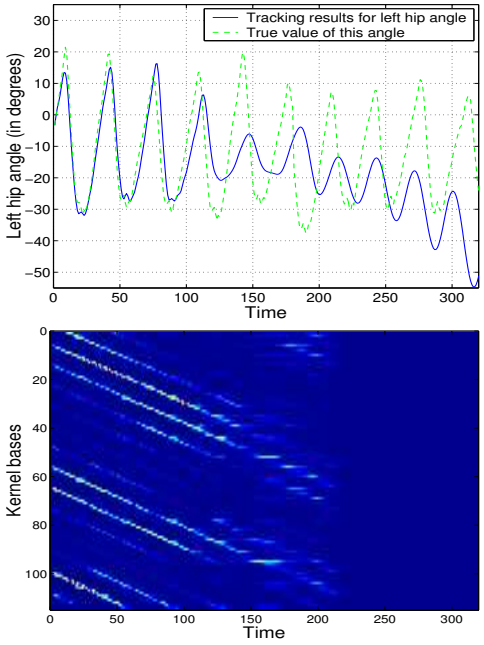


Fig. 12. An example of mistracking caused by an over-narrow pose kernel K_x. The kernel width is set to 1/10 of the optimal value, causing the tracker to lose track from about t=120, after which the state estimate drifts away from the training region and all kernels stop firing by about t=200. *Left:* the variation of a left hip angle parameter for a test sequence of a person walking in a spiral. *Right:* The temporal activity of the 120 kernels (training examples) during this track. The banded pattern occurs because the kernels are samples taken from along a similar 2.5 cycle spiral walking sequence, each circuit involving about 8 steps. The similarity between adjacent steps and between different circuits is clearly visible, showing that the regressor can locally still generalize well.

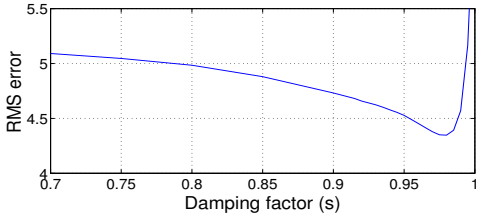


Fig. 13. The variation of the RMS test-set tracking error with damping factor s. See the text for discussion.

relevant only if they have similar image silhouettes *and* similar underlying poses.

Mistracking due to extinction. Kernelization in joint $(\mathbf{x}, \mathbf{z})$ space allows the relevant branch of the inverse solution to be chosen, but it is essential to choose the relative widths of the kernels appropriately. If the $\mathbf{x}$-kernel is chosen too wide, the method tends to average over (or zig-zag between) several alternative pose-from-observation solutions, which defeats the purpose of including $\check{\mathbf{x}}$ in the observation regression. On the other hand, too much locality in $\mathbf{x}$ effectively 'switches off' the observation-based state corrections whenever the estimated state happens to wander too far from the observed training examples $\mathbf{x}_k$. So if the $\mathbf{x}$-kernel is set too narrow, observation information is only incorporated sporadically and mistracking can easily occur. Fig. 12 illustrates this effect, for an $\mathbf{x}$-kernel a factor of 10 narrower than the optimum. The method initially seemed to be sensitive to the kernel width parameters, but after fixing good default values by cross-validation on an independent motion sequence we observed accurate performance over a sufficiently wide range for both the kernel widths: a tolerance factor of about 2 on β_x and about 4 on β_z.

Neutral vs Damped Dynamics. The coefficient matrix $\mathbf{C}$ in (9) plays an interesting role. Setting $\mathbf{C} \equiv \mathbf{I}$ forces the correction model to act as a differential update on $\check{\mathbf{x}}_t$ (what we refer to as having a 'neutral' dynamical model). On the other extreme, $\mathbf{C} \equiv \mathbf{0}$ gives largely observation-based state estimates with only a latent dependence on the dynamics. An intermediate setting, however, turns out to give the best overall results. Damping the dynamics slightly ensures stability and controls drift — in particular, preventing the observations from disastrously 'switching off' because the state has drifted too far from the training examples — while still allowing a reasonable amount of dynamical smoothing. Usually we estimate the full (regularized) matrix $\mathbf{C}$ from the training data, but to get an idea of the trade-offs involved, we also studied the effect of explicitly setting $\mathbf{C} = s\mathbf{I}$ for $s \in [0, 1]$. We find that a small amount of damping, $s_{opt} \approx .98$ gives the best results overall, maintaining a good lock on the observations without losing too much dynamical smoothing (see fig. 13.) This simple heuristic setting gives very similar results to the model obtained by learning the full matrix $\mathbf{C}$.

B. Tracking Results

We trained the new regression model (9) on our motion capture data as in section IV. For these experiments, we used 8 different sequences totalling about 2000 instantaneous poses for training, and another two sequences of about 400 points each as validation and test sets. Errors are again reported as described by (6).

The dynamical model is learned from the training data exactly as described in §V-A.1, but when training the observation model, we find that its coverage and capture radius can be increased by including a wider selection of $\check{\mathbf{x}}_t$ values than those produced by the dynamical predictions. Hence, we train the model $\mathbf{x} = \mathbf{x}_t(\check{\mathbf{x}}, \mathbf{z})$ using a combination of 'observed' samples $(\check{\mathbf{x}}_t, \mathbf{z}_t)$ (with $\check{\mathbf{x}}_t$ computed from (8)) and artificial samples generated

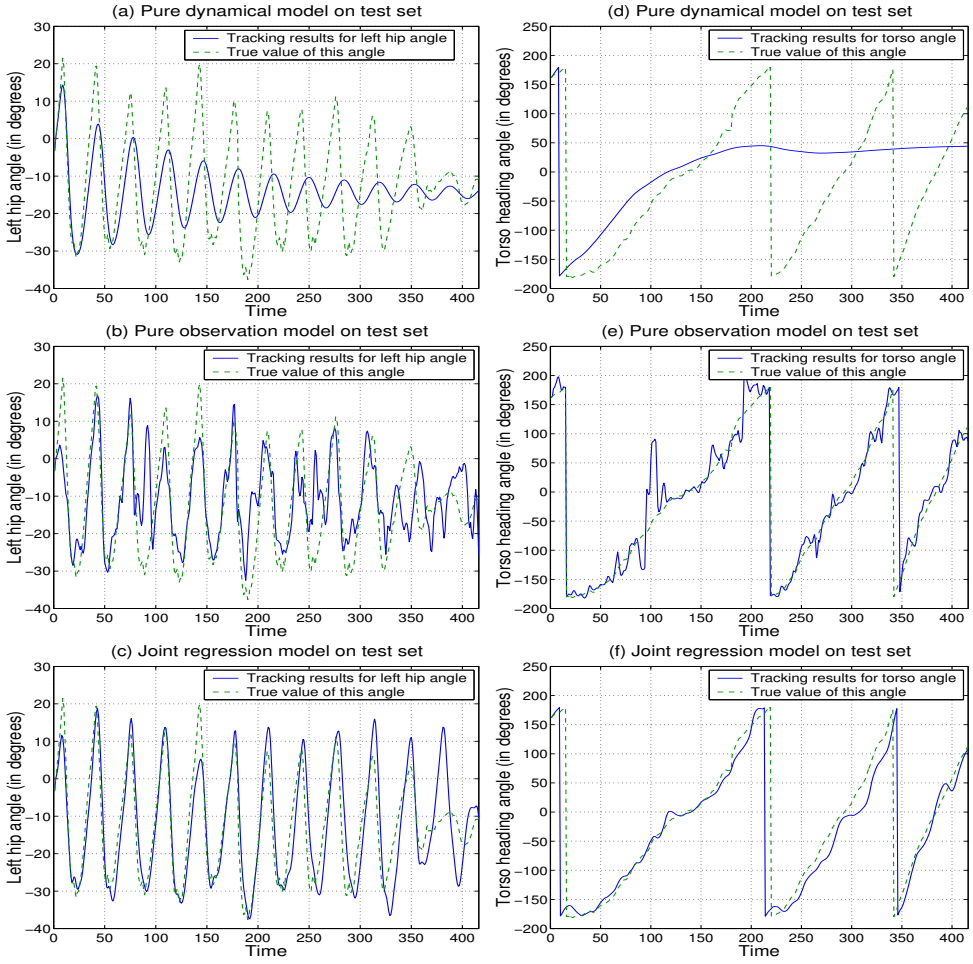


Fig. 14. Sample tracking results on a spiral walking test sequence. (a) Variation of the left hip-angle parameter, as predicted by a pure dynamical model initialized at $t = \{0, 1\}$, (b) Estimated values of this angle from regression on observations alone (*i.e.* no initialization or temporal information), (c) Results from our novel joint regressor, obtained by combining dynamical and state+observation based regression models. (d,e,f) Similar plots for the overall body rotation angle. Note that this angle wraps around at 360°, i.e. $\theta \simeq \theta \pm 360^\circ$.

by Gaussian sampling $\mathcal{N}(\mathbf{x}_t, \Sigma)$ around the training state $\mathbf{x}_t$. The observation $\mathbf{z}_t$ corresponding to $\mathbf{x}_t$ is still used, forcing the observation based part of the regressor to rely mainly on the observations, *i.e.* on recovering $\mathbf{x}_t$ (or at least an update to $\check{\mathbf{x}}_t$) from $\mathbf{z}_t$, using $\check{\mathbf{x}}_t$ mainly as a hint about the inverse solution to choose. The covariance matrix Σ is chosen to reflect the local scatter of the training examples, with a larger variance along the tangent to the trajectory at each point to ensure that phase lag between the state estimate and the true state is reliably detected and corrected.

Fig. 14 illustrates the relative contributions of the dynamics and observation terms in our model by plotting tracking results for a motion capture test sequence in which the subject walks in a decreasing spiral. This sequence was not included in the training set, although similar ones were. The purely dynamical model (8) provides good estimates for a few time steps, but gradually damps and drifts out of phase. Such damped oscillations are characteristic of second order linear autoregressive dynamics, trained with enough regularization to ensure model stability. The results based on observations alone without any temporal information are included again here for comparison. These are obtained from (7), which is actually a special case of (9) where $\mathbf{C} = \mathbf{0}$ and $K_x = 1$. Panels (c),(f) show that jointly regressing dynamics and observations gives a significant improvement in estimation quality, with smoother and stabler tracking. There is still some residual misestimation of the hip angle in (c) at around t=140 and t=380. At these points, the subject is walking directly towards the camera (heading angle $\theta \sim 0^\circ$), so the only cue for hip

Fig. 15. Sample pose reconstructions for the spiral walking sequence using the tracking method. This sequence was not included in the training data, and corresponds to figures 14(c) & (f). The reconstructions were computed with a Gaussian kernel RVM, using only 18% training examples. The average RMS estimation error per d.o.f. over the whole sequence is 4.1°.

angle is the position of the corresponding foot, which is sometimes occluded by the opposite leg. Humans also find it difficult to estimate this angle from the silhouette at these points.

Fig. 15 shows some silhouettes and corresponding maximum likelihood pose reconstructions, for the same test sequence. The 3D poses for the first two time steps were set by hand to initialize the dynamical predictions. The average RMS estimation error over all joints using the RVM regressor in this test is 4.1°. Well-regularized least squares regression over the same basis gives similar errors, but has much higher storage requirements. The Gaussian RVM gives a sparse regressor for (9) involving only 348 of the 1927 (18%) training examples, thus allowing a significant reduction in the amount of training data that needs to be stored. The reconstruction results on two test video sequences are shown in figs 16 and 19.

In terms of computation time, the final RVM regressor already runs in real time in Matlab. Silhouette extraction and shape-context descriptor computations are currently done offline, but should be feasible online in real time. The offline learning process takes about 2-3 min for the RVM with $\sim$2000 data points, and currently about 20 min for Shape Context extraction and clustering (this being highly unoptimized Matlab code).

Automatic Initialization: The method is reasonably robust to initialization errors. Although the results shown in figs. 14 and 15 were obtained by initializing from ground truth, we also tested the effects of automatic (and hence potentially incorrect) initialization. In an experiment in which the tracker was automatically initialized at each time step in turn using the pure observation model, then tracked forwards and backwards using the dynamical tracker, the initialization lead to successful tracking in 84% of the cases. The failures were the 'glitches', where the observation model gave completely incorrect initializations.

VI. Resolving Ambiguities using a Mixture of Experts

In this section, we discuss an alternative approach to dealing with multiple possible solutions in the 3D pose estimation problem. We extend our single image regression framework from section IV to a *mixture* of regressors (often known as a mixture of experts [14]). Such a model enables the regressor to output more than one possible solution from a single silhouette — in general a multimodal probability density $p(\mathbf{x}|\mathbf{z})$. We describe the formulation of our mixture model and show how it can be used in a multiple hypothesis probabilistic tracking framework to achieve smooth reconstruction tracks free from glitches.

A. Probabilistic pose from static images

A close analysis of the nature of ambiguities in the silhouette-to-pose problem indicates that they are of more than one type in nature. Firstly, there exist instances where any 3D pose in a continuous range seems to explain the given silhouette observation quite well, *e.g.* estimating out-of-plane rotations where the limb length signal is not strong enough to estimate the angle accurately. Here one would desire a broad distribution in 3D pose space as the output from a single silhouette. Other cases of ambiguity arise due to kinematic flipping (*c.f.* [24]) or label-ambiguities (disambiguating the left and right arms/legs). In such cases, there is typically a finite discrete set of probable solutions — often only 2 or 4, but sometimes more. To deal with both of the above cases, we model the conditional density $p(\mathbf{x}|\mathbf{z})$ as a mixture of Gaussians:

$$p(\mathbf{x}|\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\bar{\mathbf{x}}_k, \mathbf{\Lambda}_k) \tag{11}$$

where $\bar{\mathbf{x}}_k$ is computed by learning a regressor $\bar{\mathbf{x}}_k = \mathbf{A}_k\ \mathbf{f}(\mathbf{z}) + \mathbf{b}_k$ within each mixture component, and $\mathbf{\Lambda_k}$ (a diagonal covariance matrix in our case) is estimated from residual errors. π_k are the gating probabilities of the regressors. Setting $\mathbf{f}(\mathbf{z}) \equiv \mathbf{z}$ simplifies the problem to learning a mixture of *linear* regressors. The model is learned by fitting a mixture of Gaussians to the joint probability density $(\mathbf{z}^\top, \mathbf{x}^\top)^\top$:

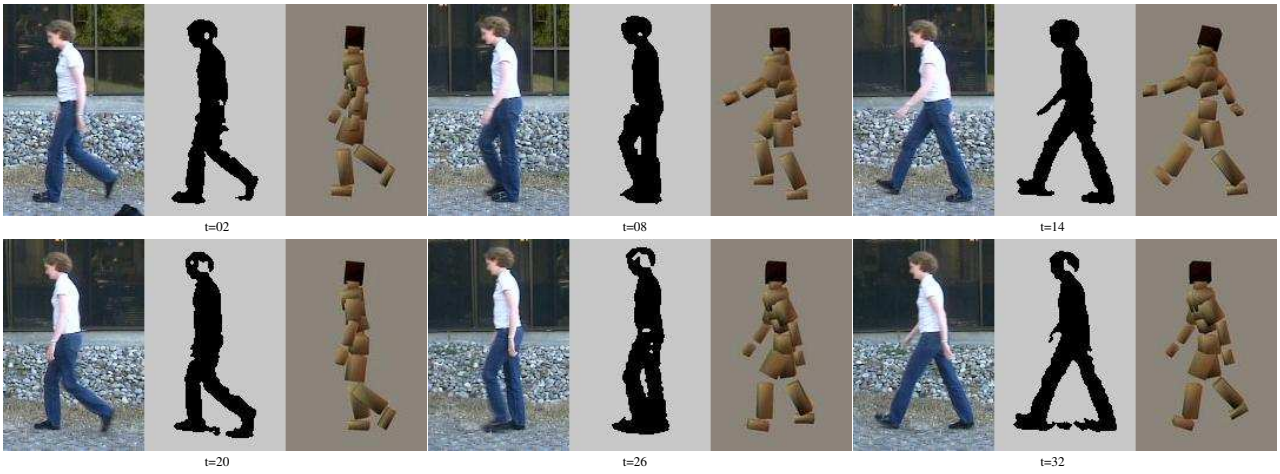


Fig. 16. 3D poses reconstructed from a test video sequence (obtained from www.nada.kth.se/∼hedvig/data.html). The presence of shadows and holes in the extracted silhouettes demonstrates the robustness of our shape descriptors — however, a weak or noisy observation signal sometimes causes failure to track accurately. *E.g.* at $t = 8, 14$, the pose estimates are dominated by the dynamical predictions, which do ensure smooth and natural motion but may cause slight mistracking of some parameters.

$$\begin{pmatrix} \mathbf{z} \\ \mathbf{x} \end{pmatrix} = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Gamma}_k)$$

$$\boldsymbol{\mu}_k = \begin{pmatrix} \bar{\mathbf{z}}_k \\ \mathbf{A}_k \bar{\mathbf{z}}_k + \mathbf{b}_k \end{pmatrix}, \boldsymbol{\Gamma}_k = \begin{pmatrix} \boldsymbol{\Sigma}_k & \boldsymbol{\Sigma}_k \mathbf{A}_k^\top \\ \mathbf{A}_k \boldsymbol{\Sigma}_k & \mathbf{A}_k \boldsymbol{\Sigma}_k \mathbf{A}_k^\top + \boldsymbol{\Lambda}_k \end{pmatrix} \tag{12}$$

To avoid overfitting, we constrain the descriptor covariance matrix $\boldsymbol{\Sigma}$ to be diagonal, thereby drastically reducing the number of parameters to be estimated in our model. The gating probabilities are given by $\pi_k(\mathbf{z}) = \frac{1}{Z}|\boldsymbol{\Sigma}_k|^{-1} e^{-\frac{1}{2}(\mathbf{z}-\bar{\mathbf{z}}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{z}-\bar{\mathbf{z}}_k)}$.

The parameters are learned using a standard Expectation Maximization (EM) algorithm. We initialize the class centers and gating probabilities by clustering in the $\mathbf{x}$-space alone in order to separate points that have similar $\mathbf{z}$-values but different $\mathbf{x}$ values. (Including $\mathbf{z}$ in the initial clustering decreased the quality of separation between ambiguous cases). Results show that most of the ambiguities are resolved and the regressors indeed learn separate models for the multiple possible solutions that come from different regions of the pose space. Figure 17 shows the two most highly weighted modes of the distribution in 3D pose obtained by using a mixture of 8 regressors over some sample silhouettes. These two solutions usually capture the principal ambiguities, but valid reconstructions are often also present in some of the remaining 6 modes of the output.

The associated probabilities of these modes are given by the gating probabilities π_k of the regressors used

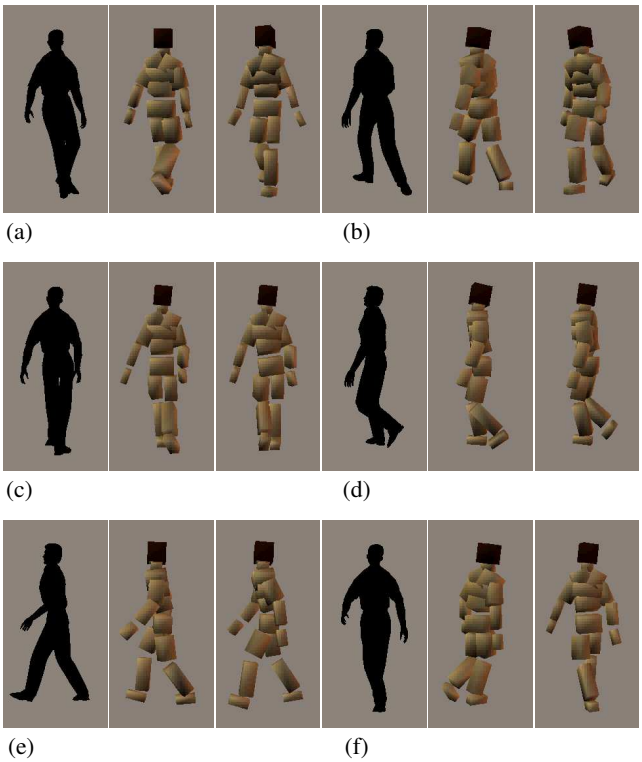


Fig. 17. Multiple possible 3D pose estimates obtained from individual silhouettes using a mixture of regressors. The two most likely modes of the distribution are shown in each case, and generally capture the two most evident reconstruction possibilities, illustrating cases of forward-backward ambiguity (a,b), kinematic flipping of the legs (c) and interchanging labels between the two legs (d,e). (f) shows an example where the first solution is a misestimate but feasible solutions are obtained in the other modes.

for the reconstruction. We find that these gating probabilities typically give a good idea of the true number of ambiguous solutions in the given case, but they do not always select the correct solution from among the generated possibilities. To get an idea of the number of cases where the system cannot choose a single 'correct' solution, we rank the various modes obtained by the regressors according to their *(i)* estimated probabilities π_k, and *(ii)* their accuracies obtained by comparison with the ground truth. We find that in 30-35% of the cases, the solution that is estimated as being most *likely* is actually incorrect — but most of these correspond to cases that are truly ambiguous — and the correct solution is usually amongst the few most probable ones.

Using a mixture model scheme in place of a single regressor allows most of the instances of compromised solutions from the single regressor to be resolved into several solutions capturing the different 3D possibilities (*e.g.* compare figures 9(e) and 17(e)). This gives the method the capability of accurately estimating possible 3D poses from single images — even in the cases of ambiguity — by outputing several possible solutions whenever they exist. Below we describe how to use these multiple possible solution sets across a sequence of silhouettes to allow smooth tracking free from glitches.

B. Condensation based tracking

The multimodal likelihoods obtained in the previous section can be used in a tracker that combines the modes across time to estimate a temporally coherent maximum likelihood trajectory of 3D poses. This is demonstrated by implementing a CONDENSATION [13] based tracking algorithm that uses the output density of our mixture model to assign likelihoods to its particles. We work with the assumption that state information from the current observation is independent of state information from the dynamics:

$$p(\mathbf{x}_t \,|\, \mathbf{z}_t, \mathbf{x}_{t-1}, \ldots) \;\propto\; p(\mathbf{x}_t \,|\, \mathbf{z}_t)\, p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}, \ldots) \quad (13)$$

The pose reconstruction ambiguity is reflected in the fact that the likelihood $p(\mathbf{x}_t|\mathbf{z}_t)$ is typically multimodal. It is often obtained by using Bayes' rule to invert to the many-to-one generative model $p(\mathbf{z}_t|\mathbf{x}_t)$, but we continue to work in our discriminative tracking framework and hence use $p(\mathbf{x}_t|\mathbf{z}_t)$ as opposed to $p(\mathbf{z}_t|\mathbf{x}_t)$. The dynamical model from section V-A.1 is used to generate an estimate of the 3D pose distribution $p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}, \ldots)$. Samples $(\check{\mathbf{x}}_t^i)$ from this distribution are then assigned weights $p(\check{\mathbf{x}}_t^i \mathbf{z})$ by the observation model density as given in (11).

Figure 18 shows tracking results obtained on our spiral walk test set using CONDENSATION with 2000 particles. In general, the method tracks through the correct modes of the observation density. Smooth tracks are produced, with the maximum likelihood reconstructions usually being more accurate than any of the 8 individual modes of the multimodal regressor output alone.

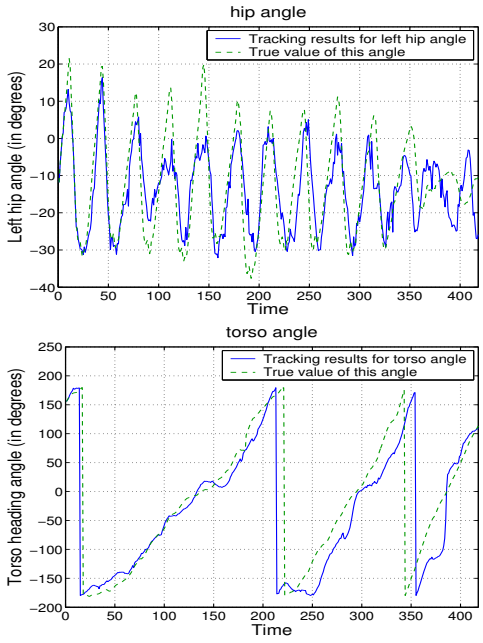


Fig. 18. Tracking results with a particle filter on a spiral walk test sequence using the mixture of regressors output as an observation model: *(Left)* left hip angle parameter, *(Right)* torso heading angle.

VII. DISCUSSIONS AND CONCLUSIONS

We have presented a method that recovers 3D human body pose from monocular silhouettes by direct nonlinear regression of joint angles against histogram-of-shape-context silhouette shape descriptors. Neither a 3D body model nor labelled image positions of body parts are needed, making the method easily adaptable to different people, appearances and representations of 3D human body pose. The regression is done in either linear or kernel space, using either ridge regression or Relevance Vector Machines. The main advantage of RVMs is that they allow sparse sets of highly relevant features or training examples to be selected for the regression. We have proposed two ways of overcoming the intrinsic ambiguity of the pose-from-monocular-observations problem: regressing the pose jointly on image observations and previous pose; and using a mixture of regressors in a multiple hypothesis tracking scheme. Both of these produce stable, temporally consistent tracking. Our mixture of regressors scheme has the capability to reconstruct 3D human pose accurately from

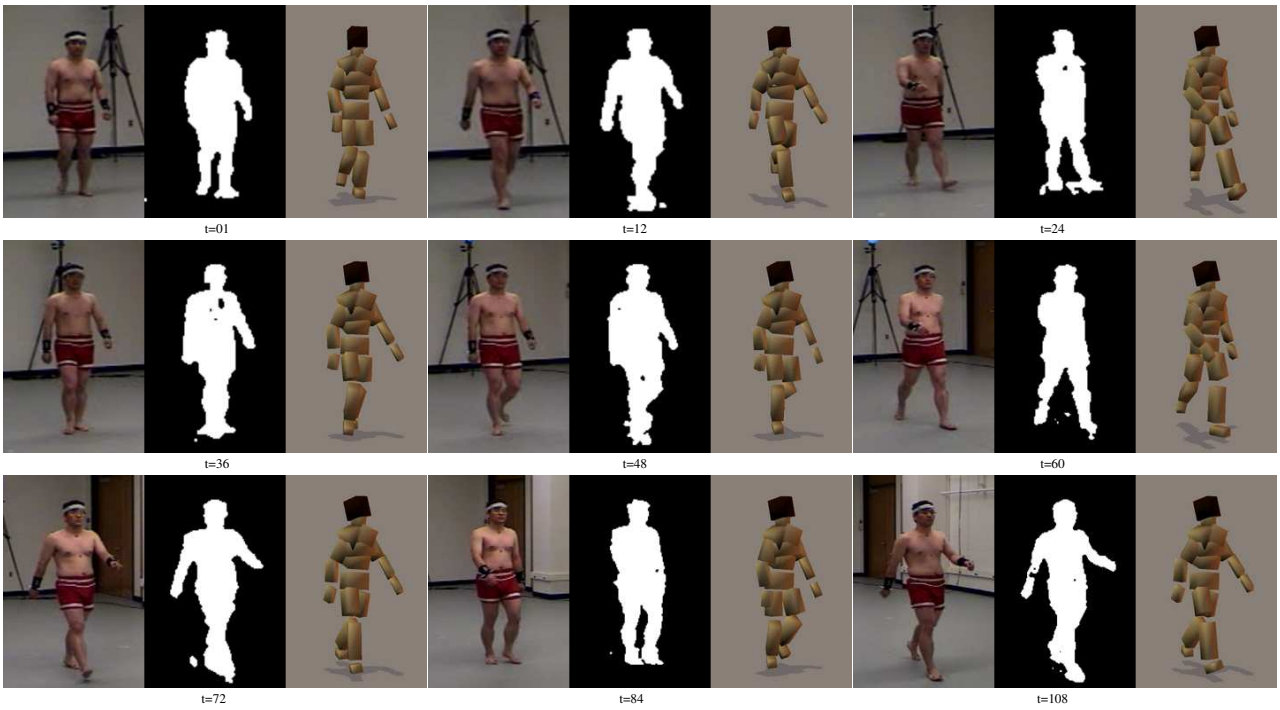


Fig. 19. 3D poses reconstructed from another test video sequence (obtained from http://mocap.cs.cmu.edu/). In this sequence the subject walks towards the camera causing a scale change by a factor of $\sim$2. (The images and silhouettes have been normalized in scale here for display purposes). Our scale invariant silhouette representation allows the algorithm to process a silhouette independent of its size or location in the image without disturbing the 3D pose recovery.

a single image, giving multiple possible poses whenever they exist.

Our kernelized RVM regressors retain only about $15-20\%$ of their training examples in the regression based tracking, thus giving a large effective reduction in storage space compared to nearest neighbour methods, which must retain the whole training database. Our methods show promising results, being about three times more accurate than the current state of the art [22].

Future work: We plan to investigate the extension of our regression based system to cover a wider class of human motions and also add structured representations to our model for dealing with greater variability in the 54 dimensional output space. On the vision side, we would like to include richer features, such as internal edges in addition to silhouette boundaries to reduce susceptibility to poor image segmentation.

Our linear RVMs directly select relevant features in the image descriptor space. This property may be useful for identifying better feature sets, not only for pose recovery and tracking, but also for human detection tasks.

ACKNOWLEDGMENTS

This work was supported by the European Union projects VIBES and LAVA, and the research network PASCAL.

REFERENCES

[1] A. Agarwal and B. Triggs. 3D Human Pose from Silhouettes by Relevance Vector Regression. In *Int. Conf. Computer Vision & Pattern Recognition*, 2004.

[2] A. Agarwal and B. Triggs. Learning to Track 3D Human Motion from Silhouettes. In *Int. Conf. on Machine Learning*, 2004.

[3] A. Agarwal and B. Triggs. Tracking Articulated Motion with Piecewise Learned Dynamical Models. In *European Conf. Computer Vision*, 2004.

[4] V. Athitsos and S. Sclaroff. Inferring Body Pose without Tracking Body Parts. In *Int. Conf. Computer Vision & Pattern Recognition*, 2000.

[5] V. Athitsos and S. Sclaroff. Estimating 3D Hand Pose From a Cluttered Image. In *Int. Conf. Computer Vision*, 2003.

[6] S. Belongie, J. Malik, and J. Puzicha. Shape Matching and Object Recognition using Shape Contexts. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 24(4):509–522, 2002.

[7] C. Bishop. *Neural Networks for Pattern Recognition*, chapter 6. Oxford University Press, 1995.

[8] M. Brand. Shadow Puppetry. In *Int. Conf. Computer Vision*, pages 1237–1244, 1999.

[9] C. Bregler and J. Malik. Tracking People with Twists and Exponential Maps. In *Int. Conf. Computer Vision & Pattern Recognition*, pages 8–15, 1998.

[10] A. D'Souza, S. Vijayakumar, and S. Schaal. Learning Inverse Kinematics. In *Int. Conf. on Intelligent Robots and Systems*, 2001.

[11] K. Grauman, G. Shakhnarovich, and T. Darrell. Inferring 3D Structure with a Statistical Image-Based Shape Model. In *Int. Conf. Computer Vision*, pages 641–648, 2003.

[12] N. Howe, M. Leventon, and W. Freeman. Bayesian Reconstruction of 3D Human Motion from Single-Camera Video. In *Neural Information Processing Systems*, 1999.

[13] M. Isard and A. Blake. CONDENSATION – Conditional Density Propagation for Visual Tracking. *Int. J. Computer Vision*, 29(1):5–28, 1998.

[14] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):79–87, 1991.

[15] T. Joachims. Making large-Scale SVM Learning Practical. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.

[16] F. Jurie and M. Dhome. Hyperplane Approximation for Template Matching. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 24(7):996–1000, 2002.

[17] D. Lowe. Object Recognition from Local Scale-invariant Features. In *Int. Conf. Computer Vision*, pages 1150–1157, 1999.

[18] G. Mori and J. Malik. Estimating Human Body Configurations Using Shape Context Matching. In *European Conf. Computer Vision*, volume 3, pages 666–680, 2002.

[19] D. Ormoneit, H. Sidenbladh, M. Black, and T. Hastie. Learning and Tracking Cyclic Human Motion. In *Neural Information Processing Systems*, pages 894–900, 2000.

[20] V. Pavlovic, J. Rehg, and J. MacCormick. Learning Switching Linear Models of Human Motion. In *Neural Information Processing Systems*, pages 981–987, 2000.

[21] Y. Rubner, C. Tomasi, and L.J. Guibas. A Metric for Distributions with Applications to Image Databases. In *Int. Conf. Computer Vision*, Bombay, 1998.

[22] G. Shakhnarovich, P. Viola, and T. Darrell. Fast Pose Estimation with Parameter Sensitive Hashing. In *Int. Conf. Computer Vision*, 2003.

[23] H. Sidenbladh, M. Black, and L. Sigal. Implicit Probabilistic Models of Human Motion for Synthesis and Tracking. In *European Conf. Computer Vision*, volume 1, 2002.

[24] C. Sminchisescu and B. Triggs. Covariance Scaled Sampling for Monocular 3D Body Tracking. In *Int. Conf. Computer Vision & Pattern Recognition*, 2001.

[25] C. Sminchisescu and B. Triggs. Kinematic Jump Processes For Monocular 3D Human Tracking. In *Int. Conf. Computer Vision & Pattern Recognition*, June 2003.

[26] A. Smola and B. Schölkpof. A Tutorial on Support Vector Regression. *NeuroCOLT2 Technical Report NC2-TR-1998-030*, 1998.

[27] B. Stenger, A. Thayananthan, P. Torr, and R. Cipolla. Filtering Using a Tree-Based Estimator. In *Int. Conf. Computer Vision*, 2003.

[28] C. Taylor. Reconstruction of Articulated Objects from Point Correspondances in a Single Uncalibrated Image. In *Int. Conf. Computer Vision & Pattern Recognition*, 2000.

[29] M. Tipping. The Relevance Vector Machine. In *Neural Information Processing Systems*, 2000.

[30] M. Tipping. Sparse Bayesian Learning and the Relevance Vector Machine. *J. Machine Learning Research*, 1:211–244, 2001.

[31] K. Toyama and A. Blake. Probabilistic Tracking in a Metric Space. In *Int. Conf. Computer Vision*, pages 50–59, 2001.

[32] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[33] O. Williams, A. Blake, and R. Cipolla. A Sparse Probabilistic Learning Algorithm for Real-Time Tracking. In *Int. Conf. Computer Vision*, 2003.

Part II

Other Works 2000–2004

Chapter 4

Low-Level Vision

This chapter contains four papers dealing with low-level vision, including image filtering and resampling, feature extraction and feature correspondence.

Summary of paper 8, "Empirical Filter Estimation for Subpixel Interpolation and Matching"

This paper, presented at the 2001 International Conference on Computer Vision [Tri01a], focuses on the issue of low-level image interpolation and resampling. It stems from dissatisfaction with the orthodox treatment of sampling found in countless textbooks, in which all signals are assumed to have been strictly bandlimited before sampling, so that sinc-function based interpolation and resampling are the gold standards against which all other methods should be judged. The paper points out that natural images notably fail to conform to this model. They almost always contain significant admixtures of abrupt intensity steps, which dominate their high-frequency spectra, and which would often generate visually unpleasant 'ringing' if the "ideal" of perfect bandlimited sampling were really put into practice. In default of a satisfying alternative sampling theory, the paper takes an empirical approach and asks what forms of resampler actually work best for real images, in the sense of minimizing empirical resampling error. 1D and 2D minimal error filters are designed by direct optimization over a database of training images, under various assumptions about the camera response and various plausible error metrics. The overall conclusion is that the empirically optimal filters are relatively insensitive to the exact assumptions made and have a windowed-sinc-like form with 2-4 oscillations on each side of the main peak, but that the optimal windowing function is distinctly non-classical, with step-like behaviour at pixel boundaries rather than smooth descent.

Summary of paper 9, "Boundary Conditions for Young - van Vliet Recursive Filtering"

Young & van Vliet gave a method of approximating Gaussian-based filters by an efficient forwards-backwards Infinite Impulse Response recursion [YvV95, vVYV98, YvVvG02]. This short technical note submitted to IEEE Transactions on Image Processing [Tri04a] points out that the heuristic method of transitioning from the forwards to the backwards recursion in these papers leads to significant amplitude and phase distortion near the right hand boundary, and derives a transition matrix

linking the forwards and backwards recursions that eliminates this distortion.

Summary of paper 10, "Detecting Keypoints with Stable Position, Orientation and Scale under Illumination Changes"

Presented at the 2004 European Conference on Computer Vision [Tri04b], this paper generalizes the popular Förstner-Harris keypoint detector to transformation groups broader than pure translations, and also shows how to incorporate additional illumination invariance. Keypoints (also called "points of interest", or more casually "corners") are isolated image points that are in some sense particularly salient, so that they can be reliably re-detected in other images. They are the foundation of the "local feature" approach to vision, in which either image patches centred on the points, or more generally local image descriptors based on the patches, are used for image correspondence, motion, scene reconstruction, object recognition, *etc*. The detectors pioneered by Moravec [Mor77], Förstner [F$\ddot{\text{o}}$86, FG87, F$\ddot{\text{y}}$94] and Harris & Stevens [HS88] use various measures of *geometric stability* as salience metrics, *i.e*., as keypoints, they explicitly select points that seem likely to serve as stable 'anchors' under geometric matching. This is quantified in terms of the stability of *self*-matching of the patch against itself — a quantity that can be approached by various differential image calculations. All of the above detectors use stability under small *translations* as their criterion, in which case it turns out that the differential calculations centre on the so-called "second moment matrix" or "structure tensor" of the local image patch, $\int_{\text{patch}} \nabla I \, \nabla^\top I \, dx$. The current paper shows how to generalize these constructs to select points that have prespecified degrees of stability under transformation groups larger than pure translation, and derives detectors for arbitrary subgroups of the 2D affine group. It also shows how to correct the resulting generalized scatter matrix to enforce invariance under various common types of illumination variations (affine changes of illumination and uniform illumination gradients). The final detector is implemented in a multiscale framework, but (although it detects locally-affinely-stable points) it has not yet been extended to incorporate full affine invariance. The overall philosophy here is that one should be able to prespecify the various degrees of stability needed by the application (for example, those needed to calculate stable local descriptors based on the detected keypoints), and ask the detector to explicitly select keypoints that match these requirements.

Summary of paper 11, "Joint Feature Distributions for Image Correspondence"

This paper from the 2001 International Conference on Computer Vision [Tri01b] presents a new approach to the problem of multi-image correspondence for rigid and partially- or near-rigid scenes. It points out that instead of using rigid geometric constructs such as conventional matching tensors, multi-image visual correspondence problems can be formulated probabilistically, as the estimation of "Joint Feature Distributions" (JFD's) — joint probability distributions for the image positions of corresponding features across several images. The resulting distributions implicitly encode the uncertain scene geometry, and could potentially be used to create a probabilistic analogue of rigid scene reconstruction. Correspondence prediction and feature transfer is achieved very naturally by conditioning on the given observations to give a lower-dimensional joint distribution for the unobserved variables. The remainder of the paper then develops an explicit parametrization and estimation method for some particularly interesting families of JFD's with close links to conventional matching constraints. Algebraically, the estimation algorithm is closely related to the

standard linear (“8 point” and “4 point”) methods for estimating the corresponding matching tensor, but with the additional benefit that certain scene geometries that are singular for conventional matching tensor estimation cause no problems at all for JFD based methods, and may even improve the efficiency of correspondence search. In particular, planar and near-planar scenes are handled gracefully by the JFD analogue of the fundamental matrix / epipolar constraint: as the scene becomes progressively more planar, the conditional JFD correspondence search windows naturally and progressively shrink from the full epipolar line to small ellipses centred on the correspondences predicted by the underlying plane homography. Hence explicit model selection is not necessary: the “epipolar” model continues to work stably for planar scenes. The paper finishes with a short technical appendix that gives some of the algebraic geometry behind the construction.

Empirical Filter Estimation for Subpixel Interpolation and Matching

Bill Triggs

CNRS-INRIA, 655 avenue de l'Europe, 38330 Montbonnot, France.

Bill.Triggs@inrialpes.fr ◇ *http://www.inrialpes.fr/movi/people/Triggs*

Abstract

We study the low-level problem of predicting pixel intensities after subpixel image translations. This is a basic subroutine for image warping and super-resolution, and it has a critical influence on the accuracy of subpixel matching by image correlation. Rather than using traditional frequency-space filtering theory or *ad hoc* interpolators such as splines, we take an empirical approach, finding optimal subpixel interpolation filters by direct numerical optimization over a large set of training examples. The training set is generated by subsampling larger images at different translations, using subsamplers that mimic the spatial response functions of real pixels. We argue that this gives realistic results, and design filters of various different parametric forms under traditional and robust prediction error metrics. We systematically study the performance of the resulting filters, paying particular attention to the influence of the underlying image sampling regime and the effects of aliasing ("jaggies"). We summarize the results and give practical advice for obtaining subpixel accuracy.

Keywords: image filtering, subpixel interpolation, super-resolution, aliasing, subpixel matching.

1 Introduction

What is the best way to obtain subpixel accuracy from images? – Ultimately, it is a question of which feature extraction, filtering or interpolation scheme to use. Interpolation schemes are often motivated theoretically [18,9,10], either as finite-width approximations to the infinite 'sinc' filters that exactly interpolate band-limited signals in Nyquist sampling theory[1], or in terms of convenient parametric forms such as cubic splines. But at bottom the question is empirical. Real images are neither cubic nor strictly band-limited to the pixel spacing. Nyquist and spline theory neither give us optimal interpolators for them, nor tell us which of the many suboptimal approximations to use. These issues exist in 1D but become even thornier for images, where multidimensional sampling artifacts — notably the 'aliasing' of non-grid-aligned edges into staircases of discrete steps ("jaggies") — can seriously degrade feature detection and geometric precision.

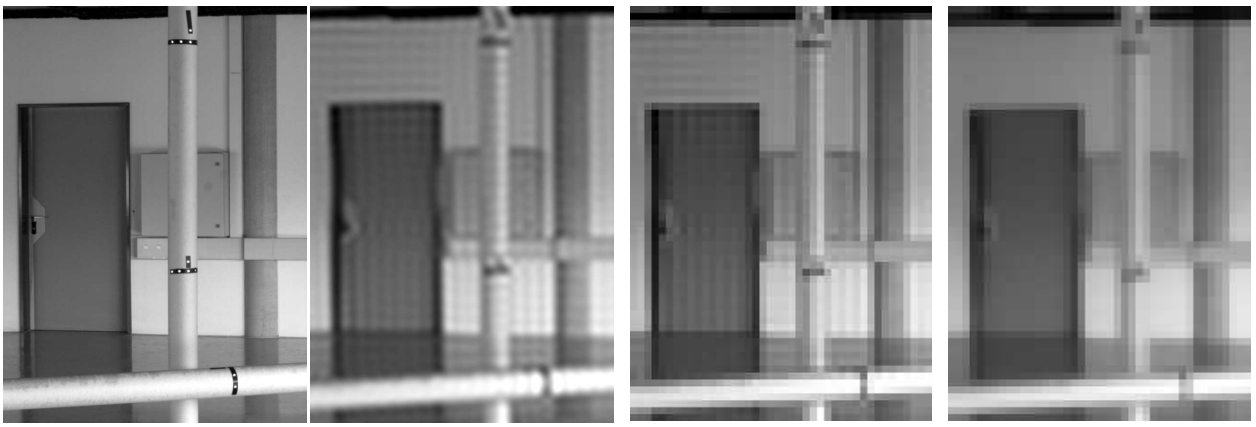
In default of an effective theory, we treat these issues empirically, designing accurate interpolators by minimizing their prediction errors over sets of training images containing subpixel translations. Our current implementation is oriented towards quantitative matching (subpixel correlation) rather than human perception: we choose intensity-related error metrics (*e.g.* L1, L2) rather than perceptual ones (*e.g.* [12]); we consider geometric as well as photometric accuracy; and we pay particular attention to the (strong) influence of the underlying pixel spatial response function. Our approach could also be used for super-resolution [16,3,5,4], but here we use individual not multiple images, and we aim to predict what the original camera would return if shifted, not an enhanced image. An illuminating 1D analytic study complementary to our 2D empirical one is [15]. For a unified "information optimizing" approach to sampling and reconstruction, see [7]. For subpixel reconstruction based on learned "codebooks" see [6,1].

Our main aim was to establish 'good working practice' for accurate subpixel image manipulation, side-stepping the bewildering range of methods available for filter design [9,10]. Forms like splines are essentially heuristic, and we do not accept that strict band-limitation and sinc interpolation are the ideal approach for vision, to be approximated as well as the available computational power allows. The high frequency spectra of natural images are dominated by phase-coherent $1/f$ components produced by step edges. Bandwidth truncation of these produces significant high frequency 'ringing', which disturbs both the human eye and accurate computer vision algorithms. As figure 1 shows, abruptly band-limited images simply do not "look right": oscillations propagate out from each edge, making featureless surfaces look textured and generally confusing the signal[2] . If band-limited images are to

Extended version of a paper appearing in the 2001 IEEE Int. Conf. Computer Vision.

[1] We assume familiarity with basic sampling theory. Sampling a continuous signal on a discrete grid folds ('aliases') high spatial frequencies down to their fractional parts (in cycles per grid unit), and thus confuses the signal. Signals limited to the **Nyquist frequency** band $[-\frac{1}{2}, \frac{1}{2}]$ (no wavelengths less than 2 pixels) have no aliasing and hence can be reconstructed exactly. Bandwidth limitation / band-limited signal reconstruction can be implemented by continuous / discrete convolution against a **sinc function** $\mathrm{sinc}(\pi x) \equiv \sin(\pi x)/(\pi x)$, whose abruptly truncated Fourier transform (1 for $|f| < \frac{1}{2}$ and 0 elsewhere) but infinite, slowly decaying oscillating tails in x make it expensive to implement accurately by direct convolution.

[2] These artifacts are more visible on the computer than in print. Here



(a) original image *(b)* band-limited *(c)* band-limited + subsampled *(d)* blurred + subsampled

Figure 1: The effects of abrupt bandwidth limitation. An original image *(a)*, limited to $1/15$ of the bandwidth *(b)* then decimated (point-subsampled) $15\times$ *(c)*. The band-limited images *(b,c)* have characteristic "ringing" artifacts: Nyquist frequency waves propagate outwards from each edge making smooth surfaces look striped or plaided, especially between parallel edges. Blurring *(a)* with a $\sigma = 7$ Gaussian before subsampling gives a slightly less sharp image *(d)*, but no ringing.

be used, the downstream image processing modules must be capable of handling such artifacts.

Notation: PRF stands for Pixel (spatial) Response Function (§2). Image points are denoted $\boldsymbol{x} = (x, y)$ with input pixels (centres) at integer coordinates. w denotes filter half-width: a half-width w interpolator reconstructing sub-pixel location (x, y) accesses all integer positions (i, j) with $|x - i|, |y - j| < w$. §2 discusses our method of generating training data, §3 comments on aliasing, §4 sketches our filter design method, §5 presents our experimental findings, and §6 summarizes.

2 Image Formation & Subsampling

To estimate subpixel interpolation filters, our training method (§4) needs a large body of training images with accurately labelled subpixel shifts. Suitable data could perhaps be collected experimentally, but this would be time consuming and error-prone. Instead, we have preferred to synthesize suitable data by carefully subsampling real source images. This allows us to rapidly generate a large body of training examples with guaranteed-accurate subpixel shifts, and it can synthesize measurements corresponding to any given camera response (*i.e.* PRF, see below), which gives us great flexibility in filter design. The danger is that subsampled images may be "unrealistic" — too unrepresentative of real scenes to produce useful interpolators. This section argues that appropriate subsampling does capture the relevant aspects of real scenes. The argument relies on two empirical properties of the underlying continuous input images:

(*i*) Within small enough regions, many natural scenes are nearly scale invariant, with smooth featureless power-law spectra. Zooming out does not change their local statistics: local windows on zoomed scenes look similar to local windows on unzoomed ones. In particular, rescaling step edges changes neither their appearance nor their $1/f$ spectra, so images dominated by abrupt transitions between uniform regions are locally scale-invariant, and interpolators fitted to such data should remain valid for other edge-dominated scenes. One set of windows with edges at all positions and orientation looks much like another. Zooming does change the ratio of uniform regions to edge-containing ones, but this has little effect on the results as uniform regions are uninformative for interpolator training (they constrain only the average of the filter coefficients).

(*ii*) The trained interpolators depend mainly on characteristics of the input spectra within a few octaves of the pixel sampling frequency, particularly the way that useful information shades into aliasing: higher frequencies are usually too strongly attenuated to cause much aliasing and too phase-sensitive to reconstruct[3], and lower ones look like constants within the filter's limited window. Hence, the un-subsampled discrete source images contain all of the frequencies needed to synthesize realistic images of a zoomed-out scene, including sampling and aliasing effects. By 'careful subsampling' we mean exactly this syn-

they are due to abrupt bandwidth truncation alone, not finite filter width or image boundary effects (we used carefully windowed FFT on much larger images). Poorly truncated sinc filters give even worse ringing.

[3]Linear interpolators are convolution-like but do not necessarily preserve sinusoids, so they can synthesize frequencies not present in their input. They can be viewed as convolvers acting on densified signals constructed by intercalating zeros between the input samples, which extends their spectra.

thesis.

At least for scene classes with some local scale invariance (and in practice, most scenes are locally edge-dominated), these properties suggest that subsampling should be a useful strategy for synthesizing training data provided that we arrange for the subsampled images to mimic as closely as possible those that would have been produced by the same camera looking at a rescaled (zoomed out) scene, including all signal degradation, sampling and aliasing effects. To approximate this we use a simplistic local model of image formation. We assume that pixels are identical; that each responds linearly to the total light flux falling on it which is in turn a linear function of the underlying scene luminance; and that this linear process is shift-invariant: if the scene luminance is represented in image plane coordinates as an 'ideal' image $L(\boldsymbol{x})$, a pixel at $\boldsymbol{x}_0$ responds with $R(\boldsymbol{x}_0) = (P * L)(\boldsymbol{x}_0) \equiv \int_{\boldsymbol{x}} P(\boldsymbol{x}_0 - \boldsymbol{x})\, L(\boldsymbol{x})\, \boldsymbol{dx}$, where "$*$" denotes convolution and the **Pixel Response Function (PRF)** $P(\boldsymbol{x}_0 - \boldsymbol{x})$ represents the combined effects of scattering, blurring, diffraction, other signal degradations, and flux integration across the pixel's receptive zone. Clearly this model is only approximate — real pixel responses are nonlinear (saturation, clipping, quantization), and the various degradations depend on 3D and image position, wavelength, optics and geometry[4] — but it will suffice as a local model for predicting "average" subpixel behaviour. Interpolators that adapt to local imaging conditions are left for future work.

To predict the image R' of the rescaled scene, we apply the desired PRF P at the new scale rather than the old one: $R'(\boldsymbol{x}) = (P * L_\lambda)(\boldsymbol{x}) = (P_{1/\lambda} * L)(\lambda \boldsymbol{x})$ where $L_\lambda(\boldsymbol{x}) \equiv L(\lambda \boldsymbol{x})$ is the reduced scene ($\lambda > 1$) and $P_{1/\lambda}(\boldsymbol{x}) \equiv P(\boldsymbol{x}/\lambda)$ is the expanded PRF. In practice this amounts to simply convolving the discrete source image with a sampled version of the expanded PRF before subsampling: the original PRF is not eliminated, but for $\lambda \gtrsim 4$ it is so much narrower than the expanded one that its additional smoothing effect is negligible.

Typical PRF's (see fig.2) are around one pixel wide, with a form dominated by pixel geometry for narrower PRF's and optics (blurring, diffraction) for wider ones. §3 shows how to estimate the PRF of a real camera. §5 experiments with a number of idealized PRF's in order to determine which aspects of PRF form are important for subpixel accuracy.

[4] Apart from blur, pixel geometry is the main PRF determinant. "Interline" CCD's include shielded channels to prevent smearing caused by incoming light during readout shifting, so only about 40–70% of the pixel area is light sensitive, while "full frame" CCD's use the light collectors themselves for readout and have sensitive areas nearer 100% [17, 10]. Chip-surface micro-lenses and colour filters alter the PRF, and position dependent variations occur near the borders of large chips where the incoming rays strike the surface quite obliquely. Sharpening filters are often included in the electronics to reduce blur, but cause asymmetry and ringing that may lead to clipping near strong edges. Saturation and clipping severely degrade the geometric precision, so for precise applications it is best to underexpose the images and to switch off any deblurring filters.

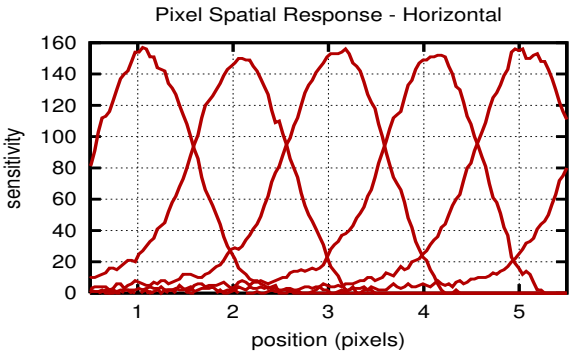
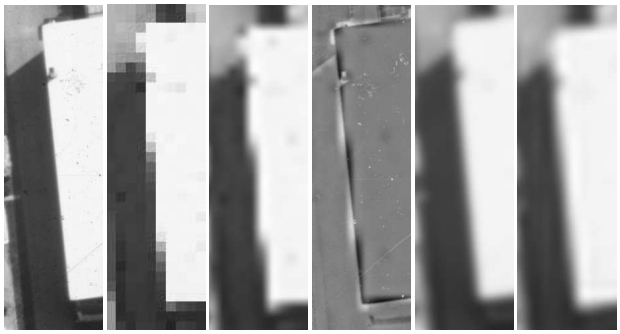


Figure 2: Experimental horizontal pixel responses (PRF's) for an analog video camera. These give the light sensitivity of the pixels as the signal moves across them.



(a) (b) (c) (d) (e) (f)

Figure 3: Aliasing along an edge due to poor sampling. When the original image (a) is decimated $15\times$ it shows clear jaggies (b). Even an optimal reinterpolation filter can do little to recover from these (c). The prediction error (d) is concentrated along strong edges, in phase with the jaggies. A more smoothly (0.5 pixel Gaussian) subsampled image (e) has no visible jaggies and can be reconstructed (f) with far less error.

3 Aliasing

Digitization converts smooth sloping edges into staircases of more or less abrupt steps or "jaggies", an effect (incorrectly) called "aliasing". The experiments below show that for well-focused images at typical daylight noise levels, aliasing is easily the dominant source of error in subpixel interpolation. Fig.3 illustrates this on the extreme case of a point-subsampled image. Even well designed interpolation filters can do little to recover from aliasing once it is present. In fact, *no* finite-window filter can eliminate all aliasing. As an edge moves forwards one pixel, its jaggies move sideways by one cycle (one "jag") by a Moiré effect. As the edge becomes more closely aligned with one of the coordinate axes, its jaggies become ever more widely

spaced and move ever more rapidly sideways. Eventually, they are more widely spaced than the filter window and the filter has no way of predicting when they will pass through it.

The Moiré properties of aliasing along edges provide an easy means of measuring the effective horizontal and vertical PRF's of the camera. If we take a shallow step edge and move parallel to the coordinate axis it is nearly aligned with, the step gradually encroaches on the pixels we cross as we move. Hence, these pixel intensities effectively give a fine sampling (with spatial resolution $1/(\text{aliasing period of edge})$) of the cumulative response of a pixel as an edge moves across it. Taking derivatives along the line of samples gives the non-cumulative pixel response function, *i.e.* the response as a line edge moves across the pixel. For example, in fig.3 (b), sampling along a vertical line gives a fairly abrupt step as the line hits its jaggie, and differentiating this gives the vertical PRF, which is nearly a delta function. The results in fig.2 were obtained in this way.

4 Filter Design

Our design method for subpixel interpolators is extremely simple: we collect a *training set* of pairs of images with known subpixel relative displacements, choose a *parametric form* for the filter and an *error metric* for its prediction error, and *numerically minimize* the filter's total prediction error on the test images over its parameters. Consider each element in turn.

Training images: Training requires a large set of accurately labelled subpixel data. Any images could be used. The study below subsampled large photogrammetric images to generate the training data. As discussed in §2, we believe subsampling to be adequately realistic, and it allows arbitrarily large and varied data sets to be generated quickly and reliably. It also allowed a detailed comparison of the effects of different PRF's, which would not otherwise have been possible. A $15\times$ subsampling factor provided reasonably dense interpolation and ensured that the PRF of the original source camera had negligible influence on the results. We considered only grayscale images as the non-collocated sampling of Bayer-style $\begin{smallmatrix}\text{RG}\\ \text{GB}\end{smallmatrix}$ colour arrays greatly complicates subpixel issues.

Parametric filter: We designed filters from several different parametric families. All were based on square windows centred on the inter-pixel square containing the subpixel position being predicted. The mask of a half-width w filter includes exactly w input pixels on each side. Half-width 1 filters access just the four corners of the 1 pixel square containing the point being predicted, and in practice tend to approximate bilinear interpolation. The exact conventions for our convolution sums are given in the appendix. Our simplest parametrization leaves the weights

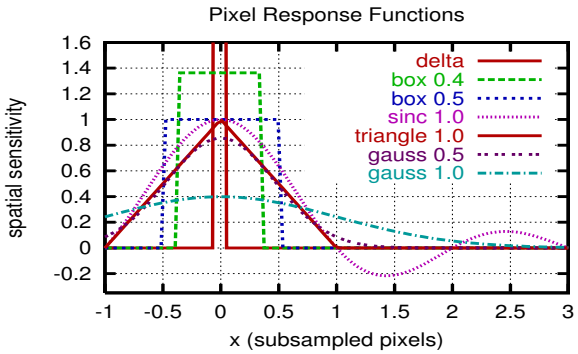


Figure 4: The pixel response functions for which filters were designed in this study.

at all positions and shifts as free parameters. Alternatively, we can fit a separable filter, where each weight is the product of the corresponding x-filter and y-filter ones. Finally, we can parametrize each inter-pixel section of the separable filters as a separate cubic, which further reduces the number of parameters, couples the weights at different shifts, and allows subpixel shifts not in the original training set to be generated. We did not implement continuity constraints between the cubic sections because the optimal filters often have small discontinuities in practice. We can also require the impulse response to be symmetric if desired, but real PRF's are often asymmetric, especially along scan lines.

Error metric: Our designs minimize intensity (grey-level) prediction error rather than, *e.g.* perceptual or feature location metrics. The errors are summed over all resampled pixels and all subpixel shifts being considered. Let δI_i be the difference between the observed and the predicted intensity at pixel i and σ be a robustness-scale parameter. We designed filters under the following error norms. L2: classical least squares, $|\delta I_i|^2$. L1: least absolute value, $|\delta I_i|$. Lorentz: $\log\left(1+\frac{1}{2}|\delta I_i/\sigma|^2\right)$. Smoothed L1: $\sqrt{1+\frac{1}{2}|\delta I_i/\sigma|^2}-1$.

Optimization method: The most conspicuous feature of the optimization problem is the huge number of measurements available — more than 10^7 for the designs given below. It often has many parameters too. Separable filters have comparatively few, but the largest non-separable filters tested below — half-width 6 with $15\times$ subsampling — have $(2\cdot 6\cdot 15)^2 = 32400$ free parameters in 255 groups of 144 (the groups being independent if no inter-shift smoothness constraints are enforced — and none seem to be needed). However, the data is very strong, so apart from its size the optimization problem is relatively benign. Given these characteristics, we chose to use Liu & Nocedal's elegant LBFGS (limited memory quasi-Newton) optimizer [11,13] from Netlib (http://www.netlib.org). The required values and

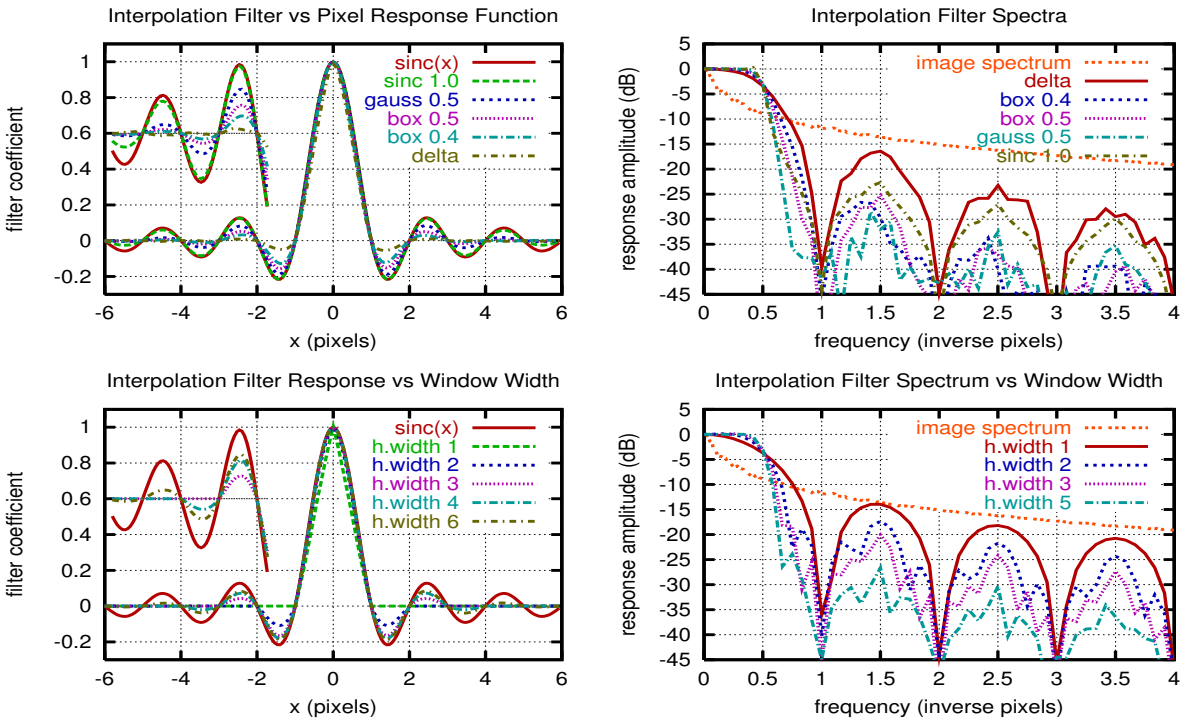


Figure 5: Optimal interpolation filter impulse responses (left) and spectra (right) for separable L2-norm filters designed for various pixel response functions (top, for half-width $w = 6$) and filter widths (bottom, for the gauss 0.5 PRF). A sinc function (left) and the image spectrum (right) are plotted for comparison.

gradients of the error function with respect to the filter parameters are calculated by a predict-and-accumulate cycle through the test images to evaluate gradients with respect to the parameters of the non-separable filter, followed by chain-rule reduction to a smaller parameter space if required.

The basis of the LBFGS approach [11,13] is to observe changes in the cost gradient as the method progresses, and use these to dynamically build up and maintain a rank k approximation to the problem's Hessian, for some fixed k. The approximation is maintained using BFGS updates and used to predict quasi-Newton optimization steps. LBFGS is designed for large problems in which the full Hessian can not be stored or decomposed. In typical applications k is chosen quite small (in the range 2–10), but given the high cost of evaluating our cost function we prefer to set it to 100 or more, as this gives more Newton-like steps. The tests below typically required around 80 iterations for the easier problems, and up to about 200 for the harder ones (particularly the large spline filters).

Note that we use LBFGS not just for the nonlinear (separable filter or robust error norm) problems, but also for the classical linear least squares ones (free non-separable filter in the L2 error norm) and the non-smooth (L1 error norm) ones. We originally included LAPACK's direct linear least squares routine DGELS for the linear problems, but found that it was much slower than the iterative code LBFGS on large problems. For the L1 norm, the use of a smooth code is not ideal, but there are so many equations here that the simplex method based linear programming codes we tried were hopelessly inefficient. We would expect interior point L1 codes to do better, but LBFGS is perhaps not a bad approximation to these as it 'learns' curvature in directions in which the active set (and hence the gradient) changes. In any event it handles the L1 problems fairly well, although we had to loosen the convergence threshold to prevent thrashing near convergence.

5 Experimental Results

We made a systematic study of subpixel interpolation filter design using the above methods. There is only space for a brief summary of what we learned. The results shown are based on the *Ascona Workshop* test images — eight 1800×1800 aerial views of a suburban scene, taken with a photogrammetric camera and a high-resolution colour film scanner (http://www.photogrammetry.ethz.ch/news/events/ascona/dataset/dataset.html, see fig.6(a)). The images were converted to grayscale and subsampled $15\times$ with the desired PRF's at integer shifts between 0 and 15 (*i.e.* subpixel x and y shifts in $(0, 1...15)/15$), so the training PRF's are accurate to around $15\times$ the target resolution. The Ascona images have $\sigma \sim 2$ blur and some spots and

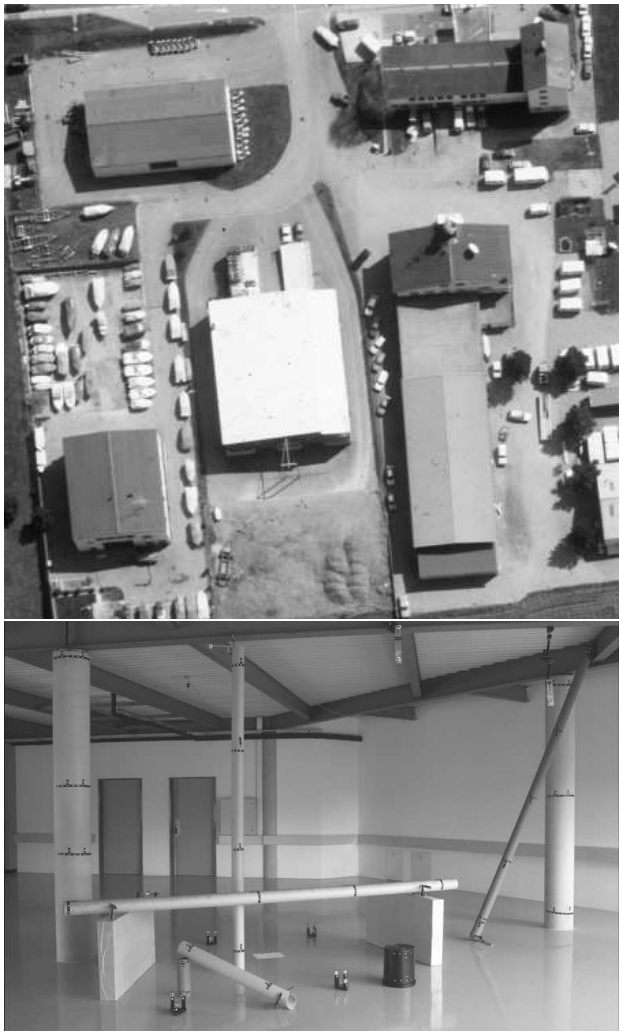


Figure 6: Examples of the Ascona (top) and Imetric (bottom) test images.

scratches, but training on sharper, cleaner indoor images (six 3072×2048 images from IMETRIC using a Kodak DCS460 digital camera with $\sigma \sim 0.7$, fig.6(b)) gave essentially identical results, as expected.

Design results: We designed separable and non-separable $15\times$ interpolation filters of half-widths $w = 1...6$ pixels under the above four error norms for the following seven PRF's (see fig.4). *Delta:* unsmoothed decimation / point subsampling (by far the worst case for aliasing). *Box 0.4, 0.5:* box filters of half-width $5/15$ and $7/15$ representing ideal square pixels with sensitive areas of full widths $(2\cdot5+1)/15 = 0.73$ and 1.0 pixels, the former being representative of 'interline' CCD's, the latter of 'full frame' ones [17]. *Triangle 1.0:* width 1 triangular response representing ideal bilinear interpolation. *Gauss 0.5, 1.0:* Gaussian responses of $\sigma = 0.5, 1.0$ pixels, representing typical, and fairly large, amounts of optical blurring. *Sinc 1.0:* the ideal Nyquist-band-limiting PRF (may not be physically realizable owing to negative coefficients). Sinc subsampling used carefully windowed FFT to minimize truncation and boundary effects. Where not otherwise noted the following default parameters were used: training and test PRF, Gaussian 0.5; filter, separable, all entries free, half-width $w = 3$; error norm, L2. We train and test on the same (subsampled Ascona) images: validation against the IMETRIC set showed that the effects of overfitting were negligible.

Fig.5 shows how the impulse responses and spectra of the interpolators vary with design PRF and width. All of the designs have sinc-like oscillations, but except for the sinc PRF these decay much more rapidly than a sinc or classically-windowed sinc function. The Delta PRF interpolator is nearly triangular with very small side-lobes, while those for the broader PRF's have 2–3 visible side-lobes. Reducing the design width progressively suppresses the side-lobes until the filters become triangular at $w = 1$. Half-widths 2–4 are probably the best compromise in practice. The spectra show the same story, with all filters having side-lobes significantly smaller than a truncated sinc of the same width[5]. As expected, the spectra of the training images are relatively featureless at these scales, so the designs are insensitive to the particular images used.

It is instructive to view the sinc PRF interpolators as windowed versions of the ideal sinc interpolator. Dividing by $\mathrm{sinc}(x)$ gives the implied empirically optimal windowing functions (fig.7). The recovered window functions are broader and more abrupt than most classical ones (the relatively broad Welch $(1-(x/w)^2)$ and Lanczos $(\mathrm{sinc}(x/w))$ ones are shown for comparison). As a result, the sinc PRF interpolators are far from band-limited: their spectra are actually *broader* than those resulting from most other PRF's. The optimal windows also descend by abrupt steps, rigidly scaling each side-lobe. This gives the optimal interpolators small but significant derivative discontinuities at each zero crossing, especially where the last two lobes drop to zero in two $\sim 40\%$ steps. In effect, *the input data is windowed, not the interpolator*: the interpolators for all fractional shifts share the same input points and the same windowing.

The fitted bicubic interpolators tell the same story for the other PRFs. They have significantly larger side-lobes than commonly recommended bicubics such as Catmull-Rom or Mitchell-Netravali splines [12], and again small but distinct derivative discontinuities (fig.8). Optimal interpolation does *not* imply either gentle windowing or high-order smoothness.

Filters designed under different error norms are for the most part nearly identical, although L1 designs typically

[5]The slight bumpiness of the spectra is due to FFT aliasing — the Nyquist limit for $15\times$ interpolators is only 7.5 (pixels)$^{-1}$ — and noise — $\mathcal{O}(10^7)$ training pixels leaves $\mathcal{O}(\sqrt{10^{-7}}) \sim -35$ dB noise.

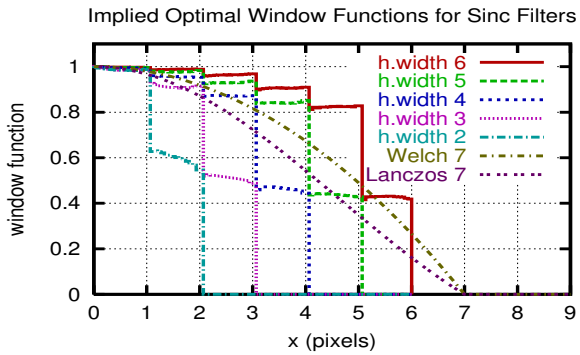


Figure 7: The ratios $\text{filter}(\boldsymbol{x})/\,\text{sinc}(\boldsymbol{x})$ for $w = 2...6$ sinc PRF filters give the implied optimal windowing functions for sinc interpolators of these widths.

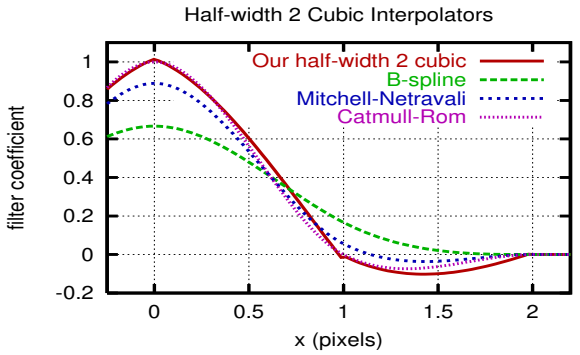


Figure 8: The optimal cubic filters (here $w = 2$ for the Gauss 0.5 PRF) have small derivative discontinuities and larger side lobes than either the cubic-convergent Catmull-Rom spline or Mitchell & Netravali's visually preferred "$(\frac{1}{3}, \frac{1}{3})$" design [12].

have slightly smaller side lobes than L2 ones, with the two robust norms intermediate between these two. The differences are largest for the Delta PRF. Aliasing ensures that the input data contains significant numbers of outliers, but these are distributed fairly symmetrically so they do not upset the L2 fits too much. The 2D designs have small but consistent non-separabilities (about 5% of amplitude for most PRF's, mostly in the central ± 1 pixel square), but this does not seem to give them a noticeable performance edge over their separable counterparts, except perhaps for the Delta PRF.

Test results: Now consider the experimental performance of the designed filters. Our tests use large images subsampled as for the training images. We consider two aspects of subpixel performance: intensity interpolation error (which the filters were designed to minimize) and feature location error. Intensity testing is self evident: we interpolate the test images and compare the result to the original shifted versions, accumulating errors in some chosen error metric (here L1). The location tests were designed to study the "ground truth" performance of subpixel correlation matching (not, *e.g.*, feature detection). We defined a set of strong isolated test features with accurately-known subpixel locations by selecting 100–150 "points of interest" — locations at which correlation matching of the image against itself gives high spatial accuracy in all directions — in each unsubsampled test image. For this we used a Harris (Lucas-Kanade-Harris-Förstner) detector with inflated scale parameters (so that the detected features should be strong points of interest in the subsampled images), and suppression parameters set to eliminate all spurious responses along edges and in textured regions. This provided a test set that was independent of the PRF and shift used (real 3D features exist independently of these!), and that had minimal aliasing artifacts (in particular, no spurious responses along edges). Visual inspection confirmed the high quality of the chosen points. The location tests used the subsampled locations of these points as correlation centres. For each centre and shift, we interpolated the image and made an exhaustive search of the correlation at all $1/15^{th}$-pixel locations within a window of $\pm 20/15$ pixels around the known output centre. The best result found was taken as the match, and the resulting translational errors were recorded and analyzed. To minimize edge effects, correlations were calculated by integrating the L2 (squared) intensity prediction error over a $\sigma = 2.0$ Gaussian window. Optionally, independent noise was added to the base and target images before matching. This approach is idealistic in using near-perfect features with no geometric or photometric distortion between the base and target windows, but it allows us to study just one thing at a time, here aliasing and subpixel accuracy.

Fig.9 summarizes the performance of the designed filters. The salient point is the overwhelming influence of the test image PRF on all aspects of subpixel performance. Aliasing along edges is by far the dominant source of error in these plots, and the main factor controlling aliasing is the test PRF: the narrower it is, the more aliasing and error there is. The Delta PRF (decimation) is particularly bad in this respect and dominates the error plots. In contrast, as the top row shows, the design PRF for the interpolation filter has relatively little influence. Basically, filters designed for any PRF of similar width give similar results. The plots in the bottom row show that there is little advantage in extending the filter half-width beyond about 3 pixels. Spatially, the errors are largest at half-pixel shifts, as would be expected.

Given the extent to which smoother PRF's reduce aliasing, we can ask whether it is also useful to smooth the images *after* sampling. Unsurprisingly, post-sampling smoothing of both source and test images does reduce interpolation error. Fig.10 suggests that small amounts ($\sigma \approx 0.5$ pixels) of post-smoothing also slightly reduce the location error in correlation matching, both for noise-

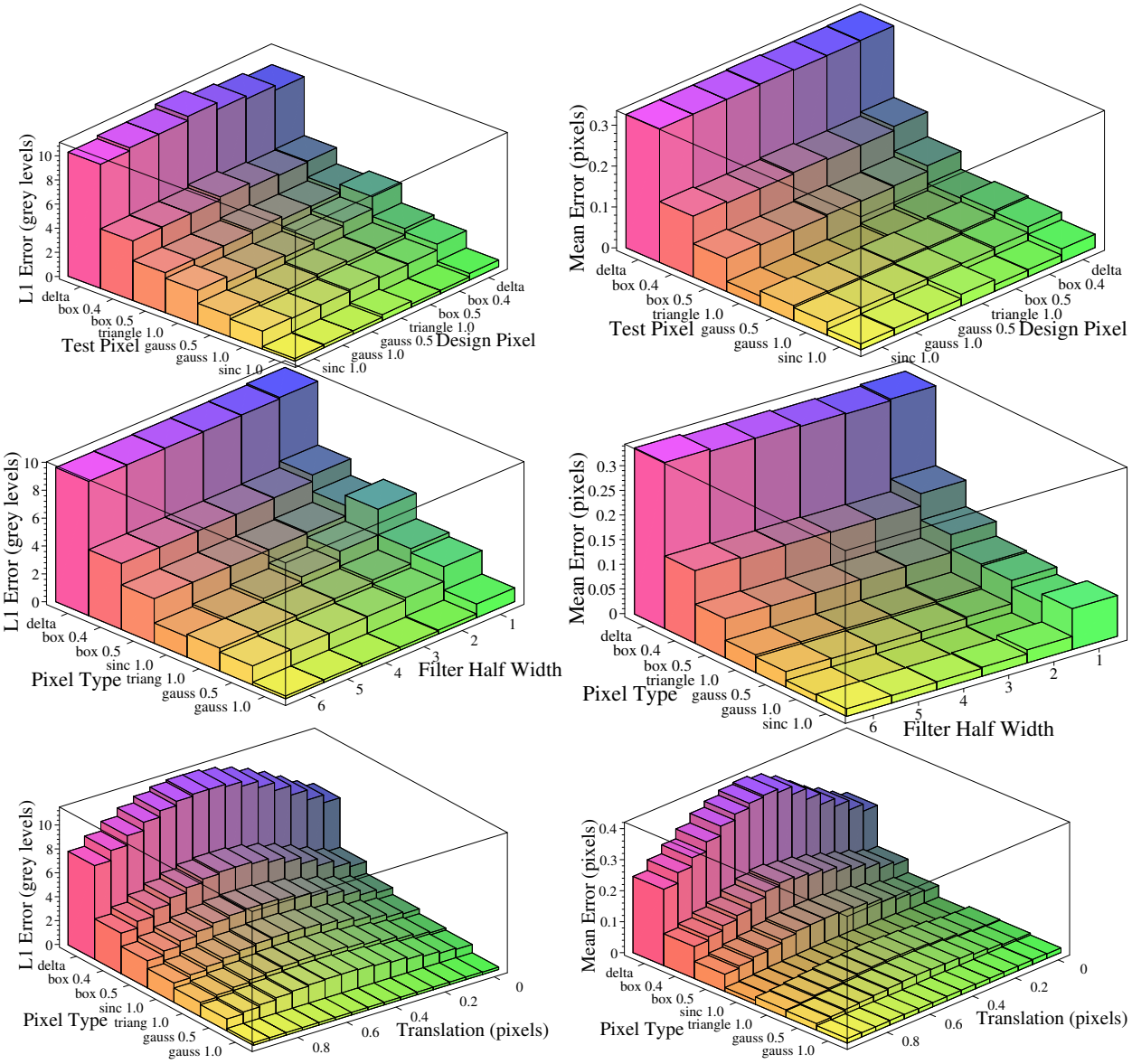


Figure 9: Interpolation error (left) and location error (right) versus (top) design and test pixel type, (middle) filter half-width, (bottom) actual subpixel x translation (averaged over y). Unless otherwise noted all filters are tested on the PRF's they were designed for.

less (no added noise) and quite noisy images. However the benefit is minor and is probably due to the effective increase in the size of the correlation integration window. Post-smoothing greater than about 0.8 pixels rapidly degrades spatial accuracy. Note also the poor performance of the sinc PRF filter at high noise levels — the extended oscillations of sinc filters give them larger squared integrals, and hence more noise accumulation, than comparable Gaussians.

6 Summary and Conclusions

For everyday cameras and noise levels, aliasing ("jaggies") is the dominant cause of accuracy loss in subpixel interpolation and matching, not random noise. Aliasing is determined mainly by the Pixel spatial Response Function (PRF). Over-narrow PRF's have large aliasing and much lower accuracy. Image processing after sampling does little to reduce the effects of aliasing, but a small amount of additional optical blurring *before* sampling can sometimes improve accuracy considerably. The effective PRF should ideally have a standard deviation of around 0.5 pixels but

Half width	Interval	x	$1-x$	$x^2(1-x)$	$x(1-x)^2$
2	$[0,1]$	-0.03125	1.01464	0.21293	0.67536
	$[1,2]$	0.00881	-0.00709	-0.30911	-0.49580
3	$[0,1]$	-0.03884	1.01786	0.23888	0.82584
	$[1,2]$	0.02460	-0.00784	-0.54197	-0.80243
	$[2,3]$	-0.00803	0.00154	0.15611	0.19234
4	$[0,1]$	-0.04076	1.01453	0.26186	0.86611
	$[1,2]$	0.02788	-0.00502	-0.59981	-0.88859
	$[2,3]$	-0.01095	-0.00009	0.25106	0.35183
	$[3,4]$	0.00215	0.00114	-0.05962	-0.10759

Figure 11: Coefficients of optimal cubic spline filters of half width w=2,3,4 for the gauss 0.5 pixel. x is measured from the start of each interval.

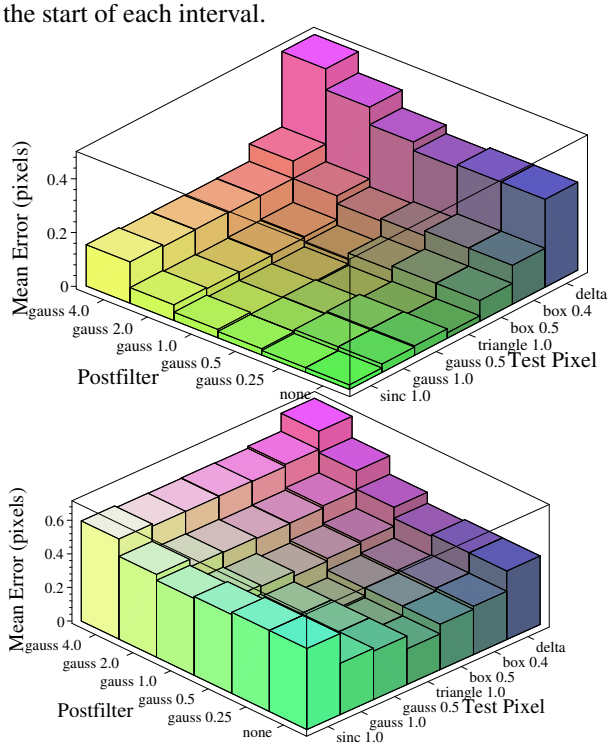


Figure 10: Location error versus test pixel type and amount of post-sampling smoothing, for the gauss 0.5 filter. The top plot is with no added noise, the bottom one with Gaussian noise of $\sigma = 20$ gray-levels.

its exact form is not critical. Visually, this corresponds to just enough blurring to make jaggies along strong edges disappear, *c.f.* fig.3(e).

Whatever the PRF, it is easily measured by taking derivatives along strong nearly-axis-aligned edges, and a custom subpixel interpolation filter can then be designed for it. Optimal interpolators of any desired parametric form can be designed by explicitly minimizing their total L2 or robust interpolation error over a set of training images. Suitable training images can be synthesized by subsampling larger ones, using the desired design PRF at different relative shifts. The resulting optimal filters have rapidly decaying sinc-function-like oscillations, separable filters of half-width 2–3 being fully sufficient for most practical applications. The filter forms depend on the widths of their design PRF's, but are relatively insensitive to the exact PRF shape, the error norm and the training images used. Filters designed for different but broadly similar PRFs give similar interpolation and matching performance, so the design and application PRF's do not need to be closely matched. Smooth windowing and high order continuity are *not* needed for optimality: the optimal subpixel interpolators have small derivative discontinuities at their zero crossings, and their implied windowing functions have abrupt steps there.

Future work: A useful minor extension would be to design interpolators for half-windows, to allow optimal signal reconstruction near occlusion boundaries. Other low-level image operators such as feature detectors would probably also benefit from explicit empirical performance optimization, the main barrier being (as here) the lack of suitable training data.

We only considered linear convolution filters here but the approach extends easily to nonlinear ones, *e.g.* linear combinations of nonlinear functions of pixel intensities or intensity differences... It would be interesting to see whether such filters could provide better resistance to aliasing. Kernel-based learning methods such as Support Vector Machines could perhaps also be used [14], although speed might be a problem and our initial attempts to train an SVM filter failed owing to the huge number of measurements involved. Despite the many advantages of linear sinusoidal sampling theory, we do feel that some sort of nonlinear theory built from steps, corners and their integrals (ramps, *etc.*) is needed for computer vision, as the underlying signal is to a large extent built out of such elements at sparse but arbitrary subpixel positions and orientations, rather than sinusoids. At present we have no such theory, but psychophysical data and recent results on independent components analysis and kernel based learning all seem to point in this direction [2, 8, 14].

Conventions

Conventions for subpixel filtering can be confusing. To understand ours, consider the standard continuous convolution integral for output signal I_1 in terms of input signal I_0, evaluated at position $\boldsymbol{x} + \boldsymbol{dx}$:

$$I_1(\boldsymbol{x} + \boldsymbol{dx}) \;=\; \int_{\boldsymbol{y}} W(\boldsymbol{x} + \boldsymbol{dx} - \boldsymbol{y})\, I_0(\boldsymbol{y}) \boldsymbol{dy}$$

For the discrete case we simply change the integral to a sum, supposing that $\boldsymbol{x}, \boldsymbol{y}$ are integer positions and $\boldsymbol{dx} \in [0,1) \times [0,1)$ is the subpixel position we want to evaluate I_1 at, and changing variables to $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{y}$:

$$I_1(\boldsymbol{x} + \boldsymbol{dx}) \;=\; \sum_{z_1, z_2 = -w}^{w-1} W(\boldsymbol{z} + \boldsymbol{dx})\, I_0(\boldsymbol{x} - \boldsymbol{z})$$

Hence we need the input I_0 at integer positions, and the impulse response W at whatever subpixel ones we need in $[-w, w) \times [-w, w)$. Which integer position we store the output $I_1(\boldsymbol{x} + \boldsymbol{dx})$ at is up to us.

References

[1] S. Baker and T. Kanade. Limits on super-resolution and how to break them. In *Int. Conf. Computer Vision & Pattern Recognition*, volume 2, pages 372–379, 2000.

[2] A.J. Bell and T.J. Sejnowski. The 'independent components' of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997.

[3] T. Boult and G. Wolberg. Local image reconstruction and subpixel restoration algorithms. *Computer Vision, Graphics & Image Processing*, 55(1):63–77, 1993.

[4] D. Capel and A. Zisserman. Automated mosaicing with super-resolution zoom. In *Int. Conf. Computer Vision & Pattern Recognition*, pages 885–891, June 1998.

[5] M.C. Chiang and T.E. Boult. Efficient super-resolution via image warping. *Image & Vision Computing*, 18(10):761–771, 2000.

[6] W. Freeman, E. Pasztor, and O. Carmichael. Learning low-level vision. *Int. J. Computer Vision*, 40(1):25–48, 2000.

[7] F.O. Huck, C.L. Fales, and Z. Rahman. An information theory of visual communication. *Phil. Trans. Roy. Soc.*, A 354:2193–2248, 1996.

[8] A. Hyvärinen and P.O. Hoyer. Emergence of phase and shift invariant features by decomposition of natural images into independent feature subspaces. *Neural Computation*, 12(7):1705–1720, 2000.

[9] B. Jähne, editor. *Handbook of Digital Image Processing for Scientific Applications*. CRC Press, Florida, 1997.

[10] B. Jähne, H. Haußecker, and P. Geißler, editors. *Handbook of Computer Vision and Applications*. Academic Press, 1999.

[11] D. Liu and J. Nocedal. On the Limited Memory BFGS method for large scale optimization. *Mathematical Programming*, B 45:503–528, 1989.

[12] D. P. Mitchell and A. N. Netravali. Reconstruction filters in computer graphics. *Computer Graphics (SIGGRAPH'88)*, 22(4):221–228, 1988.

[13] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer-Verlag, 1999.

[14] C. Papageorgiou, F. Girosi, and T. Poggio. Sparse correlation kernel analysis and reconstruction. Technical report, MIT A.I. Lab, 1998. Memo 1635 (CBCL Memo 162).

[15] S. K. Park and R. A. Schowengerdt. Image sampling, reconstruction and the effect of sample-scene phasing. *Applied Optics*, 21(7):3142–3151, 1982.

[16] S. Peleg, D. Keren, and L. Schweitzer. Improving image resolution using subpixel motion. *Pattern Recognition Letters*, 5:223–226, 1987.

[17] M.R. Shortis and H.A. Beyer. Sensor technology for digital photogrammetry and machine vision. In K.B. Atkinson, editor, *Close Range Photogrammetry and Machine Vision*, pages 106–155. Whittles Publishing, Roseleigh House, Latheronwheel, Caithness, Scotland, 1996.

[18] G. Wolberg. *Digital Image Warping*. IEEE Computer Society Press, 1990.

Boundary Conditions for Young - van Vliet Recursive Filtering

Bill Triggs

***Abstract*— Young & van Vliet designed computationally efficient methods for approximating Gaussian-based convolutions by running a recursive IIR filter forwards over the input signal, then running a second IIR filter backwards over the first filter's output. However, to transition between the two filters, they used a suboptimal heuristic that produces significant amplitude and phase distortion for all points within about 3 standard deviations of the right-hand boundary. We derive a simple linear transition rule that eliminates this distortion.**

***Index Terms*— Gaussian smoothing, bidirectional recursive filtering, boundary conditions.**

I. Introduction

Young & van Vliet (YvV) have developed computationally efficient forwards-backwards IIR recursions for Gaussian filters [1], Gaussian derivatives [2], and Gabor filters [3]. See [3] for their most recent design rules for Gaussians, and [4] for space-variant extensions and a performance comparison with other IIR Gaussian methods including Deriche's original method [5], [6]. All of the YvV filters work forwards, recursively calculating a running sum u_t as a linear combination of the input signal i_t and the k previous u values, then work backwards calculating a running sum v_t as a linear combination of u_t and the l previously-calculated v values:

$$u_t \;=\; i_t + \textstyle\sum_{j=1}^{k} a_j\, u_{t-j}\,, \quad t = 1 \ldots n \qquad (1)$$

$$v_t \;=\; u_t + \textstyle\sum_{j=1}^{l} b_j\, v_{t+j}\,, \quad t = n \ldots 1 \qquad (2)$$

The final output is a scaled version of v_t, and $\{a_{j,j=1\ldots k}\}$ and $\{b_{j,j=1\ldots l}\}$ are suitably chosen filter coefficients. For Gaussians, YvV choose $k{=}l$ and $\boldsymbol{a}{=}\boldsymbol{b}$ [1], [3]. For other filters, i_t may be a linear transformation of the original input signal, *e.g.* a discrete derivative for derivative filters [1].

II. The Problem with Heuristic Boundary Conditions

To complete the specification (1, 2), we must fix initial conditions for u near $t{=}1$ and for v near $t{=}n$. For u, we can pretend that the signal existed and took some nominal constant value i_- (typically either 0 or i_1) for all $t{<}1$. The correct initialization at $t{=}1$ is then to set all $u_{1-j,j=1\ldots k}$ to $i_-/(1-\sum_{j=1}^{k} a_j)$, the steady state response to an infinite stream of i_-'s. Similarly, if we could suppose that for all $t{>}n$, u_t took some constant value u_+, the correct condition at $t{=}n$ would be to set $v_{n+j,j=1\ldots l}$ to $u_+/(1-\sum_{j=1}^{l} b_j)$, the steady state response to an infinite stream of u_+'s. YvV apparently do exactly this, with $i_-{=}i_1$ and $u_+{=}u_n$ (*c.f.* [3] equations (20,21)). Another plausible choice for u_+ would be $i_+/(1-\sum_{j=1}^{k} a_j)$, the steady state u resulting from an infinite stream of constant input values i_+ above $t{=}n$ (typically, i_+ would be either i_n or 0).

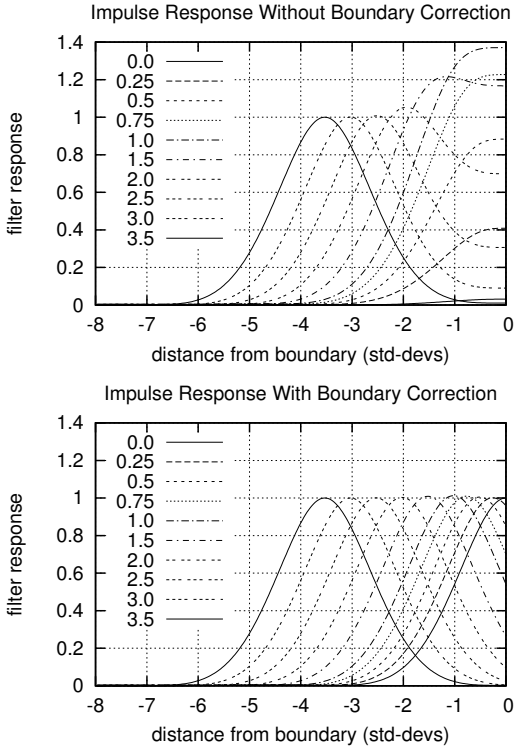


Fig. 1. Impulse responses for points at various numbers of standard deviations from the right boundary, for a YvV Gaussian filter, with the standard YvV boundary heuristic $u_+{=}u_n$ (top) and with our new boundary correction (10) (bottom). The corrected responses are much closer to the desired (truncated Gaussian) form.

Unfortunately, neither choice for u_+ is correct. If the forwards filter were continued to $t{\gg}n$ with input i_+, its output would decay smoothly from u_n to $i_+/(1-\sum_{j=1}^{k} a_j)$ within a few standard deviations, and the corresponding backwards filter would take all elements of this "advance warning" signal into account when calculating its response. In fact, the forwards-backwards process *only* gives the correct overall impulse response when the full double recursion is run without truncation. Incorrect truncation causes significant amplitude and phase (geometric position) distortion for all points within about 3.5 standard deviations of the boundary. Fig. 1 illustrates the extent of the problem.

III. Derivation of Linear Boundary Correction

To correct for the effects of truncation, we notionally extend the forwards-backwards pass to $t{\to}\infty$ assuming a constant input value i_+ above $t{=}n$, and calculate the coefficients $\{v_{n+j,j=1\ldots l}\}$ that would result from this infinite extension, given i_+ and the final forwards filter state $\{u_{n-j,j=0\ldots k-1}\}$. The whole process is linear so the v's must be linear functions of the u's and i_+. First suppose that $i_+ = 0$. Gathering the u's, v's into running k, l vectors $\boldsymbol{u}$, $\boldsymbol{v}$, the forwards and backwards

Manuscript received 19 July 2004.

The author is with GRAVIR-CNRS-INRIA, 655 avenue de l'Europe, 38330 Montbonnot, France. *Bill.Triggs@inrialpes.fr*, *http://www.inrialpes.fr/lear/people/triggs*. This work was partially supported by European Union research projects LAVA and aceMedia.

$$\boldsymbol{M} \;=\; \frac{1}{\begin{array}{c}(1{+}a_1{-}a_2{+}a_3)\,(1{-}a_1{-}a_2{-}a_3)\\ \cdot\,(1{+}a_2{+}(a_1{-}a_3)\,a_3)\end{array}} \begin{pmatrix} -a_3a_1{+}1{-}a_3^2{-}a_2 & (a_3{+}a_1)(a_2{+}a_3a_1) & a_3(a_1{+}a_3a_2) \\ a_1{+}a_3a_2 & -(a_2{-}1)(a_2{+}a_3a_1) & -(a_3a_1{+}a_3^2{+}a_2{-}1)a_3 \\ a_3a_1{+}a_2{+}a_1^2{-}a_2^2 & a_1a_2{+}a_3a_2^2{-}a_1a_3^2{-}a_3^3{-}a_3a_2{+}a_3 & a_3(a_1{+}a_3a_2) \end{pmatrix} \tag{3}$$

passes become:

$$\boldsymbol{u}_t \;=\; \boldsymbol{A}\,\boldsymbol{u}_{t-1} \;=\; \boldsymbol{A}^{t-n}\,\boldsymbol{u}_n \tag{4}$$

$$\boldsymbol{v}_t \;=\; \boldsymbol{I}_1\,\boldsymbol{u}_t + \boldsymbol{B}\,\boldsymbol{v}_{t+1} \tag{5}$$

where

$$\boldsymbol{u}_t \;\equiv\; \begin{pmatrix} u_t \\ \vdots \\ u_{t-k+1} \end{pmatrix} \qquad \boldsymbol{A} \;\equiv\; \begin{pmatrix} a_1 & \cdots & a_{k-1} & a_k \\ 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \\ 0 & \cdots & 1 & 0 \end{pmatrix} \tag{6}$$

$$\boldsymbol{v}_t \;\equiv\; \begin{pmatrix} v_t \\ \vdots \\ v_{t+l-1} \end{pmatrix} \qquad \boldsymbol{B} \;\equiv\; \begin{pmatrix} b_1 & \cdots & b_{l-1} & b_l \\ 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \\ 0 & \cdots & 1 & 0 \end{pmatrix} \tag{7}$$

and $\boldsymbol{I}_1 = (1\ 0\ \ldots\ 0)^\top(1\ 0\ \ldots\ 0)$ is an $l{\times}k$ matrix with a 1 in the top left corner and zeros elsewhere. Combining these equations for all $t{\geq}n$, we have $\boldsymbol{v}_n \;=\; \left(\sum_{i=0}^{\infty} \boldsymbol{B}^i\,\boldsymbol{I}_1\,\boldsymbol{A}^i\right)\,\boldsymbol{u}_n$. We need to calculate the $l{\times}k$ matrix $\boldsymbol{M} \equiv \sum_{i=0}^{\infty} \boldsymbol{B}^i\,\boldsymbol{I}_1\,\boldsymbol{A}^i$ that links the final forwards state $\boldsymbol{u}_n$ to the initial backwards one $\boldsymbol{v}_n$. By its recursive definition:

$$\boldsymbol{M} \;=\; \boldsymbol{I}_1 + \boldsymbol{B}\,\boldsymbol{M}\,\boldsymbol{A} \tag{8}$$

Given $\boldsymbol{a}, \boldsymbol{b}$, it is easy to find the corresponding entries of $\boldsymbol{M}$ by solving this linear system, *e.g.* using a symbolic algebra package such as MAPLE. For example, for the Gaussian filter recommended by YvV, $k{=}l{=}3$, $\boldsymbol{a}{=}\boldsymbol{b}$, the corresponding $\boldsymbol{M}$ is given in (3) above.

Finally, to handle nonzero i_+, we can simply reduce to the $i_+{=}0$ case by subtracting the constant-u response $u_+ = i_+/(1 - \sum_{j=1}^{k} a_j)$ from each component of $\boldsymbol{u}_n$, apply $\boldsymbol{M}$, then add back the corresponding constant-v response $u_+/(1 - \sum_{j=1}^{l} b_j)$ to get $\boldsymbol{v}_n$.

IV. Summary of Method

In summary, Young & Van Vliet recursive filters suffer from severe amplitude and phase distortion at the right boundary unless the backwards running coefficients are initialized from the forwards ones as follows, where $\boldsymbol{M}$ is given by (8, 3):

$$\begin{pmatrix} v_n \\ \vdots \\ v_{n+l-1} \end{pmatrix} \;=\; \boldsymbol{M} \begin{pmatrix} u_n - u_+ \\ \vdots \\ u_{n-k} - u_+ \end{pmatrix} + \begin{pmatrix} v_+ + u_n \\ \vdots \\ v_+ \end{pmatrix} \tag{9}$$

$$u_+ \;=\; \frac{i_+}{1 - \sum_{j=1}^{k} a_j} \qquad v_+ \;=\; \frac{u_+}{1 - \sum_{j=1}^{l} b_j} \tag{10}$$

An implementation of this for 2D Gaussian image filtering is available on the author's homepage.

References

[1] I. Young and L. van Vliet, "Recursive implementation of the Gaussian filter," *Signal Processing*, vol. 44, pp. 139–151, 1995.

[2] L. van Vliet, I. Young, and P. Verbeek, "Recursive Gaussian derivative filters," in *Int. Conf. Pattern Recognition*, Brisbane, 1998, pp. 509–514.

[3] I. Young, L. van Vliet, and M. van Ginkel, "Recursive Gabor filtering," *IEEE Trans. Sig. Proc.*, vol. 50, no. 11, pp. 2799–2805, 2002.

[4] S. Tan, J. Dale, and A. Johnston, "Performance of three recursive algorithms for fast space-variant Gaussian filtering," *Real-Time Imaging*, vol. 9, pp. 215–228, 2003.

[5] R. Deriche, "Recursively implementing the Gaussian and its derivatives," in *Int. Conf. Image Processing*, Singapore, 1992.

[6] ——, "Recursively implementing the Gaussian and its derivatives," INRIA, Tech. Rep., 1993.

Detecting Keypoints with Stable Position, Orientation and Scale under Illumination Changes

Bill Triggs
GRAVIR-CNRS-INRIA, 655 avenue de l'Europe, 38330 Montbonnot, France.
Bill.Triggs@inrialpes.fr ◇ *http://www.inrialpes.fr/lear/people/triggs*

Abstract

Local feature approaches to vision geometry and object recognition are based on selecting and matching sparse sets of visually salient image points, known as 'keypoints' or 'points of interest'. Their performance depends critically on the accuracy and reliability with which corresponding keypoints can be found in subsequent images. Among the many existing keypoint selection criteria, the popular Förstner-Harris approach explicitly targets geometric stability, defining keypoints to be points that have locally maximal self-matching precision under translational least squares template matching. However, many applications require stability in orientation and scale as well as in position. Detecting translational keypoints and verifying orientation/scale behaviour post hoc is suboptimal, and can be misleading when different motion variables interact. We give a more principled formulation, based on extending the Förstner-Harris approach to general motion models and robust template matching. We also incorporate a simple local appearance model to ensure good resistance to the most common illumination variations. We illustrate the resulting methods and quantify their performance on test images.

Keywords: keypoint, point of interest, corner detection, feature based vision, Förstner-Harris detector, template matching, vision geometry, object recognition.

1 Introduction

Local-feature-based approaches have proven successful in many vision problems, including scene reconstruction [16,5], image indexing and object recognition [20,21,32, 33,23,24,25]. The basic idea is that focusing attention on comparatively sparse sets of especially salient image points — usually called **keypoints** or **points of interest** — both saves computation (as most of the image is discarded) and improves robustness (as there are many simple, redundant local cues rather than a few powerful but complex and delicate global ones) [37]. However, local methods must be able to find 'the same' keypoints again in other images, and their performance depends critically on the reliability and accuracy with which exactly corresponding points can be found. Many approaches to keypoint detection exist, including 'corners' [2,17,38,28,4], parametric image models [3,31,1], local energy / phase congruency [27,29,30,18], and morphology [35,19]. One of the most popular is that developed by Förstner & Gülch [7,9] and Harris & Stephens [15] following earlier work by Hannah [14] and Moravec [26]. This approach brings

Published in the 2004 European Conf. on Computer Vision.
This research was supported by the European Union FET-Open research project VIBES.

the accuracy issue to the fore by *defining* keypoints to be points at which the predicted precision of local least squares image matching is locally maximal [14,22,6,10,12,11]. Notionally, this is implemented by matching the local image patch against itself under small translations, using one of a range of criteria to decide when the 'sharpness' of the resulting correlation peak is locally optimal. Moravec did this by explicit single-pixel translations [26]; Hannah by autocorrelation [14]; and Förstner by implicit least squares matching, using Taylor expansion to re-express the accuracy in terms of the eigenvalues of the **scatter matrix** or **normal matrix** of the local image gradients, $\int \nabla I^\top \nabla I \, \boldsymbol{dx}$ [7,9,8]. All of these methods use rectangular patches, usually with a scale significantly larger than that of the image gradients used. This is problematic for patches that contain just one strong feature, because the self-matching accuracy for these is the same wherever the feature is in the patch, *i.e*. the matching-based approach guarantees good self-matching accuracy, but not necessarily accurate *centring* of the patch on a visible feature. Working independently of Förstner, Harris & Stephens improved the localization performance by replacing the rectangular patches with Gaussian windows (convolutions) with a scale similar to that of the derivatives used [15]. With Gaussian-based derivative calculations and more careful attention to aliasing, the method has proven to be one of the most reliable keypoint detectors, especially in cases where there are substantial image rotations, scalings or perspective deformations [33,24].

One problem with the Förstner-Harris approach is that it optimizes keypoints only for good *translational* precision, whereas many applications need keypoints that are stable not only under translations, but also under rotations, changes of scale, perspective deformations, and changes of illumination (*c.f*. [34]). In particular, many local feature based object recognition / matching methods calculate a vector of local image descriptors at each keypoint, and later try to find keypoints with corresponding descriptors in other images [20,21,32,23,24,25]. This usually requires the extraction of a dominant orientation and scale at each keypoint, and keypoints that have poorly defined orientations or scales tend to produce descriptors that vary too much over re-detections to be useful. Hence, it seems useful to develop keypoint detectors that explicitly guarantee good orientation and scale stability, and also good stability under local illumination variations. This is the goal of the current paper, which generalizes the Förstner-Harris self-matching argument to include non-translational motions, and also provides improved resistance to illumination variations by replacing simple least squares matching with an illumination-compensated matching method related to Hager & Belhumeur's [13].

Much of the paper focuses on the low-level task of *characterizing the local stability of matching under geometric transformations and illumination variations.* The Förstner-Harris approach shows that such analysis is a fruitful route to practical keypoint detection in the translational case, and we argue that this continues to hold for more general transformations. Also note the relationship to invariance: if we use image descriptors based at the keypoints for matching, the more invariant the descriptors are to a given type of transformation, the less accurate the keypoint detection needs to be with respect to these transformations. But exactly for this reason, it is useful to develop detectors whose performance under different types of transformations is quantifiable and controllable, and our approach explicitly does this. We adopt the following basic philosophy:

(*i*) *There is no such thing as generic keypoints.* They should be selected specifically for the use to which they will be put, using a purpose-designed detector and parameters.

(*ii*) *Keypoints are not just positions.* Stability in orientation and scale and resistance to common types of appearance variations are also needed.

(*iii*) *Each image (template) matching method defines a corresponding self-matching based keypoint detector.* If the keypoints will be used as correspondence hypotheses that are later verified by inter-image template matching, the keypoint detector and parameters corresponding to the matching method should be used.

Contents: §2 describes our matching based framework for keypoint detection. §3 gives some specific examples and implementation details. §4 gives a few experimental results.

Notation: $\boldsymbol{x}$ stands for image coordinates, $\boldsymbol{\nabla}$ for $\boldsymbol{x}$-derivatives, I, R for the images being matched (treated as functions of $\boldsymbol{x}$), $\boldsymbol{t}$ for the image motion/warping model, c for the pixel comparison functional. Derivatives are always row vectors, *e.g.* $\delta I \approx \boldsymbol{\nabla} I \, \boldsymbol{\delta x}$. For most of the paper we assume continuous images and ignore sampling issues.

2 General Framework

This section develops a general framework for robust image (template) matching under analytical image deformation and appearance variation models, uses it to derive stability estimates for locally optimal matches, and applies this to characterize keypoint stability under self-matching.

Template matching model: We will use the following generalized error model for template matching, explained element-by-element below:

$$Q(\boldsymbol{\mu}, \boldsymbol{\lambda}) \;\equiv\; \int c\left(I(\boldsymbol{t}(\boldsymbol{x}, \boldsymbol{\mu}), \boldsymbol{\lambda}), R(\boldsymbol{x}), \boldsymbol{x}\right) \, \boldsymbol{dx} \tag{1}$$

I is the image patch being matched, R is the reference patch it is being matched against, $\boldsymbol{x}$ is a set of 2D image coordinates centred on R, and $c \geq 0$ (discussed further below) is a weighted image pixel comparison functional that is integrated over the patch to find the overall matching quality metric Q. $\boldsymbol{x}' = \boldsymbol{t}(\boldsymbol{x}, \boldsymbol{\mu})$ is an image motion / warping model that maps R's coordinates $\boldsymbol{x}$ forwards into I's natural coordinate system, *i.e.*, I is effectively being pulled back (warped backwards) into R's frame before being compared. The motion model $\boldsymbol{t}$ is controlled by a vector of **motion parameters** $\boldsymbol{\mu}$ (2D translation, perhaps rotation, scaling, affine deformation…). Before being compared, I may also undergo an optional appearance correction controlled by a vector of **appearance parameters** $\boldsymbol{\lambda}$ (*e.g.*, luminance or colour shifts/rescalings/normalizations, corrections for local illumination gradients…). Note that we think of the input patch I as an ad hoc function $I(\boldsymbol{x}, \boldsymbol{\lambda})$ of both the position and appearance parameters, rather than as a fixed image $I(\boldsymbol{x})$ to which separate appearance corrections are applied. This allows the corrections to be image-content dependent and nonlocal within the patch (*e.g.* subtracting the mean in Zero Mean Cross Correlation). We assume that $\boldsymbol{\mu} = \boldsymbol{0}$ represents a neutral position or reference transformation for the patch (*e.g.* no motion, $\boldsymbol{t}(\boldsymbol{x}, \boldsymbol{0}) = \boldsymbol{x}$). Similarly, $\boldsymbol{\lambda} = \boldsymbol{0}$ represents a default or reference appearance setting (*e.g.* the unchanged input, $I(\boldsymbol{x}, \boldsymbol{0}) = I(\boldsymbol{x})$).

The patch comparison integral is over a spatial window centred on R, but for compactness we encode this in the pixel comparison metric c. So c usually has the form:

$$c(I(\boldsymbol{x}), R(\boldsymbol{x}), \boldsymbol{x}) \;\equiv\; w(\boldsymbol{x}) \cdot \rho(I(\boldsymbol{x}), R(\boldsymbol{x})) \tag{2}$$

where $w(\boldsymbol{x})$ is a spatial windowing function (rectangular, Gaussian…) that defines the extent of the relevant patch of R, and $\rho(I(\boldsymbol{x}), R(\boldsymbol{x}))$ is a spatially-invariant image pixel comparison metric, *e.g.*, the squared pixel difference $\|I(\boldsymbol{x}) - R(\boldsymbol{x})\|^2$ for traditional unweighted least squares matching. The "pixels" here may be greyscale, colour,

multi-band, or even pre-extracted edge, feature or texture maps, so $\rho()$ can be quite complicated in general, *e.g.* involving nonlinear changes of luminance or colour space, perceptual or sensitivity-based comparison metrics, robust tailing-off at large pixel differences to reduce the influence of outliers, *etc.* Ideally, $\rho()$ should return the negative log likelihood for the pixels to correspond, so that (assuming independent noise in each pixel) Q becomes the total negative log likelihood for the patchwise match. For practical inter-image template matching, the reliability depends critically on the robustness (large difference behaviour) of $\rho()$. But for keypoint detection, we always start from the self-matching case $I{=}R$, so only the *local* behaviour of $\rho()$ near $I{=}R$ is relevant: keypoint detectors are oblivious to large-difference robustification of $\rho()$. We will assume that $\rho()$ has least-squares-like behaviour for small pixel differences, *i.e.* that it is locally differentiable with zero gradient and positive semi-definite Hessian at $I{=}R$, so that:

$$\frac{\boldsymbol{\delta} c}{\boldsymbol{\delta} I(\boldsymbol{x})}\Big|_{I=R} = \boldsymbol{0}, \qquad \frac{\boldsymbol{\delta}^2 c}{\boldsymbol{\delta} I(\boldsymbol{x})^2}\Big|_{I=R} \geq \boldsymbol{0} \tag{3}$$

Our derivations will be based on 2^{nd} order Taylor expansion at $I{=}R$, so they exclude both non-differentiable L_1 matching metrics like Sum of Absolute Differences (SAD) and discontinuous L_0 (on-off) style ones. Our overall approach probably extends to such metrics, at least when used within a suitable interpolation model, but their abrupt changes and weak resampling behaviour make general derivations difficult.

Finally, we allow c to be a *functional*, not just a function, of I, R. (*I.e.* a function of the local patches, not just their pointwise pixel values). In particular, c may run I, R through convolutional filters (‘**prefilters**’) before comparing them, *e.g.* to restrict attention to a given frequency band in scale-space matching, or simply to suppress high frequencies for reduced aliasing and/or low frequencies for better resistance to global illumination changes. In general, the resampling implied by $\boldsymbol{t}()$ could significantly change I’s spatial frequency content, so prefiltering only makes sense if we do it *after* warping. We will thus assume that prefilters run in $\boldsymbol{x}$-space, *i.e.* they are defined relative to the coordinates of the reference image R. For example, for affine-invariant keypoint detection [32, 24, 25], keypoint comparison should typically be done, and in particular prefiltering should be applied, in the characteristic affine-normalized frame of the reference keypoint, so $\boldsymbol{x}$ would typically be taken to be the affine-normalized coordinates for R. For any $\boldsymbol{t}()$, derivatives of the unwarped input image I can always be converted to derivatives of its prefilter using integration by parts, so the effective scale of derivative masks always ends up being the $\boldsymbol{x}$-space scale of the prefilter.

Matching precision: Now suppose that we have already found a locally optimal template match. Consider the behaviour of the matching quality metric Q under small perturbations $I \rightarrow I{+}\delta I$. Under 2^{nd} order Taylor expansion:

$$\delta Q \approx \int \left(\frac{\boldsymbol{\delta} c}{\boldsymbol{\delta} I}\, \delta I + \tfrac{1}{2}\, \delta I^{\top}\, \frac{\boldsymbol{\delta}^2 c}{\boldsymbol{\delta} I^2}\, \delta I \right)_{\boldsymbol{x}'=\boldsymbol{t}(\boldsymbol{x})} \boldsymbol{dx} \tag{4}$$

For any perturbation of an exact match, $I(\boldsymbol{t}(\boldsymbol{x})) = R(\boldsymbol{x})$, the first order ($\delta I$) term vanishes identically by (3). More generally, if we are already at a local optimum of Q under some class of perturbations δI, the integrated first order term vanishes for this class. Both hold for keypoints, so we will ignore the δI term from now on.

Using the parametric model $I(\boldsymbol{t}(\boldsymbol{x}, \boldsymbol{\mu}), \boldsymbol{\lambda})$, the image I changes as follows under first order changes of the motion and appearance parameters $\boldsymbol{\mu}, \boldsymbol{\lambda}$:

$$\delta I \approx \boldsymbol{L}\, \boldsymbol{\delta\lambda} + \boldsymbol{M}\, \boldsymbol{\delta\mu}\,, \quad \text{where} \quad \boldsymbol{L} \equiv \frac{\boldsymbol{\partial} I}{\boldsymbol{\partial\lambda}}, \quad \boldsymbol{M} \equiv \boldsymbol{\nabla} I \cdot \boldsymbol{T}, \quad \boldsymbol{T} \equiv \frac{\boldsymbol{\partial t}}{\boldsymbol{\partial\mu}} \tag{5}$$

Here, $\boldsymbol{\nabla} I \equiv \frac{\partial I}{\partial \boldsymbol{t}}(\boldsymbol{t}(\boldsymbol{x}))$ is the standard gradient of the original unwarped image I, evaluated in I's own frame at $\boldsymbol{t}(\boldsymbol{x})$. The columns of the Jacobians $\boldsymbol{L}$ and $\boldsymbol{M}$ can be thought of as appearance and motion basis images, characterizing the linearized first-order changes in I as the parameters are varied. Putting (4, 5) together gives a quadratic local cost model for perturbations of the match around the optimum, based on a positive semidefinite **generalized scatter matrix** $\mathbf{S}$:[1]

$$\delta Q(\boldsymbol{\delta\lambda}, \boldsymbol{\delta\mu}) \;\approx\; \tfrac{1}{2} \begin{pmatrix} \boldsymbol{\delta\lambda}^\top & \boldsymbol{\delta\mu}^\top \end{pmatrix} \mathbf{S} \begin{pmatrix} \boldsymbol{\delta\lambda} \\ \boldsymbol{\delta\mu} \end{pmatrix} \tag{6}$$

$$\mathbf{S} \;\equiv\; \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^\top & \boldsymbol{C} \end{pmatrix} \;\equiv\; \int \begin{pmatrix} \boldsymbol{L}^\top \\ \boldsymbol{M}^\top \end{pmatrix} \frac{\boldsymbol{\delta}^2 c}{\boldsymbol{\delta} I^2} \begin{pmatrix} \boldsymbol{L} & \boldsymbol{M} \end{pmatrix} \boldsymbol{dx} \tag{7}$$

$\mathbf{S}$ generalizes the matrix $\int \boldsymbol{\nabla} I^\top \boldsymbol{\nabla} I \, \boldsymbol{dx}$ that appears in the Förstner-Harris keypoint detector (which assumes pure translation, $\boldsymbol{T} = \boldsymbol{I}, \boldsymbol{M} = \boldsymbol{\nabla} I$, quadratic pixel difference metric $\frac{\boldsymbol{\delta}^2 c}{\boldsymbol{\delta} I^2} = \boldsymbol{I}$, and empty illumination model $\boldsymbol{L}$). To the extent that c gives the negative log likelihood for the match, $\mathbf{S}$ is the maximum likelihood saddle point approximation to the Fisher information matrix for estimating $\boldsymbol{\lambda}, \boldsymbol{\mu}$ from the match. *I.e.*, $\mathbf{S}^{-1}$ approximates the covariance with which the parameters $\boldsymbol{\lambda}, \boldsymbol{\mu}$ can be estimated from the given image data: the larger $\mathbf{S}$, the stabler the match, in the sense that the matching error δQ increases more rapidly under given perturbations $\boldsymbol{\delta\lambda}, \boldsymbol{\delta\mu}$.

Now suppose that we want to ensure that the two patches match stably *irrespective of appearance changes*. For a given perturbation $\boldsymbol{\delta\mu}$, the appearance change that gives the best match to the original patch — and hence that masks the effect of the motion as well as possible, thus creating the greatest matching uncertainty — can be found by minimizing $\delta Q(\boldsymbol{\delta\mu}, \boldsymbol{\delta\lambda})$ w.r.t. $\boldsymbol{\delta\lambda}$. By inspection from (6), this is $\boldsymbol{\delta\lambda}(\boldsymbol{\delta\mu}) = -\boldsymbol{A}^{-1}\boldsymbol{B}\,\boldsymbol{\delta\mu}$. Back-substituting into (6) gives an effective quadratic **reduced penalty function** $\delta Q_{\text{red}}(\boldsymbol{\delta\mu}) \;\equiv\; \delta Q(\boldsymbol{\delta\mu}, \boldsymbol{\delta\lambda}(\boldsymbol{\delta\mu})) \;\approx\; \frac{1}{2}\boldsymbol{\delta\mu}^\top \boldsymbol{C}_{\text{red}}\,\boldsymbol{\delta\mu}$ characterizing motion-with-best-appearance-adaptation, where the **reduced scatter matrix** is

$$\boldsymbol{C}_{\text{red}} \;\equiv\; \boldsymbol{C} - \boldsymbol{B}^\top \boldsymbol{A}^{-1} \boldsymbol{B} \tag{8}$$

with $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ as in (7). $\boldsymbol{C}_{\text{red}}$ and $\boldsymbol{C}$ quantify the precision of motion estimation respectively with and without appearance adaptation. Some precision is always lost by factoring out appearance, so $\boldsymbol{C}_{\text{red}}$ is always smaller than $\boldsymbol{C}$. To the extent that the matching error metric c is a statistically valid log likelihood model for image noise, $\boldsymbol{C}^{-1}$ and $\boldsymbol{C}_{\text{red}}^{-1}$ estimate the covariances of the corresponding motion parameter estimates under trials with independent noise samples. More generally, if we also have prior information that appearance variations are not arbitrary, but have zero mean and covariance $\boldsymbol{D}^{-1}$, the optimal $\boldsymbol{\delta\lambda}(\boldsymbol{\delta\mu})$ becomes $-(\boldsymbol{A}+\boldsymbol{D})^{-1}\boldsymbol{B}\,\boldsymbol{\delta\mu}$ and $\boldsymbol{C}_{\text{red}}$ is replaced by the less strongly reduced covariance $\boldsymbol{C}'_{\text{red}} \;\equiv\; \boldsymbol{C} - \boldsymbol{B}^\top(\boldsymbol{A}+\boldsymbol{D})^{-1}\boldsymbol{B}$.

Keypoint detection: Ideally, we want to find keypoints that can be *stably* and *reliably* re-detected under arbitrary motions from the given transformation family $\boldsymbol{t}(\boldsymbol{x}, \boldsymbol{\mu})$, despite arbitrary changes of appearance from the appearance family $I(\boldsymbol{x}, \boldsymbol{\lambda})$. We focus on the 'stability' aspect[2], which we characterize in terms of the *precision of self-matching* under our robust template matching model. The idea is that the patch itself is its own

[1] Strictly, to be correct to $\mathcal{O}\big((\boldsymbol{\delta\mu}, \boldsymbol{\delta\lambda})^2\big)$ we should also expand (5) to 2nd order, which introduces a 2nd order 'tensor' correction in the δI term of (4). But, as above by (3), the latter term vanishes identically for keypoint detection. Even for more general matching, the correction is usually negligible unless the match is poor *and* the motion / appearance models are very nonlinear. One can think of (7) as a Gauss-Newton approximation to the true $\mathbf{S}$. It guarantees that $\mathbf{S}$ is at least positive semidefinite (as it must be at a locally optimal match). We will adopt it from now on.

[2] We do not consider other matchability properties [7] such distinctiveness here, as this is more a matter for

best template — if it can not be matched stably even against itself, it is unlikely to be stably matchable against other patches. We are interested in stability despite appearance changes, so we use the reduced scatter matrix $\boldsymbol{C}_{\text{red}}$ (8) to quantify geometric precision.

The amount of precision that is needed depends on the task, and we adopt the design philosophy that visual routines should be explicitly parametrized in terms of objective performance criteria such as output accuracy. To achieve this we require keypoints to meet a lower bound on matching precision (equivalently, an upper bound on matching uncertainty). We quantify this by introducing a user-specified **criterion matrix** $\boldsymbol{C}_0$ and requiring keypoints to have reduced precisions $\boldsymbol{C}_{\text{red}}$ greater than $\boldsymbol{C}_0$ (*i.e.* $\boldsymbol{C}_{\text{red}} - \boldsymbol{C}_0$ must be positive semidefinite). Intuitively, this means that for a keypoint candidate to be accepted, its transformation-space motion-estimation uncertainty ellipse $\boldsymbol{C}_{\text{red}}^{-1}$ must be strictly contained within the criterion ellipse $\boldsymbol{C}_0^{-1}$.

In textured images there may be whole regions where this precision criterion is met, so for isolated keypoint detection we must also specify a means of selecting 'the best' keypoint(s) within these regions. This requires some kind of 'saliency' or 'interest' metric, ideally an index of perceptual distinctiveness / reliable matchability modulo our appearance model. But here, following the Förstner-Harris philosophy, we simply use an index of overall matching precision as a crude substitute for this. In the translation-only case, Förstner [7,9] and Harris & Stephens [15] discuss several suitable precision indices, based on the determinant, trace and eigenvalues of the scatter matrix. In our case, there may be several (more than 2) motion parameters, and eigenvalue based criteria seem more appropriate than determinant based ones, owing to their clear links with uncertainty analysis. Different motion parameters also have different units (translations in pixels, rotations in radians, dilations in log units), and we need to normalize for this. The criterion matrix $\boldsymbol{C}_0$ provides a natural scaling, so as our final saliency criterion we will take the *minimum eigenvalue of the normalized reduced motion precision matrix* $\boldsymbol{C}_0^{-1/2}\,\boldsymbol{C}_{\text{red}}\,\boldsymbol{C}_0^{-1/2}$. Intuitively, this requires the longest axis of the motion-estimation covariance ellipse, as measured in a frame in which $\boldsymbol{C}_0$ becomes spherical, to be as small as possible. With this normalization, the keypoint-acceptability criterion $\boldsymbol{C}_{\text{red}} > \boldsymbol{C}_0$ simplifies to the requirement that the saliency (the minimum eigenvalue) must be greater than one. Typically, $\boldsymbol{C}_0$ is diagonal, in which case the normalization matrix $\boldsymbol{C}_0^{-1/2}$ is the diagonal matrix of maximum user-permissible standard errors in translation, rotation and scale.

As usual, pixel sampling effects introduce a small amount of aliasing or jitter in the image derivative estimates, which has the effect of spreading gradient energy across the various eigenvalues of $\mathbf{S}$ even when the underlying image signal is varies only in one dimension (*e.g.* a straight edge). As in the Förstner-Harris case, we compensate for this heuristically by subtracting a small user-specified multiple α of the maximum eigenvalue of $\boldsymbol{C}_0^{-1/2}\,\boldsymbol{C}_{\text{red}}\,\boldsymbol{C}_0^{-1/2}$ (the 1-D 'straight edge' signal) before testing for threshold and saliency, so our final keypoint saliency measure is $\lambda_{\min} - \alpha\,\lambda_{\max}$.

In practice, the Schur complement in $\boldsymbol{C}_{\text{red}} = \boldsymbol{C} - \boldsymbol{B}^{\top}\boldsymbol{A}^{-1}\boldsymbol{B}$ is calculated simply and efficiently by outer-product based partial Cholesky decomposition. A standard symmetric eigendecomposition method is then used to calculate the minimum eigenvalue, except that 2D eigenproblems are handled as a special case for speed.

---

the descriptors calculated once the keypoint is found. Distinctiveness is usually characterized by probability of mismatch within a population of extracted keypoints (*e.g.* [33]). For a recent entropic approach to image-wide distinctiveness, see [36].

3 Examples of Keypoint Detectors

Given the above framework, it is straightforward to derive keypoint detectors for specific pixel types and motion and appearance models. Here we only consider the simplest few motion and appearance models, and we assume greyscale images.

Comparison function: As in the traditional Harris detector, we will use simple squared pixel difference to compare pixels, and a circular Gaussian spatial integration window. So modulo prefiltering, $\frac{\delta^2 c}{\delta I^2}$ in (7) reduces to simple weighting by the window function.

Affine deformations: For keypoints, only local deformations are relevant, so the most general motion model that is useful is probably the affine one. We will use various subsets of this, parametrizing affine motions linearly as $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{T}\,\boldsymbol{\mu}$ where:

$$\boldsymbol{T}\,\boldsymbol{\mu} = \begin{pmatrix} 1 & 0 & -y & x & x & y \\ 0 & 1 & x & y & -y & x \end{pmatrix} \begin{pmatrix} u \\ v \\ r \\ s \\ a \\ b \end{pmatrix} = \begin{pmatrix} 1{+}s{+}a & -r{+}b \\ r{+}b & 1{+}s{-}a \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} u \\ v \end{pmatrix} \tag{9}$$

Here, (x, y) are window-centred pixel coordinates, (u, v) is the translation, s the scale, and for small motions, r is the rotation and a, b are axis- and $45°$-aligned quadrupole deformations. The resulting $\boldsymbol{M}$ matrix is as follows, where $\boldsymbol{\nabla} I = (I_x, I_y)$:

$$\boldsymbol{M} = \begin{pmatrix} I_x & I_y & -yI_x{+}xI_y & xI_x{+}yI_y & xI_x{-}yI_y & yI_x{+}xI_y \end{pmatrix} \tag{10}$$

If the input image is being prefiltered (which, as discussed, must happen *after* warping, *i.e.* after (10)), we can integrate by parts to reduce the prefiltered $\boldsymbol{M}$ vector to the form:

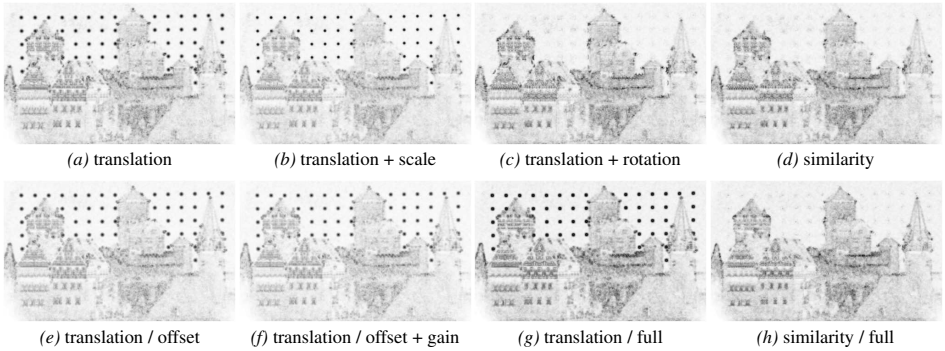
$$\boldsymbol{M}^p = \begin{pmatrix} I^p_x,\ I^p_y,\ -(yI)^p_x{+}(xI)^p_y,\ (xI)^p_x{+}(yI)^p_y{-}2I^p,\ (xI)^p_x{-}(yI)^p_y,\ (yI)^p_x{+}(xI)^p_y \end{pmatrix} \tag{11}$$

where $I^p \equiv p * I$, $(xI)^p_y \equiv p_y * (xI)$, *etc.*, denote convolutions of I, xI, *etc.*, against the prefilter p and its derivatives p_x, p_y. The $-2I^p$ term in the s entry corrects for the fact that prefiltering should happen after any infinitessimal scale change coded by $\boldsymbol{M}$: without this, we would effectively be comparing patches taken at different derivative scales, and would thus overestimate the scale localization accuracy. If p is a Gaussian of width σ, we can use (10) or (11) and the corresponding identities $(xI)^p = xI^p + \sigma^2 I^p_x$ or $(xI)^p_x = x\,I^p_x + \sigma^2 I^p_{xx} + I^p$ (from $(x{-}x')g(x{-}x') = -\sigma^2 g_x(x{-}x')$, *etc.*) to move x, y outside the convolutions, reducing $\boldsymbol{M}^p$ to:

$$\begin{pmatrix} I^p_x,\ I^p_y,\ -yI^p_x{+}xI^p_y,\ xI^p_x{+}yI^p_y{+}\sigma^2 I^p_{xx+yy},\ xI^p_x{-}yI^p_y{+}\sigma^2 I^p_{xx-yy},\ yI^p_x{+}xI^p_y + 2\sigma^2 I^p_{xy} \end{pmatrix} \tag{12}$$

Appearance model: Class-specific appearance models like [1, 13] can include elaborate models of appearance variation, but for generic keypoint detection we can only use simple generic models designed to improve resistance to common types of local illumination variations. Here, we allow for (at most) a scalar illumination shift, addition of a constant spatial illumination gradient, and illumination rescaling. So our linear appearance model is $I{+}\boldsymbol{L}\,\boldsymbol{\lambda}$ where $\boldsymbol{L}(\boldsymbol{x})$ is a subset of:

$$\boldsymbol{L}(\boldsymbol{x}) = \begin{pmatrix} 1 & x & y & I(\boldsymbol{x}) \end{pmatrix} \tag{13}$$



(a) translation *(b)* translation + scale *(c)* translation + rotation *(d)* similarity

(e) translation / offset *(f)* translation / offset + gain *(g)* translation / full *(h)* similarity / full

Figure 1: Minimum-eigenvalue strength maps for a popular test image under various motion and illumination models. The saliency differences are much larger than they seem: the maps have been very strongly gamma compressed, normalized and inverted for better visibility. The prefilter and integration windows had σ=1 pixel, and $\alpha = 0$. Criterion standard deviations were 1 pixel in translation, 1 radian in rotation, $\sqrt{2}$ in scale, but these values are not critical.

As with $\boldsymbol{M}$, the elements of $\boldsymbol{L}$ must be prefiltered, but I is just smoothed to I^p and $1, x, y$ typically have trivial convolutions (*e.g.*, they are unchanged under Gaussian smoothing, and hence generate a constant diagonal block $\mathrm{diag}(1, \sigma_w^2, \sigma_w^2)$ in $\mathbf{S}$).

Putting it all together: The main stages of keypoint detection are: (*i*) prefilter the input image to produce the smoothed image and derivative estimates $I^p, I^p_x, I^p_y, I^p_{xx}, I^p_{xy}, I^p_{yy}$ needed for (12, 13); (*ii*) for each keypoint location $\boldsymbol{x}$, form the outer product matrix of the (desired components of the) combined L/M vector at all pixels in its window, and sum over the window to produce the scatter matrix $\mathbf{S}(\boldsymbol{x})$ (7) (use window-centred coordinates for x, y in (12, 13)); (*iii*) at each $\boldsymbol{x}$, reduce $\mathbf{S}(\boldsymbol{x})$ to find $\boldsymbol{C}_{\mathrm{red}}(\boldsymbol{x})$, normalize by $\boldsymbol{C}_0$, and find the smallest eigenvalue (saliency). Keypoints are declared at points where the saliency has a dominant local maximum, *i.e.* is above threshold and larger than at all other points within a suitable non-maximum-suppression radius. For multiscale detection, processing is done within a pyramid and keypoints must be maxima in both position and scale. As usual, one can estimate subpixel keypoint location and scale by quadratic interpolation of the saliency field near its maximum. But note that, as in the standard Förstner-Harris approach, keypoints do not necessarily contain nameable features (corners, spots) that clearly mark their centres — they may just be unstructured patches with locally maximal matching stability[3].

When calculating $\mathbf{S}$, instead of separate *ab initio* summation over each integration window, one can also use image-wide convolution of quadratic 'energies' as in the standard Förstner-Harris detector, but for the more complicated detectors there are many such maps to be calculated (76 for the full 10-entry L/M model). See the extended version of this paper for details.

In our current implementation, run times for the full 10-L/M-variable detector (which is more than one would normally use in practice) are a factor of about 10 larger than for the original two variable Förstner-Harris detector.

[3]If well-localized centres are needed, specialized locators exist for specific image structures such as spots and corners (*e.g.* [8]), or more generally one could search for *sharp* (high-curvature) and preferably *isolated* maxima of the minimum eigenvalue field or local saliency measure, not just for *high* (but possibly broad) ones. For example, a minimum acceptable peak curvature could be specified via a second criterion matrix.

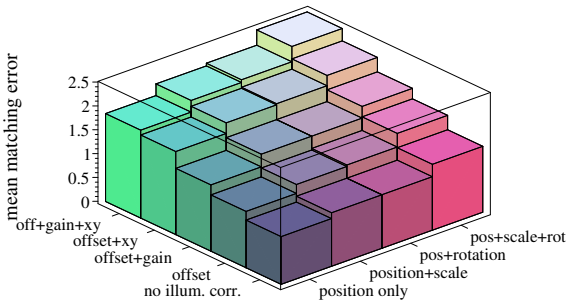


Figure 2: Mean predicted standard error (inverse square root of saliency / minimum eigenvalue in normalized units) for template matching of keypoints under our motion and lighting models, for the model's top 100 keypoints on the Summer Palace image in fig. 3.

Relation to Zero Mean Matching: This common matching method compares two image patches by first subtracting each patches mean intensity, then summing the resulting squared pixel differences. We can relate this to the simplest nonempty illumination correction model, $\boldsymbol{L}=(1)$, whose reduced scatter matrix over window $w(\boldsymbol{x})$ is:

$$\boldsymbol{C}_{\text{red}} \;=\; \int w\,\boldsymbol{M}^{\top}\boldsymbol{M}\,\boldsymbol{dx} - \overline{\boldsymbol{M}}^{\top}\overline{\boldsymbol{M}} \;=\; \int w\,(\boldsymbol{M}-\overline{\boldsymbol{M}})^{\top}(\boldsymbol{M}-\overline{\boldsymbol{M}})\,\boldsymbol{dx}$$
$$\overline{\boldsymbol{M}} \;\equiv\; \textstyle\int w\,(\boldsymbol{M})\,\boldsymbol{dx} \;/\; \left(\int w\,\boldsymbol{dx}\right)^{1/2} \tag{14}$$

For the translation-only model, $\boldsymbol{T}$ is trivial, so the illumination correction simply has the effect of subtracting from each image gradient its patch mean (*c.f.* (10)). If w changes much more slowly than I, $\overline{\boldsymbol{\nabla} I} \approx \boldsymbol{\nabla}\overline{I}$ and hence $\boldsymbol{\nabla} I - \overline{\boldsymbol{\nabla} I} \approx \boldsymbol{\nabla}(I-\overline{I})$, so this is approximately the same as using the gradient of the bandpassed image $I-\overline{I}$. The standard Förstner-Harris detector embodies least squares matching, not zero mean matching. It is invariant to constant illumination shifts, but it does not subtract the gradient of the mean $\boldsymbol{\nabla}\overline{I}$ (or more correctly, the mean of the gradient $\overline{\boldsymbol{\nabla} I}$) to discount the effects of smooth local illumination gradients superimposed on the pattern being matched. It thus systematically overestimates the geometric strength of keypoints in regions with strong illumination gradients, *e.g.* near the borders of smoothly shaded objects, or at the edges of shadows.

4 Experiments

Fig. 1 shows that the saliency (minimum eigenvalue) map emphasizes different kinds of image structures as the motion and illumination models are changed. Image (a) is the original Förstner-Harris detector. Images $(b),(c),(d)$ successively add scale, rotation and scale + rotation motions, while images $(e),(f),(g)$ adjust for illumination offset, offset + gain, and offset + gain + spatial gradients. Note the dramatic extent to which enforcing rotational stability in $(a)\rightarrow(c)$ and $(b)\rightarrow(d)$ eliminates the circular dots of the calibration pattern. In comparison, enforcing scale stability in $(a)\rightarrow(b)$ and $(c)\rightarrow(d)$ has more subtle effects, but note the general relative weakening of the points at the summits of the towers between (a) and (b): straight-edged 'corners' are scale invariant, and are therefore suppressed. Unfortunately, although ideal axis- and $45°$-aligned corners are strongly suppressed, it seems that aliasing and blurring effects destroy much of the notional scale invariance of most other rectilinear corners, both in real images and in

non-axis-aligned ideal ones. We are currently working on this problem, which also reduces the cross-scale performance of the standard Förstner-Harris detector.

Adding illumination invariance seems to have a relatively small effect in this example, but note the general relative sharpening caused by including x and y illumination gradients in $(a), (e), (f) \rightarrow (g)$. Points on the borders of intensity edges have enhanced gradients owing to the slope alone, and this tends to make them fire preferentially despite the use of the minimum-eigenvalue (most uncertain direction) criterion. Subtracting the mean local intensity gradient reduces this and hence sharpens the results. However a negative side effect of including x, y gradients is that locally quadratic image patches — in particular small dots and ridge edges — become much less well localized, as adding a slope to a quadratic is equivalent to translating it.

Allowing more general motions and/or quotienting out illumination variations always reduces the precision of template matching. Fig. 2 shows the extent of this effect by plotting the relative standard errors of template matching for our complete set of motion and lighting models, where the matching for each model is performed on the model's own keypoints. There is a gradual increase in uncertainty as parameters are added, the final uncertainty for a similarity transform modulo the full illumination model being about 2.5 times that of the original translation-only detector with no illumination correction.

Fig. 3 shows some examples of keypoints selected using the various different motion/lighting models. The main observation is that different models often select different keypoints, and more invariant models generate fewer of them, but beyond this it is difficult to find easily interpretable systematic trends. As in the Förstner-Harris case, keypoints are optimized for matching precision, not for easy interpretability in terms of idealized image events.

5 Summary and Conclusions

Summary: We have generalized the Förstner-Harris detector [7,9,15] to select keypoints that provide repeatable scale and orientation, as well as repeatable position, over re-detections, even in the face of simple local illumination changes. Keypoints are selected to maximize a minimum-eigenvalue-based local stability criterion obtained from a second order analysis of patch self-matching precision under affine image deformations, compensated for linear illumination changes.

Future work: The approach given here ensures accurate re-localizability (by inter-image template matching) of keypoint image patches under various transformations, but it does not always provide accurate 'centres' for them. To improve this, we would like to characterize the stability and localization accuracy of the local maxima of the saliency measure (minimum eigenvalue) under the given transformations. In other words, just as we derived the local transformational-stability matrix $\boldsymbol{C}_{\text{red}}(\boldsymbol{x})$ for *matching* from the scalar matching metric $Q(\boldsymbol{x})$, we need to derive a local transformational-stability matrix for *saliency* from the scalar saliency metric. Only here, the saliency measure is already based on matching stability, so a second level of analysis will be needed.

References

[1] S. Baker, S. Nayar, and H. Murase. Parametric feature detection. *Int. J. Computer Vision*, 27(1):27–50, 1998.

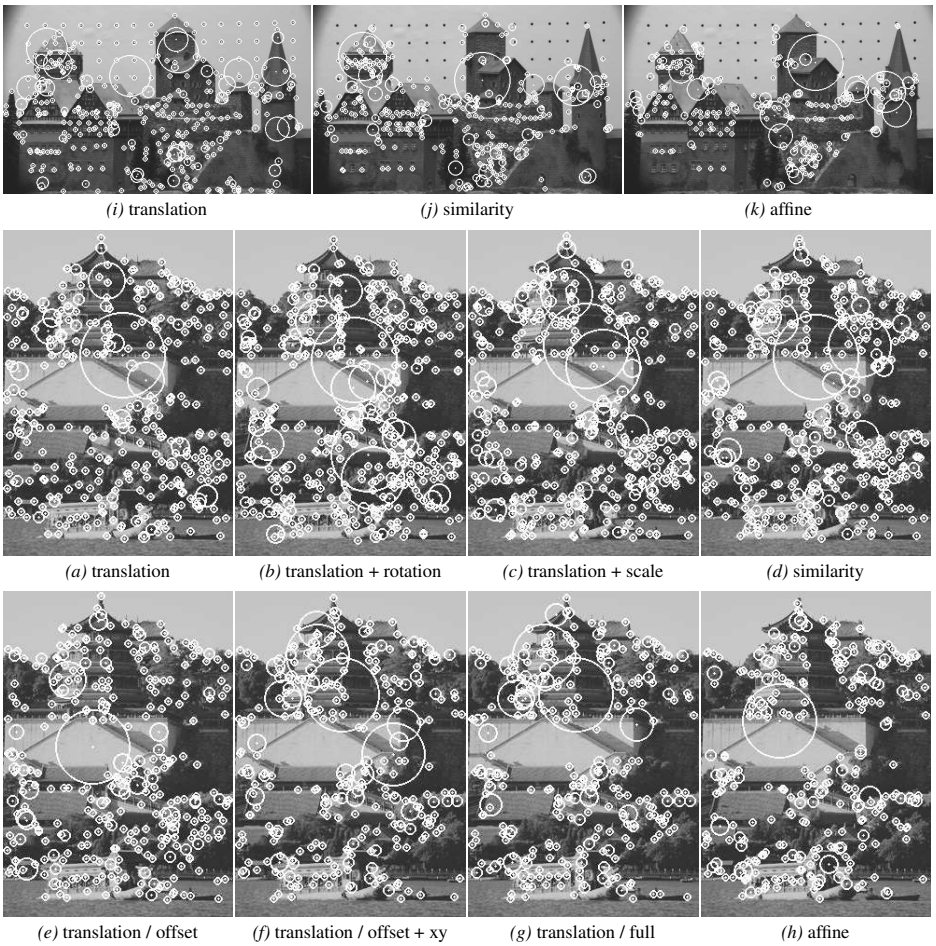


Figure 3: Examples of keypoints from the CMU and Summer Palace (Beijing) test images, under various motion and illumination models. The prefilter and integration windows had σ=2 pixels, $\alpha = 0$, and non-maximum suppression within 4 pixels radius and scale factor 1.8 was applied. Note that, *e.g.*, 'affine' means 'resistant to small affine deformations', not affine invariant in the sense of [32, 24, 25].

[2] P.R. Beaudet. Rotationally invariant image operators. In *Int. Conf. Pattern Recognition*, pages 579–583, 1978.

[3] R. Deriche and T. Blaszka. Recovering and characterizing image features using an efficient model based approach. In *Int. Conf. Computer Vision & Pattern Recognition*, pages 530–535, 1993.

[4] R. Deriche and G. Giraudon. A computational approach for corner and vertex detection. *Int. J. Computer Vision*, 10(2):101–124, 1993.

[5] O. Faugeras, Q-T. Luong, and T. Papadopoulo. *The Geometry of Multiple Images*. MIT Press, 2001.

[6] W. Förstner. On the geometric precision of digital correlation. *Int. Arch. Photogrammetry & Remote Sensing*, 24(3):176–189, 1982.

[7] W. Förstner. A feature-based correspondence algorithm for image matching. *Int. Arch. Photogrammetry & Remote Sensing*, 26 (3/3):150–166, 1986.

[8] W. Förstner. A framework for low-level feature extraction. In *European Conf. Computer Vision*, pages II 383–394, Stockholm, 1994.

[9] W. Förstner and E. Gülch. A fast operator for detection and precise location of distinct points, corners and centres of circular features. In *ISPRS Intercommission Workshop*, Interlaken, June 1987.

[10] A. Grün. Adaptive least squares correlation — concept and first results. Intermediate Research Report to Helava Associates, Ohio State University. 13 pages, March 1984.

[11] A. Grün. Least squares matching: A fundamental measurement algorithm. In *Close Range Photogrammetry and Machine Vision*, pages 217–255. Whittles Publishing, Caithness, Scotland, 1996.

[12] A. Grün and E.P. Baltsavias. Adaptive least squares correlation with geometrical constraints. In *SPIE Computer Vision for Robots*, volume 595, pages 72–82, Cannes, 1985.

[13] G. Hager and P. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 20(10):1025–1039, 1998.

[14] M.J. Hannah. *Computer Matching of Areas in Stereo Images*. Ph.D. Thesis, Stanford University, 1974. AIM Memo 219.

[15] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 147–151, 1988.

[16] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[17] L. Kitchen and A. Rosenfeld. Gray-level corner detection. *Patt. Rec. Lett.*, 1:95–102, 1982.

[18] P. Kovesi. Image features from phase congruency. *Videre: A Journal of Computer Vision Research*, 1(3), 1999.

[19] R. Laganière. Morphological corner detection. In *Int. Conf. Computer Vision*, pages 280–285, 1998.

[20] D. Lowe. Object recognition from local scale-invariant features. In *Int. Conf. Computer Vision*, pages 1150–1157, 1999.

[21] D. Lowe. Local feature view clustering for 3d object recognition. In *Int. Conf. Computer Vision & Pattern Recognition*, pages 682–688, 2001.

[22] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981.

[23] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Int. Conf. Computer Vision*, pages 525–531, 2001.

[24] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *European Conf. Computer Vision*, pages I.128–142, 2002.

[25] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In *Int. Conf. Computer Vision & Pattern Recognition*, 2003.

[26] H.P. Moravec. Towards automatic visual obstacle avoidance. In *IJCAI*, page 584, 1977.

[27] M. C. Morrone and R. A. Owens. Feature detection from local energy. *Patt. Rec. Lett.*, 6:303–313, 1987.

[28] J.A. Noble. Finding corners. *Image & Vision Computing*, 6(2):121–128, 1988.

[29] D. Reisfeld. The constrained phase congruency feature detector: Simultaneous localization, classification, and scale determination. *Patt. Rec. Lett.*, 17:1161–1169, 1996.

[30] B. Robbins and R. Owens. 2d feature detection via local energy. *Image & Vision Computing*, 15:353–368, 1997.

[31] K. Rohr. Localization properties of direct corner detectors. *J. Mathematical Imaging & Vision*, 4(2):139–150, 1994.

[32] F. Schaffalitzky and A. Zisserman. Viewpoint invariant texture matching and wide baseline stereo. In *Int. Conf. Computer Vision*, pages 636–643, Vancouver, 2001.

[33] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *Int. J. Computer Vision*, 37(2):151–172, 2000.

[34] J. Shi and C. Tomasi. Good features to track. In *Int. Conf. Computer Vision & Pattern Recognition*, pages 593–600, Seattle, 1994.

[35] S. M. Smith and J. M. Brady. SUSAN - a new approach to low level image processing. *Int. J. Computer Vision*, 23(1):45–78, 1997.

[36] M. Toews and T. Arbel. Entropy-of-likelihood feature selection for image correspondence. In *Int. Conf. Computer Vision*, pages 1041–1047, Nice, France, 2003.

[37] P. H. S. Torr and A. Zisserman. Feature based methods for structure and motion estimation. In B. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, pages 278–294, Corfu, Greece, 2000. Springer-Verlag LNCS.

[38] O. Zuniga and R. Haralick. Corner detection using the facet model. In *Int. Conf. Computer Vision & Pattern Recognition*, pages 30–37, 1983.

Joint Feature Distributions for Image Correspondence

Bill Triggs

CNRS-INRIA, 655 avenue de l'Europe, 38330 Montbonnot, France.

Bill.Triggs@inrialpes.fr ◇ *http://www.inrialpes.fr/movi/people/Triggs*

Abstract

We introduce 'Joint Feature Distributions', a general statistical framework for feature based multi-image matching that explicitly models the joint probability distributions of corresponding features across several images. Conditioning on feature positions in some of the images gives well-localized distributions for their correspondents in the others, and hence tight likelihood regions for correspondence search. We apply the framework in the simplest case of Gaussian-like distributions over the direct sum (affine images) and tensor product (projective images) of the image coordinates. This produces probabilistic correspondence models that generalize the geometric multi-image matching constraints, roughly speaking by a form of model-averaging over them. These very simple methods predict accurate correspondence likelihood regions for any scene geometry including planar and near-planar scenes, without ill-conditioning or explicit model selection. Small amounts of distortion and non-rigidity are also tolerated. We develop the theory for any number of affine or projective images, explain its relationship to matching tensors, and give results for an initial implementation.

Keywords: Joint Feature Distributions, matching constraints, multi-image geometry, feature correspondence, statistical modelling.

1 Introduction

This paper introduces a natural statistical framework for multi-image feature correspondence, **Joint Feature Distributions (JFD's)**, and uses them to "probabilize" the entire range of affine and projective geometric matching constraints [12,15,2,3,5,7,21,20]. JFD's are simply joint probability distributions over the positions of corresponding features in $m > 1$ different images, used as a summary of some population of interesting correspondences (all valid ones, those near a particular surface or object, background ones...). Conditioning on some of the features gives tight probabilistic correspondence search regions for the remaining ones. Although we will choose parametric forms that reproduce and generalize the standard matching constraints, JFD's are in essence descriptive statistical models rather than normative geometric ones: they aim to summarize the observed behaviour of the given training correspondences, not to rigidly constrain them to an ideal predefined geometry. We believe that JFD's will become the standard method for many correspondence search problems. Their benefits over matching constraints include: more precise search focusing; built-in handling of noise and distortion (small non-rigidities, lens distortion...); and globally stable estimation, even for geometries that are degenerate for classical matching constraints. The projective two image model is perhaps the most useful. It generalizes the epipolar constraint, but instead of searching along the full length of epipolar lines, it searches ellipses (Gaussians) whose centre, axis, length and width are determined by the point being matched, its epipolar line, the range of disparities seen in the training data, and the noise level. JFD's stably and accurately adapt to any scene geometry from deep 3D through to coplanar: as the depth range of the training data decreases, the search ellipses progressively shorten until ultimately the model becomes essentially homographic. There is no ill-conditioning for near-coplanar scenes, no need to choose between epipolar and homographic correspondence models, and no under- or over-estimation of the plausible correspondence regions. In contrast, epipolar models typically search entire (and perhaps inaccurate) epipolar lines, wasting effort and greatly increasing the probability of false correspondences.

The idea of using JFD's for image correspondence is very natural and obvious in retrospect. As far as we know it has not appeared before, but there are many related threads in the literature. JFD's react against explicit model selection for matching constraints [9,10,17] by incorporating the well-known statistical rule that you can only predict events similar to the ones you trained on — extrapolating a full epipolar geometry from near-coplanar data is unstable, but irrelevant for predicting near-coplanar correspondences, *c.f. e.g.* [13]. JFD's have analogies with Bayesian model averaging [18] but are much simpler and more direct. Plane+parallax [11,14,8,23,1,22] offers stabler geometric parametrizations than matching tensors for near-planar scenes. These could no doubt be "probabilized" in much the same way as we do here.

Another aspect of this work is a new theoretical framework for studying multi-image geometry and especially matching constraints, based on the notion of the **tensor joint image**. We will use some isolated results from this,

Extended version of a paper appearing in the 2001 IEEE Int. Conf. Computer Vision. This work was supported by European Union FET project VIBES.

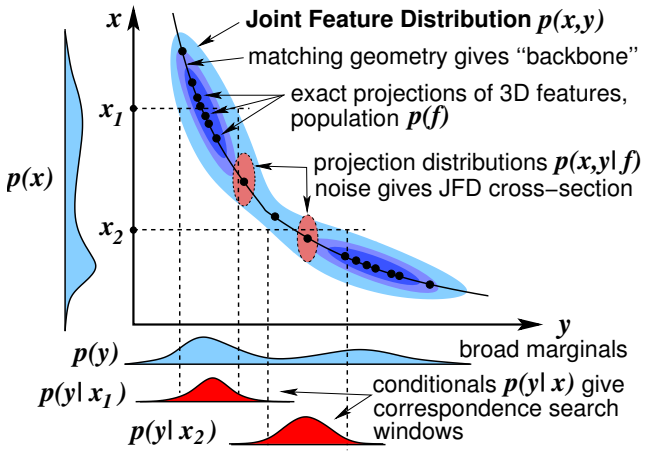


Figure 1: The basic principle of JFD feature correspondence.

but the full development had to be omitted for lack of space.

§2 sketches the general principles of JFD matching, §3 develops some tools, then we focus on Gaussian-like JFD models for affine (§4) and perspective (§5) camera geometries. §6 briefly discusses the implementation and some preliminary experiments and §7 concludes.

Notation: We assume familiarity with affine and projective matching constraints at the level of, *e.g.* [6], and the ability to think tensorially at need [21,20]. Slanted fonts denote inhomogeneous $\boldsymbol{x},\boldsymbol{y}$ and homogeneous x,y image vectors, upright fonts inhomogeneous $\mathbf{x},\mathbf{y}$ and homogeneous x,y 3D ones. P denotes 3×4 image projection matrices, $\mathbf{p}()$ probability distributions, $[\,\cdot\,]_\times$ 3×3 cross product matrices ($[\,\mathsf{x}\,]_\times\,\mathsf{y}=\mathsf{x}\wedge\mathsf{y}$), $\langle-\rangle_{\boldsymbol{x}}$ expectation over the distribution of $\boldsymbol{x}$.

2 Joint Feature Distributions

We can model the m noisy image projections $\boldsymbol{x}_{i\,|\,i=1\ldots m}$ of a fixed 3D feature $\mathbf{f}$ as probability distributions $\mathbf{p}_i(\boldsymbol{x}_i\,|\,\mathbf{f})$ centred on $\mathbf{f}$'s true projections, with widths determined by the relevant noise levels. More generally, the joint distribution $\mathbf{p}(\boldsymbol{x}_1,...,\boldsymbol{x}_m\,|\,\mathbf{f})$ is typically well-localized, and for independent noise factors as $\prod_i \mathbf{p}_i(\boldsymbol{x}_i\,|\,\mathbf{f})$. If $\mathbf{f}$ now varies across some population of 3D features with distribution $\mathbf{p}(\mathbf{f})$, the **Joint Feature Distribution (JFD)** of the resulting population of image features is:

$$\mathbf{p}(\boldsymbol{x}_1,...,\boldsymbol{x}_m) \;\equiv\; \int \mathbf{p}(\boldsymbol{x}_1,...,\boldsymbol{x}_m\,|\,\mathbf{f})\,\mathbf{p}(\mathbf{f})\,\mathbf{df} \qquad (1)$$

For broad priors $\mathbf{p}(\mathbf{f})$, the one-image marginals $\mathbf{p}(\boldsymbol{x}_i) \;\equiv\; \int \mathbf{p}(\boldsymbol{x}_1,...,\boldsymbol{x}_m)\,\boldsymbol{dx}_1...\boldsymbol{dx}_{i-1}\boldsymbol{dx}_{i+1}...\boldsymbol{dx}_m \;=\; \int \mathbf{p}_i(\boldsymbol{x}_i\,|\,\mathbf{f})\,\mathbf{p}(\mathbf{f})\,\mathbf{df}$ are typically broad and uninformative. But the 'sharpness' of the original image projections $\mathbf{p}_i(\boldsymbol{x}_i\,|\,\mathbf{f})$ is not entirely lost: *The JFD* $\mathbf{p}(\boldsymbol{x}_1,...,\boldsymbol{x}_m)$ *remains highly correlated and still encodes most of the precise location information.* In particular, the **Conditional Feature Distributions (CFD's)** like

$$\mathbf{p}(\boldsymbol{x}_1\,|\,\boldsymbol{x}_2,...,\boldsymbol{x}_m) \;\equiv\; \frac{\mathbf{p}(\boldsymbol{x}_1,\boldsymbol{x}_2,...,\boldsymbol{x}_m)}{\mathbf{p}(\boldsymbol{x}_2,...,\boldsymbol{x}_m)} \qquad (2)$$

encode precise inter-image dependencies that are efficient tools for correspondence search. Fig.1 illustrates the principle for two 1D projective images $\boldsymbol{x},\boldsymbol{y}$ of a 1D scene. The JFD $\mathbf{p}(\boldsymbol{x},\boldsymbol{y})$ encodes a strong probabilistic dependency between $\boldsymbol{x}$ and $\boldsymbol{y}$ whose "backbone" is the underlying geometric correspondence (here a 1D homography). The marginals $\mathbf{p}(\boldsymbol{x})$, $\mathbf{p}(\boldsymbol{y})$ are broad and uninformative, but given a particular feature $\boldsymbol{x}$, the CFD $\mathbf{p}(\boldsymbol{y}\,|\,\boldsymbol{x})$ (the normalized cross-section through $\mathbf{p}(\boldsymbol{x},\boldsymbol{y})$ at $\boldsymbol{x}$) is sharply peaked at the corresponding $\boldsymbol{y}$ value. We can use this to predict tight probabilistic search regions for $\boldsymbol{y}$ given $\boldsymbol{x}$, and vice versa.

The abstract JFD framework has rich analogies with, and generalizes, conventional multi-image geometry — see table 1. It applies to any correspondence relationship that can be modelled probabilistically, regardless of feature type, parametrization, number of images, rigidity or distortion. But it is most useful when (suitable parametric forms can be chosen so that) the estimated JFD's have strong correlations that provide accurate search focusing. The link with geometry is strongest for correspondences governed by matching constraints. Then, as in fig.1, the matching geometry forms the "backbone" of the JFD $\mathbf{p}(\boldsymbol{x},\boldsymbol{y})$ and fixes the locations of the CFD's $\mathbf{p}(\boldsymbol{y}\,|\,\boldsymbol{x})$, while the image noise determines the cross-section of the JFD and the widths of the CFD's. The 3D feature population $\mathbf{p}(\mathbf{f})$ or its image marginals $\mathbf{p}(\boldsymbol{x}),\mathbf{p}(\boldsymbol{y})$ determine the height of the JFD along its backbone, but have little direct influence on the CFD shapes.

As with matching constraints, JFD's are image-based models originally derived from 3D quantities (here the 3D feature prior $\mathbf{p}(\mathbf{f})$ and the projection models $\mathbf{p}_i(\boldsymbol{x}_i\,|\,\mathbf{f})$, there the camera matrices P_i), but typically estimated from observed image correspondences. The familiar three stage estimation process [4] still applies: (*i*) build a large set of possible correspondences, *e.g.* by feature detection followed by correlation matching; (*ii*) hypothesize well-supported candidate models, *e.g.* using a robust clusterer such as RANSAC; (*iii*) robustly fit parametric model(s) to the most interesting candidate(s). The fitted models are parametric probability distributions for both JFD's (explicitly) and matching constraints (we actually fit a geometry-based probabilistic noise model). The clustering stage ensures reliable fitting by rejecting false matches and 'uninteresting' true ones, *e.g.* features on moving objects when we are fitting the background, or non-coplanar features when we are fitting a plane. Good clustering is even more critical for JFD's than for matching constraints owing to their polymorphism: they are designed to summarize a user-defined class of observations not to enforce

Entity	**Matching Constraint Approach**	**Joint Distribution Approach**
3D camera geometry	Camera projection mapping, matrices P_i: $\mathbf{f} \rightarrow x_i = \mathsf{P}_i \mathbf{f}$	Conditional feature projection distributions $\mathbf{p}(\boldsymbol{x}_i \,|\, \mathbf{f})$
Image signature of camera geometry	Multi-image matching tensors $\mathcal{T}_{ij\ldots k}$	Joint Feature Distributions $\mathbf{p}(\boldsymbol{x}, ..., \boldsymbol{z})$
Inter-image feature transfer	Tensor based feature transfer $x \simeq \mathcal{T}_{ij\ldots k} \cdot y \cdot \ldots \cdot z$	Conditional Feature Distributions $\mathbf{p}(\boldsymbol{x} \,|\, \boldsymbol{y}, ..., \boldsymbol{z})$
Inter-image feature correspondence	Geometric matching constraints $\mathcal{T}_{ij\ldots k} \cdot x \cdot \ldots \cdot z = 0$	Probability that features correspond, $\mathbf{p}(\boldsymbol{x}, ..., \boldsymbol{z})$, or $\mathbf{p}(\boldsymbol{x} \,|\, \boldsymbol{y}, ..., \boldsymbol{z})$
Scene reconstruction	Ray intersection, tensor-based reconstruction	Posterior 3D feature probability $\mathbf{p}(\mathbf{f} \,|\, \boldsymbol{x}, ..., \boldsymbol{z})$

Table 1: Analogies between the joint distribution approach and multi-image matching constraints.

a predefined structure, so it is much less clear what constitutes an outlier. The obvious approach is to use self-consistency, finding clusters too dense to be probable under broader members of the parametric distribution family *c.f. e.g.* [4, 18]. We will not go into these difficult grouping issues here, but we expect JFD's to be effective correspondence models for many natural grouping classes such as points on compact moving objects.

From now on we focus on deriving efficient parametric models for JFD's. We will only consider Gaussian-like models, which appear to be the simplest useful parametric forms. Gaussians can only capture linear dependencies, so to produce JFD's that can mimic the standard matching constraints, we will need parametrizations that make these constraints appear linear. As in matching constraint estimation, we do this by mapping the input observations into a suitable *joint image* space, containing the direct sum (juxtaposition) of the input coordinates for affine models, but their tensor (outer) product for projective ones. For example, for projective fundamental matrix or epipolar JFD estimation, we map the n correspondences $(\boldsymbol{x}, \boldsymbol{x}')$ to homogeneous 9-D outer product vectors $x \otimes x' \sim (1, x, y, x', y', xx', xy', yx', yy')^\top$ and build a $9 \times n$ measurement matrix M from these. The fundamental matrix estimate uses just the smallest eigenvector of $\mathsf{M}\mathsf{M}^\top$ (*e.g.* [6]), whereas the JFD model captures the underlying uncertainty using an appropriately-weighted "average" over all of the eigenvectors (in fact, $(\mathsf{M}\mathsf{M}^\top)^{-1}$). Conditioning the JFD gives compact correspondence search regions consistent with all (not just one!) of the likely models in the average. The JFD is loosely analogous to "model averaging" of fundamental matrices [18], but it is based directly on the input correspondences, not on a blurred geometric model.

3 Scatter & Covariance

Homogeneous covariance: Before starting we introduce some tools. We encode distributions homogeneously. Given an uncertain affine point $\boldsymbol{x}$ with mean $\bar{\boldsymbol{x}}$, covariance $\boldsymbol{V}$ and homogeneous vector $x = \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix}$, its **homogeneous covariance** X, **homogeneous information** X^{-1} and χ^2 **value** are:

$$X \equiv \langle x\, x^\top \rangle_{\boldsymbol{x}} = \begin{pmatrix} \bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^\top + \boldsymbol{V} & \bar{\boldsymbol{x}} \\ \bar{\boldsymbol{x}}^\top & 1 \end{pmatrix} \quad (3)$$

$$X^{-1} = \begin{pmatrix} \boldsymbol{V}^{-1} & -\boldsymbol{V}^{-1}\bar{\boldsymbol{x}} \\ -\bar{\boldsymbol{x}}^\top \boldsymbol{V}^{-1} & +\bar{\boldsymbol{x}}^\top \boldsymbol{V}^{-1}\bar{\boldsymbol{x}} + 1 \end{pmatrix} \quad (4)$$

$$\chi^2(\boldsymbol{x} \,|\, \bar{\boldsymbol{x}}, \boldsymbol{V}) \equiv (\boldsymbol{x} - \bar{\boldsymbol{x}})^\top \boldsymbol{V}^{-1} (\boldsymbol{x} - \bar{\boldsymbol{x}}) = x^\top X^{-1} x - 1 \quad (5)$$

The mean, covariance and information of the Gaussian fit neatly into the homogeneous matrices X, X^{-1}. Given a collection of training points $\{\boldsymbol{x}_p\}_{p=1\ldots n}$, their **homogeneous scatter matrix** $\frac{1}{n}\sum_p x_p\, x_p^\top$ encodes their mean and covariance, and hence defines an approximate Gaussian probability model for the point population. If the points are also uncertain, their **smoothed homogeneous scatter** $\frac{1}{n}\sum_p \langle x_p\, x_p \rangle_{\boldsymbol{x}_p}$ encodes the mean and covariance of the mixture distribution generated by the sum of the individual point distributions. Viewed as a summary of the population statistics, this double-counts the noise and hence overestimates the covariance, but when there are relatively few points this smoothed but biased estimate is often preferable to the unsmoothed one because it contains additional information about the noise level. Either type of scatter matrix can be used when estimating JFD's below.

Note that these formulae require the homogeneous point vectors to be affinely normalized (scaled so that the last coordinate is 1). We assume this throughout the paper. Although many formulae (notably in §5) appear projective, they all are based on the standard "noise in pixels" image plane error model which is intrinsically affine. For a start at building a projectively covariant error model, see [19].

Dual covariance: Some matching constraints are based on the lines through an image point rather than the point itself. The simplest is the homographic relationship $x \simeq H y$. This can be written in constraint form as $u H y = 0 = v H y$ where u, v are any two independent lines through x. For least squares estimation we square and sum the constraints: $0 \approx (u H y)^2 +$

$(v\,H\,y)^2 = y^\top H^\top (u\,u^\top + v\,v^\top)\,H\,y$. We will view $u\,u^\top + v\,v^\top$ as the "homogeneous scatter matrix" of the chosen set $\{u, v\}$ of lines through x. More generally we could use $\frac{2}{n}\sum_{i=1}^{n} u_i\,u_i^\top$ where $\{u_i\}$ is any rank 2 set of lines through x. These rank 2 matrices encode x as their null vector, and each defines its own importance weighting over the lines through x (*i.e.* the constraints). To be more systematic, we fix a standard weighting procedure that defines our notion of "the uniform distribution of lines" through any given x. Algebraically, the most uniform way to write the squared homography constraints is $\|\,[\,x\,]_\times\,H\,y\|^2 \approx 0$, which leads to the "scatter matrix" $[\,x\,]_\times^\top\,[\,x\,]_\times = \|x\|^2 I - x\,x^\top$. This is not projectively covariant, but we can make it so by introducing a fixed quadric matrix Q, which we usually take to be the identity matrix in a well-normalized projective frame. We then define the **dual covariance** of x to be $\widetilde{X} \equiv (x^\top Q\,x)\,Q - (Q\,x)(Q\,x)^\top$. Or if x is uncertain with homogeneous covariance $X = \langle x\,x^\top \rangle$, the dual covariance is the expectation of this: $\widetilde{X} \equiv \mathrm{trace}(Q\,X)\,Q - Q\,X\,Q$. For $Q = I$ this becomes (*c.f.* (3)):

$$\widetilde{X} = \begin{pmatrix} 1{+}y^2{+}V_{yy} & -xy{-}V_{xy} & -x \\ -xy{-}V_{xy} & 1{+}x^2{+}V_{xx} & -y \\ -x & -y & x^2{+}y^2{+}V_{xx}{+}V_{yy} \end{pmatrix} \quad (6)$$

For nonsingular Q, $X = \frac{1}{d}\,\mathrm{trace}(Q^{-1}\,\widetilde{X})\,Q^{-1} - Q^{-1}\,\widetilde{X}\,Q^{-1}$, so dualization is reversible (d is the space dimension, here 2). Our JFD models of line-through-point constraints (homographies, trifocal, quadrifocal) are all based on dual covariances.

4 The Affine JFD

We now develop a Gaussian JFD model for point features under affine image projection. This is the simplest useful model, and a good warm-up for the projective case. To keep the projective link clear we work in affine-homogeneous coordinates rather than centred inhomogeneous ones. Our JFD model must reproduce and "probabilize" the affine matching constraints. But these are already linear in the image coordinates and Gaussians naturally model linear relationships, so the problem is trivial. Suppose that the training data is n correspondences in m affine images $\boldsymbol{x}_{ip\,|\,i=1\ldots m,p=1\ldots n}$. Collect the components of each correspondence into a $2m+1$ component homogeneous **affine joint image** vector $\mathsf{x}_p \equiv (\boldsymbol{x}_{1p}^\top \ldots \boldsymbol{x}_{mp}^\top\ 1)^\top$, and form these into a $(2m+1) \times n$ **affine measurement matrix** $\mathsf{M} \equiv (\mathsf{x}_1, \ldots, \mathsf{x}_n)$. Viewing our correspondences as points in joint image space, their homogeneous scatter matrix is simply $\mathsf{V} \equiv \frac{1}{n}\sum_{p=1}^{n} \mathsf{x}_p\,\mathsf{x}_p^\top = \frac{1}{n}\,\mathsf{M}\,\mathsf{M}^\top$, or if we choose to use the smoothed scatter: $\mathsf{V} \equiv \frac{1}{n}\sum_p \langle \mathsf{x}_p\,\mathsf{x}_p^\top \rangle = \frac{1}{n}\sum_p \left(\mathsf{x}_p\,\mathsf{x}_p^\top + \begin{pmatrix} \boldsymbol{V}_p & \boldsymbol{0} \\ \boldsymbol{0} & 0 \end{pmatrix}\right) = \frac{1}{n}\,\mathsf{M}\,\mathsf{M}^\top + \frac{1}{n}\begin{pmatrix} \sum_p \boldsymbol{V}_p & \boldsymbol{0} \\ \boldsymbol{0} & 0 \end{pmatrix}$, where $\boldsymbol{V}_p$ is the $2m \times 2m$ inhomogeneous joint noise covariance of x_p (for independent noise, $\boldsymbol{V}_p$ is block diagonal with 2×2 blocks). Our affine JFD model is simply the Gaussian that best describes this population of joint image vectors, *i.e.* the Gaussian with homogeneous covariance V and homogeneous information V^{-1}.

Why does this work? - The theory of affine projection tells us that for ideal noiseless observations, $\mathrm{rank}(\mathsf{M}) \leq 4$, *i.e.* the vectors x_p span a 3D affine space [16]. We are modelling noisy observations, but the ideal behaviour tells us what to expect: M typically has 1 large 'homogenization', ≤ 3 large 'geometry' and $\geq 2m-4$ small 'noise' singular vectors, and similarly for $\mathsf{V} = \frac{1}{n}\mathsf{M}\,\mathsf{M}^\top$ with eigenvectors. So the JFD is typically very 'flat' — broad and featureless along the 'geometry' directions, but narrow along the remaining 'noise' ones. Conditioning on an image point $\boldsymbol{x}$ effectively freezes two of the 'geometry' directions, so at most one remains, spanning the joint epipolar line of $\boldsymbol{x}$ in the remaining images. Even this direction is restricted to the breadth of the training population, so for coplanar data it will shrink to a point.

For our Gaussian JFD's, conditioning leads to familiar Schur-complement matrix formulae. To do the calculation, partition $(\boldsymbol{x}_1^\top \ldots \boldsymbol{x}_m^\top)^\top$ into known components $\boldsymbol{k}$ and unknown ones $\boldsymbol{u}$, freeze $\boldsymbol{k}$ at their known values in $\chi^2(\mathsf{x}) = \mathsf{x}^\top\,\mathsf{V}^{-1}\,\mathsf{x} - 1$ (5), and complete the squares to find the conditional log likelihood of the remaining unknowns $\boldsymbol{u}$. Let $\bar{\boldsymbol{k}}, \bar{\boldsymbol{u}}$ be the training set means of $\boldsymbol{k}, \boldsymbol{u}$. Partition the corresponding information as $\left(\begin{smallmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^\top & \boldsymbol{C} \end{smallmatrix}\right)$, where $\boldsymbol{A} = (\mathsf{V}^{-1})_{\boldsymbol{kk}}$, *etc*. The search region for the unknowns $\boldsymbol{u}$ given the knowns $\boldsymbol{k}$ is defined by the CFD $\mathbf{p}(\boldsymbol{u}\,|\,\boldsymbol{k})$, which turns out to have mean $\bar{\boldsymbol{u}} - \boldsymbol{C}^{-1}\boldsymbol{B}^\top(\boldsymbol{k} - \bar{\boldsymbol{k}})$ and covariance $\boldsymbol{C}^{-1}$. (NB: The *population* covariance of $\boldsymbol{u}$ is $\mathsf{V}_{\boldsymbol{uu}} = (\boldsymbol{C} - \boldsymbol{B}^\top\boldsymbol{A}^{-1}\boldsymbol{B})^{-1}$, which is usually much larger). If $\boldsymbol{k}$ is an uncertain measurement with covariance $\boldsymbol{D}^{-1}$, the CFD $\mathbf{p}(\boldsymbol{u}\,|\,\boldsymbol{k})$ is broadened to mean $\bar{\boldsymbol{u}} - \boldsymbol{C}^{-1}\boldsymbol{B}^\top(\boldsymbol{A}+\boldsymbol{D} - \boldsymbol{B}\boldsymbol{C}^{-1}\boldsymbol{B}^\top)^{-1}\boldsymbol{D}(\boldsymbol{k} - \bar{\boldsymbol{k}})$ and covariance $(\boldsymbol{C} - \boldsymbol{B}^\top(\boldsymbol{A}+\boldsymbol{D})^{-1}\boldsymbol{B})^{-1}$. (Usually $\boldsymbol{D} \gg \boldsymbol{A}$ so we can drop the $\boldsymbol{A}$'s). The prior likelihood for observing the knowns $\boldsymbol{k}$ in the first place is $\chi^2(\boldsymbol{k}\,|\,\mathsf{V}) = (\boldsymbol{k} - \bar{\boldsymbol{k}})^\top\,(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}^{-1}\boldsymbol{B}^\top)\,(\boldsymbol{k} - \bar{\boldsymbol{k}})$. As usual in such calculations, there are other forms for these expressions that may be stabler or more efficient, but we will not go into this here.

Implementation is straightforward: form the homogeneous scatter V from the training data, invert to get the information V^{-1} (the Gaussian JFD model), partition and condition on known observations to get search windows for their unknown correspondents. One minor snag is that $\mathsf{V} = \mathsf{M}\,\mathsf{M}^\top$ becomes rank deficient (rank ≤ 4) for noiseless data, so the estimated information V^{-1} becomes infinite. This is correct — exact geometry allows infinitely accurate predictions — but numerically inconvenient. In practice we avoid it by adding a small diagonal regularizer $\mathrm{diag}(\epsilon, \ldots, \epsilon, 0)$ to V before inverting. Typically $\epsilon \sim 10^{-9}$:

large enough to prevent loss of numerical precision during the inversion, but not so large as to blur the final estimates significantly. Similarly, V is rank deficient for $n \leq 2m$ noisy but unsmoothed correspondences because we do not have enough observations to estimate all of the noise covariances. The solution is to incorporate more noise information, *e.g.* using smoothed scatters.

5 The Projective JFD

Affine JFD's are too rigid to model perspective distortion exactly, so we now develop more flexible projective models. As before we consider only Gaussian-like models, so to mimic the matching constraints we need to use parametrizations in which these become linear. Projective matching constraints are *multi*linear in the homogeneous coordinates of their image features, but as in "linear" matching tensor estimation we can make the problem appear linear by treating multilinear combinations as if they were independent coordinates, *i.e.* by mapping the input feature vectors to their outer (Kronecker) product tensor. For example, for two images we can not use just the image coordinates $\mathsf{x} = (x, y, 1)^\top, \mathsf{x}' = (x', y', 1)^\top$ or the affine joint image vector $(x, y, x', y', 1)^\top$, because the projective matching constraints also have bilinear terms $xx', ..., yy'$. Instead we need to collect the components of the outer product $\mathsf{x}\,\mathsf{x}'^\top$ into a vector $(xx', xy', x, yx', yy', y, x', y', 1)^\top$ and use these as working coordinates. More generally, for point features in any number of images, it turns out (proof omitted) that tensoring the input coordinates is necessary and sufficient to linearize all of the matching constraints linking the images, and generically the only linear constraints on these coordinates are matching ones. So we really have no choice: to linearize the projective matching constraints linking features $\mathsf{x}_1, ..., \mathsf{x}_m$ from m images, we have to use the 3^m components of their **joint image tensor** $\mathsf{t} = \mathsf{x}_1 \otimes ... \otimes \mathsf{x}_m$ as working coordinates. We will view t both as a 3^m-component vector and as a tensor $\mathsf{t}^{A...D} = \mathsf{x}_1^A \cdot ... \cdot \mathsf{x}_m^D$ (indices $A...D = 1...3$). Assuming affine normalization for $\mathsf{x}_1, ..., \mathsf{x}_m$, our projective JFD models are "Gaussians in t-space", $\mathbf{p}(\mathsf{t}) \sim e^{-L/2}$, with negative log likelihood:

$$L = \mathsf{t}^\top \mathsf{W}\, \mathsf{t} = \mathsf{W}_{A...D\,A'...D'}(\mathsf{x}_1^A ... \mathsf{x}_m^D)(\mathsf{x}_1^{A'} ... \mathsf{x}_m^{D'}) \tag{7}$$

The JFD is parametrized by the **homogeneous information tensor** W, viewed as a symmetric positive definite $3^m \times 3^m$ matrix generalizing the homogeneous information. This model has the following useful properties:

1. It naturally models uncertain matching constraints. Algebraically, the simplest way to represent and combine uncertain constraints is to use weighted sums of squared constraint violations. For linear constraints such as the matching constraints on t, this yields nonnegative quadratic forms in the variables, *i.e.* Gaussians in t.
2. If we freeze some of the variables x_i at arbitrary values, L retains its tensored-quadratic form in the remaining ones, with coefficients given by W contracted against the frozen variables. So conditioning on known values for search region (CFD) prediction reduces to trivial tensor contraction — even simpler than the affine case.
3. Conditioning down to a single image gives a standard Gaussian expressed in homogeneous form, so predicting its high-probability search regions is easy.

JFD estimation: Now consider how to estimate a projective JFD W that summarizes a given set of training correspondences $(\mathsf{x}_{1p}, ..., \mathsf{x}_{mp})_{\,|\,p=1...n}$. By analogy with the affine case we treat the joint image tensors $\mathsf{t}_p = \mathsf{x}_{1p} \otimes ... \otimes \mathsf{x}_{mp}$ of the training correspondences as 3^m-component affine-homogeneous vectors and build their $3^m \times 3^m$ homogeneous scatter matrix $\mathsf{V} = \frac{1}{n}\sum_p \mathsf{t}_p\, \mathsf{t}_p^\top = \frac{1}{n}\mathsf{M}\,\mathsf{M}^\top$ where $\mathsf{M} = (\mathsf{t}_1, ..., \mathsf{t}_n)$ is the $3^m \times n$ measurement matrix familiar from linear matching tensor estimation. We then invert to get the JFD parameter estimate $\mathsf{W} \approx \mathsf{V}^{-1}$. This last step is unfortunately only heuristic[1] but it appears to work reasonably well in practice, perhaps because (if we imagine the eigen-decomposition of V being inverted) it gets at least the noise model in t's block of affine coordinates and the noiseless perspective corrections right.

As in the affine case, if we have uncertainties for the features we can use them to stabilize the JFD's noise level estimates, and we do this by taking expectations over noise when calculating the scatter. We assume independent noise so that tensor expectations factor into single-image ones. Working tensorially, the **smoothed scatter tensor** is:

$$\begin{aligned}\mathsf{V}^{A...D\,A'...D'} &= \tfrac{1}{n}\textstyle\sum_p \left\langle (\mathsf{x}_{1p}^A ... \mathsf{x}_{mp}^D)(\mathsf{x}_{1p}^{A'} ... \mathsf{x}_{mp}^{D'}) \right\rangle \\ &= \tfrac{1}{n}\textstyle\sum_p \left\langle \mathsf{x}_{1p}^A \mathsf{x}_{1p}^{A'} \right\rangle \cdot ... \cdot \left\langle \mathsf{x}_{mp}^D \mathsf{x}_{mp}^{D'} \right\rangle \\ &= \tfrac{1}{n}\textstyle\sum_p \mathsf{X}_{1p}^{AA'} \cdot ... \cdot \mathsf{X}_{mp}^{DD'}\end{aligned} \tag{8}$$

where $\mathsf{X}_{ip} = \left\langle \mathsf{x}_{ip}\, \mathsf{x}_{ip}^\top \right\rangle$ are the homogeneous covariances of the input features. Once again, this smoothes the JFD estimate at the cost of some double-counting of noise. It

[1] Our projective JFD's are not really Gaussians because their input coordinates t are restricted to a nonlinear $(2m)$-D subvariety of their (3^m)-D space — tensors of the rank-one form $\mathsf{t} = \mathsf{x}_1 \otimes ... \otimes \mathsf{x}_m$. Gaussian integrals over this restricted space are intractable, so we can not calculate the normalization factor that makes the JFD into a correctly normalized probability distribution. This factor is indispensable for estimating W. For training data with scatter V on a normalized distribution family $\mathbf{p}(\mathsf{t}) = e^{-(\mathsf{t}^\top \mathsf{W}\, \mathsf{t} - N(\mathsf{W}))/2}$, where $N(\mathsf{W})$ is the normalization, maximum likelihood estimation reduces to minimizing $\mathrm{trace}(\mathsf{W}\,\mathsf{V}) - N(\mathsf{W})$ with respect to W, with implicit solution W such that $\mathsf{V} = \frac{dN(\mathsf{W})}{d\mathsf{W}}$. For a true Gaussian, $N(\mathsf{W}) = \log\det \mathsf{W} - d\log(2\pi)$ and hence $\frac{dN}{d\mathsf{W}} = \mathsf{W}^{-1}$.

is particularly useful when there are $n < 3^m$ training features (which is common for $m \geq 3$). As a safeguard, we also add a $3^m \times 3^m$ diagonal regularizer $\mathrm{diag}(\epsilon, ..., \epsilon, 0)$ to V, where typically $\epsilon \sim 10^{-8}$. These measures are even more necessary in the projective case than in the affine one, as V is both large (so that many measurements are required to span it) and structurally ill-conditioned (because "perspective effects are usually small" compared to affine ones). The ill-conditioning is normal and causes no problems so long as we regularize enough to prevent it from causing loss of numerical precision.

'Epipolar' JFD *vs*. linear fundamental matrix estimation: Both methods start with the $9 \times n$ measurement matrix M of the tensored measurements. Form the 9×9 scatter $V = \mathsf{M}\mathsf{M}^\top$, let $V = \sum_{a=1}^{9} \lambda_a f_a f_a^\top$ be its eigendecomposition, and write the eigenvectors f_a as 3×3 "fundamental matrix candidates" F_a. The conventional linear fundamental matrix estimate is the smallest eigenvector F_9 of V, or equivalently the largest of $W = V^{-1}$. On test correspondences $t = x \otimes x'$, the JFD estimate $W = V^{-1}$ has unnormalized "log likelihood" penalty function $t^\top W t = \sum_{a=1}^{9} \lambda_a^{-1} (f_a\, t)^2 = \sum_{a=1}^{9} \lambda_a^{-1} |x\, F_a\, x'|^2$, *i.e.* "a weighted sum of possible epipolar constraints". Similarly, conditioning on x gives conditional log likelihood $x'^\top A x'$ for x', where $A_{A'B'} = W_{AB\,A'B'}\, x^A\, x^B$, *i.e.* the correspondence search regions are defined by "weighted scatters of possible epipolar lines" $A = \sum_{a=1}^{9} \lambda_a^{-1} (x^\top F_a)(x^\top F_a)^\top$. The fundamental matrix estimate amounts to truncating W at its largest eigenvector, giving effective penalty function $(f_9^\top t)^2 = |x\, F_9\, x'|^2$, *i.e.* the estimated epipolar constraint violation. For small noise and strong data, V has just one very small eigenvalue, so the penalty sum is entirely dominated by F_9 and the JFD model reduces to the fundamental matrix one. But if V has several small eigenvalues owing to noisy or weak data (*e.g.* coplanar data makes V rank 6 with 3 "noise" eigenvalues), these all contribute significantly the penalty sum, which becomes a kind of weighted average over these observed "constraints" on the data, restricting the directions in which the measurements can vary and hence the size of the "averaged epipolar" correspondence search regions.

Statistical error weighting: Although they include covariances, our linear JFD methods are essentially 'algebraic': they implement heuristic polynomial error models rather than statistically weighted rational ones. We will not consider nonlinear JFD estimation here, but a step towards statistical weighting in the conditioning calculation greatly improves the accuracy of the predicted search regions. For points near the epipole, algebraic weighting produces over-broad search regions (fig.2 top right). As above, the cost transversal to epipolar lines is controlled by the JFD's epipolar line violation term $|x\, F_9\, x'|^2$, where F_9 is the fundamental matrix. For x near the epipole, $x^\top F_9 \approx 0$ and $|x\, F_9\, x'|^2$ is small for any x'. We correct for this heuristically by replacing $A_{A'B'} = W_{AB\,A'B'}\, x^A\, x^B$ in the CFD $x'^\top A x'$ with λA where $\lambda = (W_{AB\,A'B'}\, V^{AB}\, N^{A'B'}) / \mathrm{trace}(A N)$, $N = \mathrm{diag}(1, 1, 0)$, and V is the x-image population scatter. The idea is that if $W \approx f_9\, f_9^\top$ as above, $\mathrm{trace}(A N) \approx x^\top F_9\, N\, F_9^\top\, x$ is the norm of the epipolar line vector $x^\top F_9$ and the numerator is the average of such norms across the training population. So λ reinforces the cost near the epipole without changing its overall population average. This heuristic reweighting procedure works well in practice and we are currently trying to formalize it.

5.1 Dual Space JFD's

The above approach in principle allows us to produce projective JFD's for any type of matching geometry in any number of images, and it is indeed the preferred representation for the practically important 2 image 'epipolar JFD' case. However, for $m > 2$ images or known-coplanar scenes it uses training correspondences very inefficiently. In the space of joint image tensors t, the nonlinear d-D image of a d-D projective subspace of 3D space containing k centres of projection turns out to span a $(\binom{m+d}{d} - k)$-D linear subspace (proof omitted). The above JFD model "learns" (spans) at most one tensor dimension per training correspondence, so just to capture the underlying geometry (let alone the noise) we need $\binom{m+3}{3} - m = 8, 17, 31...$ training correspondences in $m = 2, 3, 4...$ images for general 3D geometry, or $\binom{m+2}{2} = 6, 10, 15...$ for known-coplanar points. In comparison, linear matching tensor estimators need only $8, 7, 6$ correspondences for 3D points and $4, 4, 4$ for coplanar ones. Matching constraints are more efficient when they are tensor-valued, so that a single matching tensor with relatively few coefficients generates several linear constraints on each tensored image correspondence. By mirroring these index structures we can build correspondingly efficient JFD models, at the cost of less image-symmetric representations and an implicit restriction to correspondence models subjacent to the mirrored matching constraint. For example, a JFD based on the index structure of a two image homography constraint can be estimated from 4 correspondences rather than 6, but implicitly commits us to quasi-planar data.

To do this, we simply need to assume a JFD whose form is an average over constraints of the desired type. Free indices arise essentially when points x appear dualized as $[x]_\times$ in the matching constraints, *i.e.* for "any line through the point" style constraints. For example, 2 image homographic constraints with matrix H and 3 image trifocal constraints with tensor T can be written symbolically as $\| [x']_\times H x \|^2$ and $\| [x']_\times (T \cdot x) [x'']_\times \|^2$, or alternatively as $\sum_i |u_i'^\top H x|^2$ and $\sum_{ij} |(u_i')^\top (T \cdot x) u_j''|^2$ where $\{u_i'\}, \{u_j''\}$ are any sets of two (or

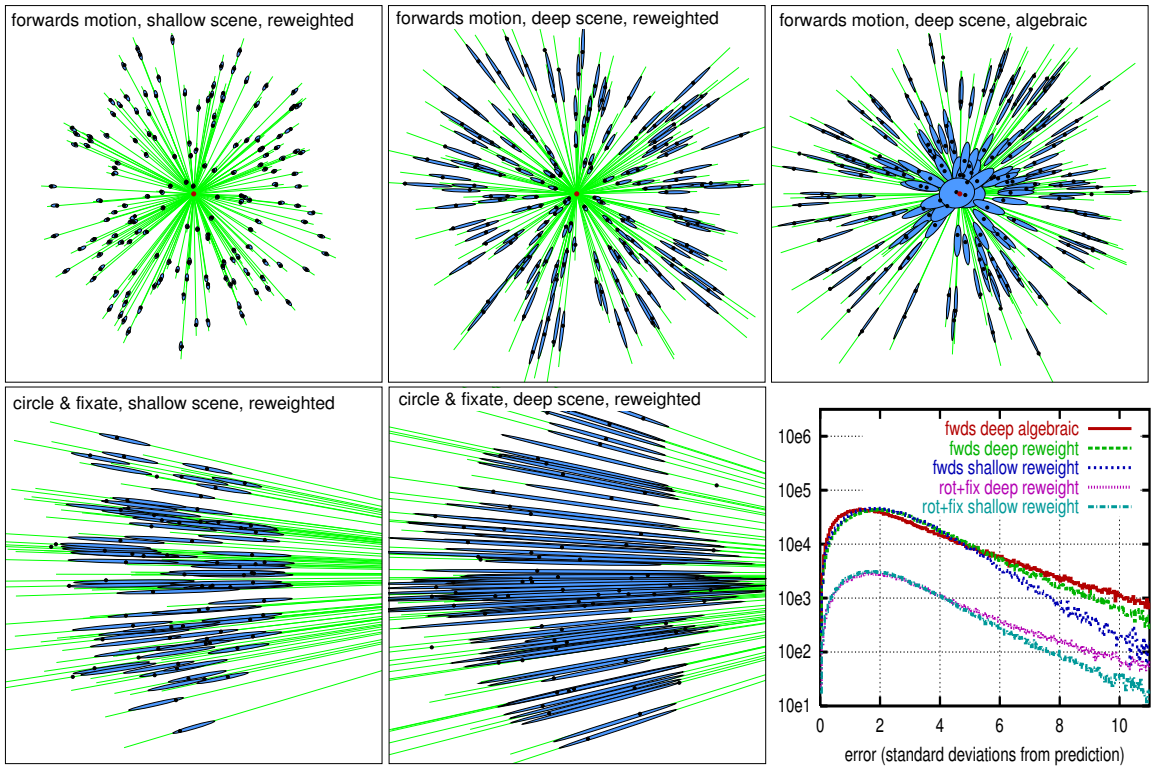


Figure 2: Predicted search ellipses for the projective 2 image JFD, versus the true correspondences and their epipolar lines. The scenes contain uniform random points in a sphere (deep scene) or a sphere flattened to 20% of its depth (shallow scene). *Top right:* Algebraic error weighting leads to incorrect broadening of the search regions near the epipole. The other plots use our CFD reweighting heuristic to correct this (see end of §5.0). *Bottom right:* For all of the geometries tested, the estimated conditional distributions accurately reflect the true correspondence likelihoods. Here we histogram the standard errors of the true observations under the estimated CFD's, for forwards (top curves) and fixation (bottom curves) motions.

more) independent lines through x', x''. Expanding these forms tensorially gives $(H^B_A\, H^{B'}_{A'})X^{AA'}\, \widetilde{X}'_{BB'}$ and $(T^{BC}_A\, T^{B'C'}_{A'})X^{AA'}\, \widetilde{X}'_{BB'}\, \widetilde{X}''_{CC'}$ where $X = x\,x^\top$ and $\widetilde{X}' = [x']^\top_\times [x']_\times$ or $\widetilde{X}' = \sum_i u'_i\, u'_i$, and similarly for $\widetilde{X}''$ with x'', u''_j. The squared constraints can be expressed compactly in terms of the homogeneous covariance X of x and the scatter matrices $\widetilde{X}', \widetilde{X}''$ of the lines through x', x''. We will fix the relative weighting of the different constraints by systematically using the dual covariances (6) of x', x'' for $\widetilde{X}', \widetilde{X}''$. For our JFD models we take averages over constraints of these forms, *i.e.* we adopt homogeneous Gaussian-like forms with unnormalized log likelihoods $W^{B\,B'}_{A\,A'}\, X^{AA'}\, \widetilde{X}'_{BB'}$ and $W^{BC\,B'C'}_{A\,A'}\, X^{AA'}\, \widetilde{X}'_{BB'}\, \widetilde{X}''_{CC'}$, where X and $\widetilde{X}', \widetilde{X}''$ are the (noiseless) normal and dual homogeneous covariances of the test correspondences (x, x', x''). These are still quadratic in the tensored measurements $x \otimes x' (\otimes x'')$, so they are reparametrizations of the general projective JFD models developed above. They are parametrized by information tensors $W^{B\,B'}_{A\,A'}$, $W^{BC\,B'C'}_{A\,A'}$, which can be viewed respectively as 9×9 and 27×27 homogeneous information matrices representing scatters $\sum_i h_i\, h_i^\top$ and $\sum_i s_i\, s_i^\top$ over possible homography matrices (9-component 'vectors' h_i) and trifocal tensors (27-component 'vectors' s_i). To estimate the models we again build (regularized and possibly smoothed) scatter tensors over training correspondences, here $V^{A\,A'}_{B\,B'} = \frac{1}{n}\sum_p X_p^{AA'}\, \widetilde{X}'_{p\,BB'}$ and $V^{A\,A'}_{BC\,B'C'} = \frac{1}{n}\sum_p X_p^{AA'}\, \widetilde{X}'_{p\,BB'}\, \widetilde{X}''_{p\,CC'}$, treat these as 9×9 and 27×27 homogeneous covariance matrices, and invert to estimate the corresponding information. To use the models for correspondence search, we condition on known feature positions by contracting their normal or dual covariances (as appropriate) into the information tensors until we reach a single-image model, which is necessarily quadratic in the remaining feature vector, *i.e.* a Gaussian whose likelihood regions are simple ellipses. This approach extends to all other matching constraint equations with free covariant indices: we just need to use normal or dual covariances as appropriate, and think tensorially when necessary.

6 Implementation & Experiments

We have implemented the above methods in MATLAB for any number of images and duality structure, but here we only show brief results for the 2 image 'epipolar' JFD. Recall the algorithm: build the 9×9 scatter matrix $V = \frac{1}{n}\sum_p t_p\, t_p^\top$ from the tensored training correspondences t_p, regularize and invert to get the JFD information $W = (V + \mathrm{diag}(\epsilon, ..., \epsilon, 0))^{-1}$, and view this as a tensor $W_{AB\,A'B'}$. Then, for each test correspondence x, form $A_{A'B'} = W_{AB\,A'B'}\, x^A x^B$ and optionally rescale it by $\lambda = (W_{AB\,A'B'}\, V^{AB} N^{A'B'})/\,\mathrm{trace}(AN)$ ($N = \mathrm{diag}(1, 1, 0)$) to correct the error weighting. The resulting A is the conditional information for $x' = (x', y', 1)^\top$, from which Gaussian log-likelihood search ellipses can be found by expanding $x'^\top A\, x'$ as a quadratic and discarding the constant term. Algebraic error weighting does produce over-wide search ellipses for points near the epipole, so it is advisable to include the reweighting factor λ. The reweighted method works well in practice for all of the geometries that we have tested, giving search ellipses aligned with the epipolar lines with realistic lengths and breadths, which progressively shrink to circles as the scene becomes planar.

7 Summary and Conclusions

We introduced Joint Feature Distributions (JFD's), a general statistical framework for image matching based on modelling the joint probability distributions of the positions of corresponding features in different images. The JFD is estimated from a population of training correspondences, then conditioned on the values of test features to produce tight likelihood regions for the corresponding features in other images. We developed relatively simple Gaussian-like JFD models for affine and projective images, which can be viewed as probabilistic "model averages" of the affine and projective multi-image matching constraints. The methods naturally and stably handle any scene geometry from deep 3D through to coplanar scenes, without explicit model selection. For example, the 'epipolar' JFD stably enforces an epipolar, homographic or near-homographic constraint, according to the behaviour of the training data.

Future work: The JFD idea is recent and we are still actively investigating its properties. There are some theoretical loose ends, particularly in the projective case where even the basic $W \approx V^{-1}$ estimation procedure for the "linear" model is only heuristic. We do not yet have JFD's with rigorous statistical error weighting, and it is unclear whether there are JFD analogues of matching tensor consistency relations like $\det(F) = 0$. Both issues are likely to lead to nonlinear models. Practically, we need to develop robust estimators for JFD's. As JFD's are less rigid than matching constraints, self-consistency based clustering will probably be needed to isolate correspondence subpopulations susceptible to JFD modelling. The full population model will thus be a mixture of JFD's. Numerically, we are developing QR and SVD based JFD representations that should be less sensitive to rounding errors than our current scatter / information ones.

A The Tensor Joint Image

This appendix develops the properties of tensor joint image representation. "Recall" two fundamental mappings from algebraic geometry. Given projective spaces $\mathbb{P}^A, ..., \mathbb{P}^D$ with generic points $x^A, ..., z^D$ and dimensions $d_1, ..., d_m$, the **Segré mapping** takes $(x^A, ..., z^D)$ in the Cartesian product (direct sum) space $\mathbb{P}^A \times ... \times \mathbb{P}^D$ to the rank one tensor[2] $t^{A...D} \equiv x^A \cdot ... \cdot z^D$ in the tensor product space $\mathbb{P}^{A...D}$. The **Segré variety** is the image of $\mathbb{P}^A \times ... \times \mathbb{P}^D$ under this mapping. It is a $(\sum_i d_i)$-dimensional algebraic variety in the $(\prod_i (d_i + 1) - 1)$-dimensional projective space $\mathbb{P}^{A...D}$, isomorphic to $\mathbb{P}^A \times ... \times \mathbb{P}^D$, and cut out by the 2×2 determinants of the form $t^{...A_i...A_j...}\, t^{...B_i...B_j...} - t^{...A_i...B_j...}\, t^{...B_i...A_j...} = 0$. Its points linearly span the whole of $\mathbb{P}^{A...D}$. The Segré mapping is the standard way of giving a Cartesian product a variety structure in algebraic geometry.

The Segré mapping *encapsulates the nonlinearity of multilinear polynomials on* $\mathbb{P}^A \times ... \times \mathbb{P}^D$, in the sense that any multilinear form $\sum_{A...D} c_{A...D}\, x^A ... z^D$ becomes a linear one $\sum_{A...D} c_{A...D}\, t^{A...D}$ in terms of the Segré coordinates $t^{A...D} = x^A ... z^D$. Any subvariety of $\mathbb{P}^A \times ... \times \mathbb{P}^D$ defined by multilinear polynomials is Segré-mapped to the intersection of a *linear* subspace (the null space of the Segré-linearized polynomials) with the Segré variety[3]. The individual homogeneous scale factors of $x^A, ..., z^D$ are confounded in $x^A \cdot ... \cdot z^D$, so the Segré mapping also turns out to be a good way of circumventing problems with homogeneous scale factors.

Similarly, given a projective space $\mathbb{P}^A$ with generic point x^A and dimension d, the **degree** m **Veronese mapping** on $\mathbb{P}^A$ takes x^A to the rank one tensor $x^{A_1 A_2 ... A_M} \equiv x^{A_1}\, x^{A_2} ... x^{A_m}$ in the symmetric tensor product space $\mathbb{P}^{(A_1 A_2 ... A_m)}$. The parentheses $(A_1 ... A_m)$ mean "take the symmetric part": in the tensor product it suffices to restrict attention to the $\binom{d+m}{d}$ *ordered* index combinations

[2] Several competing definitions of rank exist for tensors with more than 2 indices. None is entirely satisfactory, but all agree that outer products of vectors have rank 1, as here.

[3] Multilinear forms p in subsets of the variables $x^A, ..., z^D$ can be homogenized up to full multilinear forms by multiplying in turn by each multilinear combination $x^{A_i} \cdot ... \cdot y^{B_j}$ of the missing variables $x^{A_i}, ..., y^{B_j}$. Projectively, all entries of x^{A_i}, *etc.*, can not vanish at once, so the up-homogenized polynomials all vanish iff p does. In this projectivized sense, the Sergé mapping also linearizes multilinear polynomials of degree less than that of $x^A ... z^D$. The multi-image matching constraints behave this way.

$A_1 \leq A_2 \leq ... \leq A_m$ rather than the d^m unordered ones, as $\mathsf{x}^{A_1}\,\mathsf{x}^{A_2}...\mathsf{x}^{A_m}$ is automatically symmetric under arbitrary permutations of $A_1...A_m$. Analogously to the Segré case, the Veronese variety linearly spans $\mathbb{P}^{(A_1 A_2 ... A_m)}$ and is cut out by 2×2 determinants, and the Veronese mapping *linearizes all degree m polynomials on* $\mathbb{P}^A$, mapping varieties defined by such polynomials (or, by up-homogenization, lower degree ones) to linear slices of the Veronese variety in $\mathbb{P}^{(A_1 A_2 ... A_m)}$.

Now turn to vision. Consider m 3×4 image projections $\mathsf{P}_a^A, ..., \mathsf{P}_a^D$ projecting 3D points $\mathsf{X}^a \in \mathbb{P}^a$ to image points $\mathsf{x}^A \simeq \mathsf{P}_a^A\,\mathsf{X}^a, ..., \mathsf{z}^D \simeq \mathsf{P}_a^D$ in $\mathbb{P}^A, ..., \mathbb{P}^D$. Assemble the image points into a joint image[4] $(\mathsf{x}^A, ..., \mathsf{z}^D) \in \mathbb{P}^A \times ... \times \mathbb{P}^D$ — the image of $\mathbb{P}^a$ under the joint projection $(\mathsf{P}_a^A, ..., \mathsf{P}_a^D)$. A point-tuple is the image of some 3D point iff it satisfies certain well-known **geometric matching constraints** [12,15,2,3,5,7,21,20]. These constraints are multilinear in $(\mathsf{x}^A, ..., \mathsf{z}^D)$, so they become linear under the Segré mapping $(\mathsf{x}^A, ..., \mathsf{z}^D) \longrightarrow \mathsf{x}^A \cdot ... \cdot \mathsf{z}^D$, and hence cut out a linear-intersection subvariety of the Segré variety in the tensor product space $\mathbb{P}^{A...D}$. We call this the **tensor joint image**. It has coordinates of the form $t^{A...D} = (\mathsf{P}_a^A\,\mathsf{X}^a) \cdot ... \cdot (\mathsf{P}_d^D\,\mathsf{X}^d)$ and represents the image of $\mathbb{P}^a$ under the composition of joint image projection and Segré.

The image projections also act naturally on tensor products of $\mathbb{P}^a$, in particular taking a point (symmetric tensor) $\mathsf{T}^{a...d} \in \mathbb{P}^{(a...d)}$ to $t^{A...D} \simeq \mathsf{P}_a^A ... \mathsf{P}_d^D\,\mathsf{T}^{a...d}$ in the image tensor space $\mathsf{P}^{A...D}$. The image of the degree m Veronese mapping $\mathsf{X}^a \in \mathbb{P}^a \longrightarrow \mathsf{X}^a \cdot ... \cdot \mathsf{X}^d \in \mathbb{P}^{(a...d)}$ under this tensor product map is exactly the Segré mapping, so the Veronese variety of $\mathbb{P}^a$ also maps to the tensor joint image. In short, the following diagram is commutative:

$$\begin{array}{ccc} \mathbb{P}^a & \xrightarrow[\mathsf{X}^a \to \mathsf{X}^a \cdot ... \cdot \mathsf{X}^d]{\text{Veronese mapping}} & \mathbb{P}^{(a...d)} \\ \mathsf{X}^a \downarrow & & \downarrow \mathsf{T}^{a...d} \\ (\mathsf{P}_a^A\,\mathsf{X}^a, ..., \mathsf{P}_a^D\,\mathsf{X}^a) & \text{joint projection} \quad \text{tensor projection} & \mathsf{P}_a^A ... \mathsf{P}_d^D\,\mathsf{T}^{a...d} \\ \mathbb{P}^A \times ... \times \mathbb{P}^D & \xrightarrow[(\mathsf{x}^A, ..., \mathsf{z}^D) \to \mathsf{x}^A \cdot ... \cdot \mathsf{z}^D]{\text{Segré mapping}} & \mathbb{P}^{A...D} \end{array}$$

The tensor projection $\mathsf{P}_a^A \cdot ... \cdot \mathsf{P}_d^D$ from the $\binom{m+3}{3}$-linear-dimensional space $\mathbb{P}^{(a...d)}$ to the (often much larger) 3^m-linear-dimensional one $\mathbb{P}^{A...D}$ is linear. Its kernel is spanned exactly by the m camera centre tensors $\mathbf{c}_i^a \cdot ... \cdot \mathbf{c}_i^d$, where $\mathbf{c}_i$ is the centre of projection of camera i. Generically the camera centre tensors are linearly independent in $\mathbb{P}^{(a...d)}$, so the image of $\mathbb{P}^{(a...d)}$ in $\mathbb{P}^{A...D}$ under the tensor projection generically has linear dimension $\binom{m+3}{3} - m$. The composite Veronese / tensor projection mapping has a base point at each camera centre, as expected.

Similarly, the Veronese images of 3D lines and planes have linear dimensions $\binom{m+1}{1}$ and $\binom{m+2}{2}$ in $\mathbb{P}^{(a...d)}$, and $\binom{m+1}{1} - k_1$, $\binom{m+2}{2} - k_2$ under tensor projection into $\mathbb{P}^{A...D}$, where k_1, k_2 are the number of camera centres they contain.

The Veronese image of $\mathbb{P}^a$ linearly spans $\mathbb{P}^{(a...d)}$, so its projection the tensor joint image spans the linear image of $\mathbb{P}^{(a...d)}$ in $\mathbb{P}^{A...D}$. The (Segré-mapped) matching constraints are simply the orthogonal complement of this linear subspace of $\mathbb{P}^{A...D}$. They can't be more restrictive without eliminating (the joint images of) real 3D points, and if they were less restrictive they would necessarily fail to eliminate some invalid image correspondences, as the Segré image of $\mathbb{P}^A \times ... \times \mathbb{P}^D$ linearly spans the whole of $\mathbb{P}^{A...D}$.

The point of all this is that tensoring the image measurements reduces much of the geometry of matching constraints to linear considerations (modulo the nonlinearity of the Segré mapping itself, of course). We claim that this is a good way to understand certain aspects of the structure of the matching constraints. In particular, it provides a suitable space in which to run a projective joint distribution formalism, as it allows simple Gaussian-like distributions to enforce the matching constraints.

[4] If we forget the individual projective depths (homogeneous scale factors) here we get the Cartesian-product joint image, if not we get the "projective joint image" defined in [21,20]. The latter is a *linear* image of $\mathbb{P}^a$ under the $3m \times 4$ "joint projection" matrix $(\mathsf{P}_a^A \; ... \; \mathsf{P}_a^D)^\top$, but not immediately recoverable from the input images. Either will do for our purposes as the Segré mapping below obliterates the relative scales.

References

[1] A. Criminisi, I. Reid, and A. Zisserman. Duality, rigidity and planar parallax. In *European Conf. Computer Vision*, pages 846–861. Springer-Verlag, 1998.

[2] O. Faugeras and B. Mourrain. On the geometry and algebra of the point and line correspondences between n images. In *Int. Conf. Computer Vision*, pages 951–6, 1995.

[3] O. Faugeras and T. Papadopoulo. Grassmann-Cayley algebra for modeling systems of cameras and the algebraic equations of the manifold of trifocal tensors. *Transactions of the Royal Society A*, 1998.

[4] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Computer Graphics and Image Processing*, 24(6):381–395, 1981.

[5] R. Hartley. Lines and points in three views and the trifocal tensor. *Int. J. Computer Vision*, 22(2):125–140, 1997.

[6] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[7] A. Heyden and K. Åström. A canonical framework for sequences of images. In *IEEE Workshop on Representations of Visual Scenes*, Cambridge, MA, June 1995.

[8] M. Irani and P. Anandan. A unified approach to moving object detection in 2D and 3D scenes. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 20(6):577–589, June 1998.

[9] K. Kanatani. *Statistical Optimization for Geometric Computation: Theory and Practice*. Elsevier Science, Amsterdam, 1996.

[10] K. Kanatani. Statistical optimization and geometric inference in computer vision. *Transactions of the Royal Society A*, 356(1740):1303–1320, 1998.

[11] R. Kumar, P. Anandan, and K. Hanna. Direct recovery of shape from multiple views: a parallax based approach. In *Int. Conf. Pattern Recognition*, pages 685–688, 1994.

[12] A. Shashua. Algebraic functions for recognition. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 17(8):779–89, 1995.

[13] A. Shashua and S. Avidan. On the reprojection of 3D and 2D scenes without explicit model selection. In *European Conf. Computer Vision*, pages 936–950, Dublin, 2000.

[14] A. Shashua and N. Navab. Relative affine structure: Canonical model for 3d from 2d geometry and applications. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 18(9):873–883, 1996.

[15] A. Shashua and M. Werman. On the trilinear tensor of three perspective views and its underlying geometry. In *Int. Conf. Computer Vision*, Boston, MA, June 1995.

[16] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *Int. J. Computer Vision*, 9(2):137–54, 1992.

[17] P.H.S. Torr. Geometric motion segmentation and model selection. *Transactions of the Royal Society A*, 356(1740):1321–1340, 1998.

[18] P.H.S. Torr and A. Zisserman. Concerning bayesian motion segmentation, model averaging, matching and the trifocal tensor. In *European Conf. Computer Vision*, pages 511–527, Freiburg, 1998.

[19] B. Triggs. A fully projective error model for visual reconstruction. Unpublished. (Submitted to *ICCV'95 Workshop on Representations of Visual Scenes*).

[20] B. Triggs. The geometry of projective reconstruction I: Matching constraints and the joint image. Unpublished. (Submitted to *IJCV* in 1995).

[21] B. Triggs. Matching constraints and the joint image. In E. Grimson, editor, *Int. Conf. Computer Vision*, pages 338–43, Cambridge, MA, June 1995.

[22] B. Triggs. Plane + parallax, tensors and factorization. In *European Conf. Computer Vision*, pages 522–538, Dublin, 2000.

[23] D. Weinshall, P. Anandan, and M. Irani. From ordinal to euclidean reconstruction with partial scene calibration. In R. Koch and L. Van Gool, editors, *3D Structure from Multiple Images of Large-scale Environments SMILE'98*, pages 208–223. Springer-Verlag, 1998.

Chapter 5

Geometric Vision & Scene Reconstruction

This chapter contains four papers dealing with vision geometry and the mathematical theory of scene reconstruction. Thematically, they date from the period immediately after my PhD thesis.

Summary of paper 12, "Critical Motions for Auto-Calibration When Some Intrinsic Parameters Can Vary"

This paper published in the Journal of Mathematical Imaging and Vision [KTÅ00] represents the journal version of a collaboration begun several years earlier [KT99] with Fredrik Kahl and Kalle Åström of Lund University, on characterizing the algebraic critical sets for visual scene reconstruction and camera calibration. It deals specifically with the problem of "autocalibration" — the recovery of camera calibration using only weak information about the camera or scene, such as the fact that certain of the camera's intrinsic parameters remain unchanged during a motion. A "critical configuration" is a class of motions or a set of parameters for which autocalibration is intrinsically impossible owing to symmetries or degeneracies that weaken the structure of the set of equations used for autocalibration, and a "critical set" is the set of all such motions or parameters in the parameter space. The paper attacks the problem of characterizing the critical sets for several different autocalibration problems using a combination of projective geometry, geometric intuition and techniques from computational algebraic geometry (Gröbner basis methods). The characterization given is sometimes implicit, but in certain cases, notably two view problems where the only unknown internal camera parameters are the focal lengths, we are able to give a detailed characterization and explicit algorithms.

Summary of paper 13, "Le Calcul de Pose: de nouvelles méthodes matricielles"

This paper derives some new algebraic resultant based methods for the classical problem of determining the 3D position and orientation of a calibrated camera given several 3D points with known coordinates and their images. Methods for 3 points (4 solutions) and 4 or more points (1 solution)

are developed. The paper appeared in the 2002 French national conference Reconnaissance des Formes et Intelligence Artificielle [AQT02].

Summary of paper 14, "Plane + Parallax, Tensors and Factorization"

Published in the 2000 European Conference on Computer Vision [Tri00], this paper relates the tensorial approach to vision geometry that was developed at length in my PhD thesis, to the "plane + parallax" formulation where all geometric quantities are referred to a 3D reference plane and to heights or distances measured relative to this plane. The advantage of the plane + parallax point of view is that it simplifies many formulae and provides a good deal of valuable geometric intuition. The paper gives plane + parallax parametrizations of the main projective-geometric objects (points, lines, planes, *etc*.), rederives the geometric matching constraints (intrinsic tensorial relationships linking several images of an observed 3D entity) in this framework, and finishes with a projective factorization based scene reconstruction method based on the plane + parallax framework. The development takes place in a projective frame in which the reference plane is placed at infinity — an algebraic trick that simplifies matters by reducing the initial camera geometry to one in which the (projectively warped) camera motion is nominally pure-translational.

Summary of paper 15, "Bundle Adjustment — A Modern Synthesis"

This long paper (almost a small monograph) [TMHF00] presents an extended survey of the technological state of the art in Bundle Adjustment circa mid-2000. The paper expands on a panel session that we organized on this subject at our ICCV'99 workshop "Vision Algorithms: Theory and Practice", so the paper was published in the post-workshop proceedings of the same name [TZS00]. "Bundle Adjustment" is the name given in photogrammetry to simultaneous optimization of an estimated structure and motion (*i.e*. joint refinement of the estimated 3D scene parameters together with the intrinsic and extrinsic camera parameters). It is a central step in modern high-accuracy scene reconstruction methods, but when a complex scene is being reconstructed from many images, the resulting optimization problem can become very large and computation-intensive. However, the intrinsic sparse structure of the problem — sometimes several layers of structure — can be exploited to improve the efficiency of the solution. There is a wide choice of possible optimization methods and exact and approximate linear system solvers, and an extended discussion of this choice is given in the paper. One also has to consider issues such as overall problem formulation and parametrization, intrinsic degeneracies ("gauge freedoms") linked to coordinate system freedoms, and robustness, outliers, solution validation and quality control. The paper gives advice on all of these issues. It also contains historical and numerical appendices, a glossary and a bibliography listing the most notable previous papers in this area. One of our main aims in writing this paper was to reduce the amount of duplication of effort and rediscovery of known results, by providing the computer vision community with a point of reference into the extensive and often rather inaccessible photogrammetry literature on this subject.

Critical Motions for Auto-Calibration When Some Intrinsic Parameters Can Vary*

Fredrik Kahl *fredrik@maths.lth.se*
Bill Triggs *Bill.Triggs@inrialpes.fr*
Kalle Åström *kalle@maths.lth.se*

Centre for Mathematical Sciences, Mathematics (LTH),
P.O. Box 118, SE-221 00 Lund, SWEDEN

INRIA Rhône-Alpes,
655 avenue de l'Europe, Montbonnot, 38330, FRANCE

Abstract

Auto-calibration is the recovery of the full camera geometry and Euclidean scene structure from several images of an unknown 3D scene, using rigidity constraints and partial knowledge of the camera intrinsic parameters. It fails for certain special classes of camera motion. This paper derives necessary and sufficient conditions for unique auto-calibration, for several practically important cases where some of the intrinsic parameters are known (e.g. skew, aspect ratio) and others can vary (e.g. focal length). We introduce a novel subgroup condition *on the camera calibration matrix, which helps to systematize this sort of auto-calibration problem. We show that for subgroup constraints, criticality is independent of the exact values of the intrinsic parameters and depends only on the camera motion. We study such* critical motions *for arbitrary numbers of images under the following constraints: vanishing skew, known aspect ratio and full internal calibration modulo unknown focal lengths. We give explicit, geometric descriptions for most of the singular cases. For example, in the case of unknown focal lengths, the only critical motions are: (i) arbitrary rotations about the optical axis and translations, (ii) arbitrary rotations about at most two centres, (iii) forward-looking motions along an ellipse and/or a corresponding hyperbola in an orthogonal plane. Some practically important special cases are also analyzed in more detail.*

1 Introduction

One of the core problems in computer vision is the recovery of 3D scene structure and camera motion from a set of images. However, for certain configurations there are inherent ambiguities. This kind of problem was already studied in optics in the early 19th century, for example, by Vieth in 1818 and Muller in 1826. Pioneering work on the subject was also done by Helmholtz. See [13] for references. One well-studied ambiguity is when the visible features lie on a special surface, called a **critical surface**, and the cameras have a certain position relative to the surface. Critical surfaces or "gefährlicher Ort" were studied by Krames [21] based on a monograph from 1880 on quadrics [32]. See also the book by Maybank [24] for a more recent treatment. Another well-known ambiguity is that when using projective image measurements, it is only possible to recover the scene up to an

*Appeared in J. Mathematical Imaging & Vision, 13(2):131–146, Oct 2000.
Work supported by the ESPRIT Reactive LTR project 21914, CUMULI

unknown projective transformation [8, 10, 35]. Additional scene, motion or calibration constraints are required for a (scaled) Euclidean reconstruction. **Auto-calibration** uses qualitative constraints on the camera calibration, e.g. vanishing skew or unit aspect ratio, to reduce the projective ambiguity to a similarity. Unfortunately, there are situations when the auto-calibration constraints may lead to several possible Euclidean reconstructions. In this paper, such degeneracies are studied under various auto-calibration constraints.

In general it is possible to recover Euclidean scene information from $m \geq 3$ images by assuming constant but unknown intrinsic parameters of a moving projective camera [26, 7]. Several practical algorithms have been developed [39, 2, 30]. Some of the intrinsic parameters may even vary, e.g. the focal length [31], or the focal length and the principal point [14]. In [29, 15] it was shown that vanishing skew suffices for a Euclidean reconstruction. Finally in, [16] it was shown that given at least 8 images it is sufficient if just one of the intrinsic parameters is known to be constant (but otherwise unknown).

However, for certain camera motions, these auto-calibration constraints are *not* sufficient [42, 1, 40]. A complete categorization of these **critical motions** in the case of constant intrinsic parameters was given by Sturm [36, 37]. The uniformity of the constant-intrinsic constraints makes this case relatively simple to analyze. But it is also somewhat unrealistic: It is often reasonable to assume that the skew actually vanishes whereas focal length often varies between images. While the case of constant parameters is practically solved, much less is known for other auto-calibration constraints. In [43], additional scene and calibration constraints are used to resolve ambiguous reconstructions, caused by a fixed axis rotation. The case of two cameras with unknown focal lengths is studied in [12, 28, 4, 20]. For the general unknown focal length case, Sturm [38] has independently derived results similar to those presented here and in [20, 19].

In this paper, we generalise the work of Sturm [37] by relaxing the constraint constancy on the intrinsic parameters. We show that for a large class of auto-calibration constraints, the degeneracies are independent of the specific values of the intrinsic parameters. Therefore, it makes sense to speak of *critical motions* rather than critical configurations. We then derive the critical motions for various auto-calibration constraints. The problem is formulated in terms of projective geometry and the absolute conic. We start with fully calibrated cameras, and then continue with cameras with unknown and possibly varying focal length, principal point, and finally aspect ratio. Once the general description of the degenerate motions has been completed, some particular motions frequently occurring in practice are examined in more detail.

This paper is organized as follows. In Section 2 some background on projective geometry for vision is presented. Section 3 gives a formal problem statement and reformulates the problem in terms of the absolute conic. In Section 4, our general approach to solving the problem is presented, and Section 5 derives the actual critical motions under various auto-calibration constraints. Some particular motions are analyzed in Section 6. In order to give some practical insight of critical and near-critical motions, some experiments are presented in Section 7. Finally, Section 8 concludes.

2 Background

In this section, we give a brief summary of the modern projective formulation of visual geometry. Also, some basic concepts in projective and algebraic geometry are introduced. For further reading, see [6, 24, 33].

A **perspective (pinhole) camera** is modeled in homogeneous coordinates by the projection equation

$$\boldsymbol{x} \simeq \boldsymbol{P}\,\mathbf{X} \tag{1}$$

where $\mathbf{X} = (X, Y, Z, W)^T$ is a 3D world point, $\boldsymbol{x} = (x, y, z)^T$ is its 2D image, $\boldsymbol{P}$ is the 3×4 camera **projection matrix** and $\simeq$ denotes equality up to scale. Homogeneous coordinates are used for both image and object coordinates. In a Euclidean frame, $\boldsymbol{P}$ can be factored, using a QR-decomposition,

cf. [9], as

$$\boldsymbol{P} \simeq \boldsymbol{K}\,[\boldsymbol{R}\,|\,-\boldsymbol{R}\mathbf{t}] \quad \text{where} \quad \boldsymbol{K} = \begin{bmatrix} f & f\,s & u_0 \\ 0 & f\,\gamma & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

Here the **extrinsic parameters** $(\boldsymbol{R}, \mathbf{t})$ denote a 3×3 rotation matrix and a 3×1 translation vector, which encode the pose of the camera. The columns of $\boldsymbol{R} = [\mathbf{r}_1\mathbf{r}_2\mathbf{r}_3]$ define an orthogonal base. The standard base is defined by $\mathbf{e}_1 = (1,0,0)^T, \mathbf{e}_2 = (0,1,0)^T, \mathbf{e}_3 = (0,0,1)^T$. The **intrinsic parameters** in the **calibration matrix** $\boldsymbol{K}$ encode the camera's internal geometry: f denotes the **focal length**, γ the **aspect ratio**, s the **skew** and (u_0, v_0) the **principal point**.

A camera for which $\boldsymbol{K}$ is unknown is said to be **uncalibrated**. It is well-known that for uncalibrated cameras, it is only possible to recover the $3D$ scene and the camera poses up to unknown projective transformation [8, 10, 35]. This follows directly from the projection equation (1): Given one set of camera matrices and $3D$ points that satisfies (1), another reconstruction can be obtained from

$$\boldsymbol{P}\,\mathbf{X} = (\boldsymbol{P}\,T)\,(T^{-1}\,\mathbf{X}) = \tilde{\boldsymbol{P}}\,\tilde{\mathbf{X}},$$

where T is a non-singular 4×4 matrix corresponding to a projective transformation of $\mathbb{P}^3$.

A quadric in $\mathbb{P}^n$ is defined by the quadratic form

$$\boldsymbol{X}^T\boldsymbol{Q}\boldsymbol{X} = 0,$$

where $\boldsymbol{Q}$ denotes a $(n+1)\times(n+1)$ symmetric matrix and $\boldsymbol{X}$ denotes homogeneous point coordinates. The **dual** is a quadric envelope, given by

$$\boldsymbol{\Pi}^T\boldsymbol{Q}^*\boldsymbol{\Pi} = 0, \tag{3}$$

where $\boldsymbol{\Pi}$ denotes homogeneous coordinates for hyper-planes of dimension $n-1$ that are tangent to the quadric. For non-singular matrices, it can be shown that $\boldsymbol{Q} \simeq (\boldsymbol{Q}^*)^{-1}$ (see [33] for a proof). A quadric with a non-singular matrix is said to be **proper**. Quadrics with no real points are called **virtual**. In the plane, $n = 2$, quadrics are called **conics**. We will use $\boldsymbol{C}$ for the 3×3 matrix that defines the conic points $\boldsymbol{x}^T\boldsymbol{C}\boldsymbol{x} = 0$ and $\boldsymbol{C}^*$ for its dual that defines the envelope of tangent lines $\boldsymbol{l}^T\boldsymbol{C}^*\boldsymbol{l} = 0$ (where $\boldsymbol{C}^* \simeq \boldsymbol{C}^{-1}$). The image of a quadric in 3D-space is a conic, i.e. the silhouette of a 3D quadric is projected to a conic curve. This can be expressed in envelope forms as

$$\boldsymbol{C}^* \simeq \boldsymbol{P}\boldsymbol{Q}^*\boldsymbol{P}^T. \tag{4}$$

Projective geometry encodes only cross ratios and incidences. Properties like parallelism and angles are not invariant under different projective coordinate systems. An affine space, where properties like parallelism and ratios of lengths are preserved, can be embedded in a projective space by singling out a **plane at infinity** Π_∞. The points on Π_∞ are called points at infinity and be interpreted as direction vectors. In $\mathbb{P}^3$, Euclidean properties, like angles and lengths, are encoded by singling out a proper, virtual conic on Π_∞. This **absolute conic** Ω_∞ gives scalar products between direction vectors. Its dual, the **dual absolute quadric** Ω^*_∞, gives scalar products between plane normals. Ω^*_∞ is a 4×4 symmetric rank 3 positive semidefinite matrix, where the coordinate system is normally chosen such that $\Omega^*_\infty = \mathrm{diag}(1,1,1,0)$. Π_∞ is Ω^*_∞'s unique null vector: $\Omega^*_\infty\,\Pi_\infty = 0$. The **similarities** or scaled **Euclidean transformations** in projective space are exactly those transformations that leave Ω_∞ invariant. The transformations that leave Π_∞ invariant are the **affine transformations**. The different forms of the absolute conic will be abbreviated to (D)AC for (Dual) Absolute Conic.

Given image conics in several images, there may or may not be a 3D quadric having them as image projections. The constraints which guarantee this in two images are called the **Kruppa constraints** [22]. In the two-image case, these constraints have been successfully applied in order to derive the critical sets, e.g. [28]. For the more general case of multiple images, the projection equation given by (4) can be used for each image separately.

3 Problem Formulation

The problem of auto-calibration is to find the intrinsic camera parameters $(\boldsymbol{K}_i)_{i=1}^m$, where m denotes the number of camera positions. In general, auto-calibration algorithms proceed from a projective reconstruction of the camera motion. In order to auto-calibrate, some constraints have to be enforced on the intrinsic parameters, e.g. vanishing skew and/or unit aspect ratio. Thus, we require that the calibration matrices should belong to some proper subset $\mathcal{G}$ of the group $\mathcal{K}$ of 3×3 upper triangular matrices. Once the projective reconstruction and the intrinsic parameters are known, Euclidean structure and motion are easily computed.

For a general set of scene points seen in two or more images, there is a unique projective reconstruction. However, certain special configurations, known as critical surfaces, give rise to additional ambiguous solutions. For two cameras, the critical configurations occur only if both camera centres and all scene points lie on a ruled quadric surface [24]. Furthermore, when an alternative reconstruction exists, then there will always exist a third distinct reconstruction. For more than two cameras, the situation is less clear. In [25], it is proven that when six scene points and any number of camera centres lie on a ruled quadric, then there are three distinct reconstructions. If there are other critical surfaces is an open problem.

We will avoid critical surfaces by assuming unambiguous recovery of projective scene structure and camera motion. In other words, the camera matrices and the 3D scene are considered to be known up to an unknown projective transformation. We formulate the auto-calibration problem as follows: If all that is known about the camera motions and calibrations is that each calibration matrix $\boldsymbol{K}_i$ lies in some given constraint set $\mathcal{G} \subset \mathcal{K}$, when is a unique auto-calibration possible? More formally:

Problem 3.1. *Let $\mathcal{G} \subset \mathcal{K}$. Then, given the true camera projections $(\boldsymbol{P}_i)_{i=1}^m$, where $\boldsymbol{P}_i = \boldsymbol{K}_i[\,\boldsymbol{R}_i \,|\, -\boldsymbol{R}_i\mathbf{t}_i\,]$ and $\boldsymbol{K}_i \in \mathcal{G}$, is there any projective transformation T (not a similarity) such that $\tilde{\boldsymbol{P}}_i \simeq \boldsymbol{P}_i T$ has decomposition $\tilde{\boldsymbol{P}}_i = \tilde{\boldsymbol{K}}_i[\,\tilde{\boldsymbol{R}}_i \,|\, -\tilde{\boldsymbol{R}}_i\tilde{\mathbf{t}}_i\,]$ with calibration matrices $\tilde{\boldsymbol{K}}_i$ lying in $\mathcal{G}$?*

Without constraints on the intrinsic parameters T can be chosen arbitrarily, so auto-calibration is impossible. Also, T is only defined modulo a similarity,

$$T \to T \begin{bmatrix} \boldsymbol{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix},$$

as such transformations leave $\boldsymbol{K}$ in the decomposition $\boldsymbol{P} = \boldsymbol{K}[\,\boldsymbol{R} \,|\, -\boldsymbol{R}\mathbf{t}\,]$ invariant. Based on the above problem formulation, we can define precisely what is meant by a motion being critical.

Definition 3.1. Let $\mathcal{G} \subset \mathcal{K}$ and let $(\boldsymbol{P}_i)_{i=1}^m$ and $(\tilde{\boldsymbol{P}}_i)_{i=1}^m$ denote two projectively related motions, with calibration matrices $(\boldsymbol{K}_i)_{i=1}^m$ and $(\tilde{\boldsymbol{K}}_i)_{i=1}^m$, respectively. If the two motions are not related by a Euclidean transformation and $\boldsymbol{K}_i, \tilde{\boldsymbol{K}}_i \in \mathcal{G}$, they are said to be *critical* with respect to $\mathcal{G}$.

A motion is critical if there exists an alternative projective motion satisfying the auto-calibration constraints. Without any additional assumptions, it is not possible to tell which motion is the true one. One natural additional constraint is that the reconstructed $3D$ structure should lie in front of all cameras. In many (but by no means all) cases this reduces the ambiguity, but it depends on which 3D points are observed.

According to (4) the image of the absolute conic Ω_∞ is,

$$\omega_i^* \simeq \boldsymbol{P}_i \Omega_\infty^* \boldsymbol{P}_i^T \simeq \boldsymbol{K}_i[\,\boldsymbol{R}_i \,|\, -\boldsymbol{R}_i\mathbf{t}_i\,] \begin{bmatrix} \boldsymbol{I} & 0 \\ 0 & 0 \end{bmatrix} [\,\boldsymbol{R}_i \,|\, -\boldsymbol{R}_i\mathbf{t}_i\,]^T \boldsymbol{K}_i^T = \boldsymbol{K}_i\boldsymbol{K}_i^T. \tag{5}$$

Thus, knowing the calibration of a camera is equivalent to knowing its image of Ω_∞. Also, if there is a projectively related motion $(\tilde{\boldsymbol{P}}_i)_{i=1}^m$, then the false image of the true absolute conic is the true image of a "false" absolute conic:

$$\tilde{\omega}_i^* \simeq \tilde{\boldsymbol{P}}_i \Omega_\infty^* \tilde{\boldsymbol{P}}_i^T = \tilde{\boldsymbol{P}}_i T^{-1} T \Omega_\infty^* T^T T^{-T} \tilde{\boldsymbol{P}}_i = \boldsymbol{P}_i \Omega_f^* \boldsymbol{P}_i^T,$$

where $\Omega_f^* = T\Omega_\infty^* T^T$ is some dual, virtual quadric of rank 3. This observation allows us to eliminate the "false" motion $(\tilde{\boldsymbol{P}}_i)_{i=1}^m$ from the problem and work only with the true Euclidean motion, but with a false absolute dual quadric Ω_f^*.

Problem 3.2. *Let $\mathcal{G} \subset \mathcal{K}$. Then, given the true motion $(\boldsymbol{P}_i)_{i=1}^m$, where $\boldsymbol{P}_i = \boldsymbol{K}_i[\boldsymbol{R}_i \,|\, -\boldsymbol{R}_i\mathbf{t}_i]$ and $\boldsymbol{K}_i \in \mathcal{G}$, is there any other proper, virtual conic Ω_f^*, different from Ω_∞^*, such that $\boldsymbol{P}_i\Omega_f^*\boldsymbol{P}_i^T \simeq \tilde{\boldsymbol{K}}_i\tilde{\boldsymbol{K}}_i^T$, where $\tilde{\boldsymbol{K}}_i \in \mathcal{G}$?*

Given only a 3D projective reconstruction derived from uncalibrated images, the true Ω_∞ is not distinguished in any way from any other proper, virtual planar conic in projective space. In fact, given any such potential conic Ω_f^*, it is easy to find a 'rectifying' projective transformation that converts it to the Euclidean DAC form $\Omega_\infty^* = \mathrm{diag}(1,1,1,0)$ and hence defines a false Euclidean structure. To recover the true structure, we need constraints that single out the true Ω_∞ and Π_∞ from all possible false ones. Thus, ambiguity arises whenever the images of some non-absolute conic satisfy the auto-calibration constraints. We call such conics **potential absolute conics** or **false absolute conics**. They are in one-to-one correspondence with possible false Euclidean structures for the scene.

A natural question is whether the problem is dependent on the actual values of the intrinsic parameters. We will show that this is not the case whenever the set $\mathcal{G}$ is a proper subgroup of $\mathcal{K}$. Fortunately, according to the following easy lemma, most of the relevant auto-calibration constraints are **subgroup conditions**.

Lemma 3.1. *The following constrained camera matrices form proper subgroups of the 3×3 upper triangular matrices $\mathcal{K}$:*

(i) Zero skew, i.e. $s = 0$.

(ii) Unit aspect ratio, i.e. $\gamma = 1$.

(iii) Vanishing principal point, i.e. $(u_0, v_0) = (0, 0)$.

(iv) Unit focal length, i.e. $f = 1$.

(v) Combinations of the above conditions.

Independence of the values of the intrinsic camera parameters is shown as follows:

Lemma 3.2. *Let $G_i \in \mathcal{G}$ for $i = 1, ..., m$, where $\mathcal{G}$ is a proper subgroup of $\mathcal{K}$. Then, the motion $(\boldsymbol{P}_i)_{i=1}^m$ is critical w.r.t. $\mathcal{G}$ if and only if the motion $(G_i\boldsymbol{P}_i)_{i=1}^m$ is critical w.r.t. $\mathcal{G}$.*

Proof. If $(\boldsymbol{P}_i)_{i=1}^m$ is critical with the alternative motion $(\tilde{\boldsymbol{P}}_i)_{i=1}^m$ and calibrations $\boldsymbol{K}_i, \tilde{\boldsymbol{K}}_i \in \mathcal{G}$, then clearly $(G_i\boldsymbol{P}_i)_{i=1}^m$ and $(G_i\tilde{\boldsymbol{P}}_i)_{i=1}^m$ are also critical, because $G_i\boldsymbol{K}_i, G_i\tilde{\boldsymbol{K}}_i \in \mathcal{G}$ by the closure of $\mathcal{G}$ under multiplication. The converse also holds with G_i^{-1}, by the closure of $\mathcal{G}$ under inversion. ∎

Camera matrices with prescribed parameters do not in general form a subgroup of $\mathcal{K}$, but it suffices for them to be of the more general form $\boldsymbol{K}_0\boldsymbol{K}$ where $\boldsymbol{K}_0$ is a known matrix and $\boldsymbol{K}$ belongs to a proper subgroup of $\mathcal{K}$. For example, the set of all camera matrices with known focal length f has the form

$$\begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & * & * \\ 0 & * & * \\ 0 & 0 & 1 \end{bmatrix}.$$

The invariance with respect to calibration parameters simplifies things, especially if one chooses $G_i = \boldsymbol{K}_i^{-1}$ for $i = 1, ..., m$. With this in mind, we restrict our attention to proper subgroups of $\mathcal{K}$ and formulate the problem as follows.

Problem 3.3. *Let $\mathcal{G} \subset \mathcal{K}$ be a proper subgroup. Then, given the true motion $(\boldsymbol{P}_i)_{i=1}^m$ for calibrated cameras, where $\boldsymbol{P}_i = [\,\boldsymbol{R}_i \,|\, -\boldsymbol{R}_i\mathbf{t}_i\,]$, is there any other false absolute conic Ω_f^*, different from Ω_∞^*, such that*

$$[\,\boldsymbol{R}_i \,|\, -\boldsymbol{R}_i\mathbf{t}_i\,]\Omega_f^*[\,\boldsymbol{R}_i \,|\, -\boldsymbol{R}_i\mathbf{t}_i\,]^T \simeq \tilde{\boldsymbol{K}}_i\tilde{\boldsymbol{K}}_i^T,$$

where $\tilde{\boldsymbol{K}}_i \in \mathcal{G}$?

4 Approach

We want to explicitly characterize the critical motions (relative camera placements) for which particular auto-calibration constraints are insufficient to uniquely determine Euclidean 3D structure. We assume that projective structure is available. Alternative Euclidean structures correspond one-to-one with possible locations for a potential absolute conic in $\mathbb{P}^3$. Initially, any proper virtual projective plane conic is potentially absolute, so we look for such conics Ω^* whose images also satisfy the given auto-calibration constraints. Ambiguity arises *if and only if* more than one such conic exists. We work with the true camera motion in a Euclidean frame where the true absolute conic Ω_∞ has its standard coordinates.

Several general invariance properties help to simplify the problem:

Calibration invariance: As shown in the previous section, if the auto-calibration constraints are subgroup conditions, the specific parameter values are irrelevant. Hence, for the purpose of deriving critical motions, we are free to assume that the cameras are in fact secretly calibrated, $\boldsymbol{K}_i = \boldsymbol{I}$, even though we do not assume that we *know* this. (All that we actually know is $\boldsymbol{K}_i \in \mathcal{G}$, which does not allow some image conics $\omega_i^* \neq \boldsymbol{I}$ to be excluded outright).

Rotation invariance: For *known-calibrated* cameras, $\boldsymbol{K}_i$ can be set to identity, and thus the image $\omega_i^* = \boldsymbol{I}$ of any false AC must be identical to the image of the true one. Since

$$\boldsymbol{P}_i\Omega_f^*\boldsymbol{P}_i^T \simeq \boldsymbol{I} \quad \Rightarrow \quad \boldsymbol{R}\boldsymbol{P}_i\Omega_f^*\boldsymbol{P}_i^T\boldsymbol{R}^T \simeq \boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I},$$

hold for *any* rotation $\boldsymbol{R}$, the image ω_i^* is invariant to camera rotations. Hence, *criticality depends only on the camera centres, not on their orientations.* More generally, any camera rotation that leaves the auto-calibration constraints intact is irrelevant. For example, arbitrary rotations about the optical axis and 180° flips about any axis in the optical plane are irrelevant if (a, s) is either $(1, 0)$ or unconstrained, and (u_0, v_0) is either $(0, 0)$ or unconstrained.

Translation invariance: For true or false absolute conics on the plane at infinity, translations are irrelevant so criticality depends only on camera orientation.

In essence, Euclidean structure recovery in projective space is a matter of parameterizing all of the possible proper virtual plane conics, then using the auto-calibration constraints on their images to algebraically eliminate parameters until only the unique true absolute conic remains. More abstractly, if $\boldsymbol{C}$ parameterizes the possible conics and $\mathbf{X}$ the camera geometries, the constraints cut out some algebraic variety in $(\boldsymbol{C}, \mathbf{X})$ space. A constraint set is useful for Euclidean structure from motion recovery only if this variety generically intersects the subspaces $\mathbf{X} = \mathbf{X}_0$ in one (or at most a few) points $(\boldsymbol{C}, \mathbf{X}_0)$, as each such intersection represents an alternative Euclidean structure for the reconstruction from that camera geometry. A set of camera poses $\mathbf{X}$ is **critical** for the constraints if it has exceptionally (e.g. infinitely) many intersections.

Potential absolute conics can be represented in several ways. The following parameterizations have all proven relatively tractable:

(i) Choose a Euclidean frame in which Ω_f^* is diagonal, and express all camera poses relative this frame [36, 37]. This is symmetrical with respect to all the images and usually gives the simplest equations. However, in order to find explicit inter-image critical motions, one must revert to camera-based coordinates which is sometimes delicate. The cases of a finite false absolute conic and a false conic on the plane at infinity must also be treated separately, e.g. $\Omega_f^* = \mathrm{diag}(d_1, d_2, d_3, d_4)$ with either d_3 or d_4 zero.

(*ii*) Work in the first camera frame, encoding Ω_f^* by its first image ω_1^* and supporting plane $(\mathbf{n}^T, 1)$. Subsequent images $\omega_i^* \simeq \boldsymbol{H}_i\, \omega_1^*\, \boldsymbol{H}_i^T$ are given by the inter-image homographies $\boldsymbol{H}_i = \boldsymbol{R}_i + \mathbf{t}_i\, \mathbf{n}^T$ where $(\boldsymbol{R}_i, \mathbf{t}_i)$ is the i^{th} camera pose. The output is in the first camera frame and remains well-defined even if the conic tends to infinity, but the algebra required is significantly heavier.
(*iii*) Parameterize Ω_f^* implicitly by two images ω_1^*, ω_2^* subject to the Kruppa constraints. In the two-image case this approach is both relatively simple and rigorous — two proper virtual dual image conics satisfy the Kruppa constraints if and only if they define a (pair of) corresponding 3D potential absolute conics — but it does not extend so easily to multiple images.

The derivations below are mainly based on method (*i*) .

5 Critical Motions

In this section, the varieties of critical motions are derived. In most situations, the problem is solved in two separate cases. One is when there are potential absolute conics on the plane at infinity, Π_∞, and the other one is conics outside Π_∞. If the potential conics are all on Π_∞, it is still possible to recover Π_∞ and thereby obtain an affine reconstruction. Otherwise, the recovery of affine structure is ambiguous, and we say that the motion is **critical with respect to affine reconstruction**.

The following constraints on the camera calibration are considered:

(i) known intrinsic parameters,

(ii) unknown focal lengths, but the other intrinsic parameters known,

(iii) known skew and aspect ratio.

These constraints form a natural hierarchy and they are perhaps the most interesting ones from a practical point of view. In Section 3, it was shown in that it is sufficient to study the normalized versions of the auto-calibration constraints, since critical motions are independent of the specific values of the intrinsic parameters. That is, when some of the intrinsic parameters are *known*, e.g. the principal point is $(10, 20)$, we may equivalently analyze the case of principal point set to $(0, 0)$. The corresponding camera matrices give rise to subgroup conditions according to Lemma 3.1.

5.1 Known intrinsic parameters

We start with fully calibrated perspective cameras. The results may not come as a surprise, but it is important to know that there are no other possible degenerate configurations.

Proposition 5.1. *Given projective structure and calibrated perspective cameras at $m \geq 3$ distinct finite camera centres, Euclidean structure can always be recovered uniquely. With $m = 2$ distinct camera centres, there is always exactly a twofold ambiguity.*

Proof. Assuming that the cameras have $\boldsymbol{K} = \boldsymbol{I}$ does not change the critical motions. The camera orientations are irrelevant because any false absolute conic must have the same (rotation invariant) images as the true one. Calibrated cameras never admit false absolute conics on Π_∞, as the (known) visual cone of each image conic can intersect Π_∞ in only one conic, which is the true absolute conic. Therefore, consider a finite absolute conic Ω_f^*, with supporting plane outside Π_∞. As all potential absolute conics are proper, virtual and positive semi-definite [34, 37], a Euclidean coordinate system can be chosen such that Ω_f^* has supporting plane $z = 0$, and matrix coordinates

$$\Omega_f^* = \begin{bmatrix} d_1 & 0 & 0 & 0 \\ 0 & d_2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & d_4 \end{bmatrix}.$$

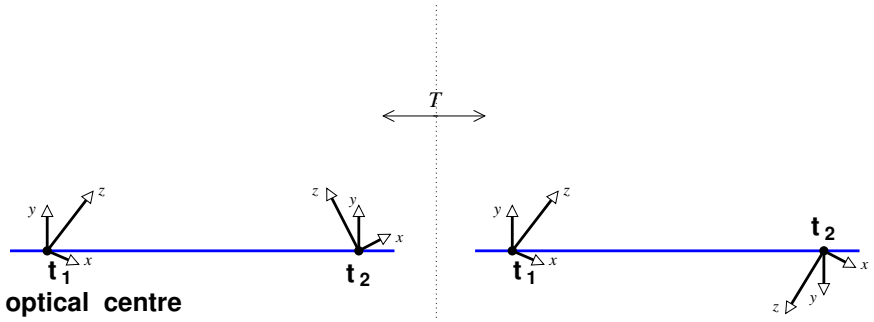


Figure 1: A twisted pair of reconstructions.

Since the cameras are calibrated, $\boldsymbol{K} = \boldsymbol{I}$, and their orientations are irrelevant, $\boldsymbol{R} = \boldsymbol{I}$, the conic projection (4) in each camera becomes

$$[\boldsymbol{I} \,|\, -\mathbf{t}]\, \Omega_f^* \, [\boldsymbol{I} \,|\, -\mathbf{t}]^T \simeq \boldsymbol{I} \quad \Leftrightarrow \quad \begin{bmatrix} d_1 + d_4 t_1^2 & d_4 t_1 t_2 & d_4 t_1 t_3 \\ d_4 t_1 t_2 & d_2 + d_4 t_2^2 & d_4 t_2 t_3 \\ d_4 t_1 t_3 & d_4 t_2 t_3 & d_4 t_3^2 \end{bmatrix} \simeq \boldsymbol{I}.$$

As the conic should be proper, both $d_4 \neq 0$ and $t_3 \neq 0$, which gives $t_1 = t_2 = 0$. Thus the only solutions are $\mathbf{t}_\pm = (0, 0, \pm z)$ and $\Omega_f^* \simeq \mathrm{diag}(1, 1, 0, 1/z^2)$ for some $z > 0$. Hence, ambiguity implies that there are at most two camera centres, and the false conic is a circle of imaginary radius $i\,z$, centred in the plane bisecting the two camera centres. ■

In the two-image case, the improper self-inverse projective transformation

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & z \\ 0 & 0 & 1/z & 0 \end{bmatrix}$$

interchanges the true Ω_∞^* and the false Ω_f^*, according to

$$T\, \Omega_f^*\, T^T \simeq \Omega_\infty^*$$

and takes the two projection matrices $\boldsymbol{P}_\pm = \boldsymbol{R}_\pm[\boldsymbol{I} \,|\, -\mathbf{t}_\pm]$ to

$$\boldsymbol{P}_- T^{-1} = \boldsymbol{P}_- \quad \text{and} \quad \boldsymbol{P}_+ T^{-1} = -\boldsymbol{R}_+ \begin{bmatrix} -1 & & \\ & -1 & \\ & & 1 \end{bmatrix} | \, -\mathbf{t}_+ \Big].$$

While the first camera remains fixed, the other has rotated 180° about the axis joining the two centres. This twofold ambiguity corresponds exactly to the well-known **twisted pair** duality [23, 18, 27]. The geometry of the duality is illustrated in Figure 1.

The 'twist' T represents a very strong projective deformation that cuts the scene in half, moving the plane between the cameras to infinity, see Figure 2. By considering twisted vs. non-twisted optical ray intersections, one can also show that it reverses the relative signs of the depths, so for one of the solutions the structure will appear to be behind one camera, cf. [17]. To conclude, Proposition 5.1 states that *any* two-view geometry has a 'twisted pair' projective involution symmetry and *any* camera configuration with three or more camera centres has a unique projective-to-Euclidean upgrade.

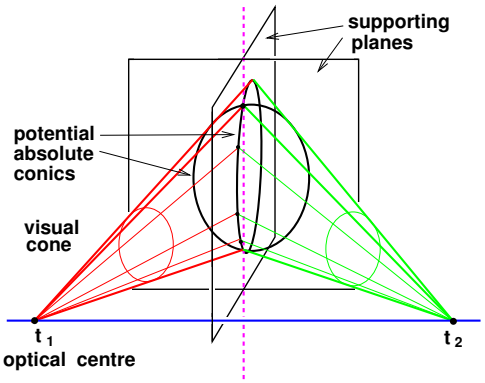


Figure 2: Intersecting the visual cones of two image conics satisfying the Kruppa constraints generates a pair of 3D conics, corresponding to the two solutions of the twisted pair duality.

5.2 Unknown focal lengths

In the case of two images and internally calibrated cameras modulo unknown focal lengths, it is in general possible to recover Euclidean structure. Since we know that the solutions always occur in twisted pairs (which can be disambiguated using the positive depth constraint), it is more relevant to characterize the motions for which there are solutions other than the twisted pair duality. Therefore, the two-camera case will be dealt with separately, after having derived the critical motions for arbitrary many images.

5.2.1 Many images

If all intrinsic parameters are known except for the focal lengths, the camera matrix can be assumed to be $\boldsymbol{K} = \mathrm{diag}(f, f, 1)$ which in turn implies that the image of a potential absolute conic satisfies

$$\omega^* = \boldsymbol{K}\boldsymbol{K}^T \simeq \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \text{for some } \lambda > 0. \tag{6}$$

We start with potential absolute conics on Π_∞.

Potential absolute conics on Π_∞
Let $\boldsymbol{C_f}$ denote a 3×3 matrix corresponding to a false absolute conic (in locus form) on the plane at infinity. Since $\boldsymbol{C_f}$ is not the true one, $\boldsymbol{C_f} \not\simeq \boldsymbol{I}$. The image of $\boldsymbol{C_f}$ is according to (4)

$$\omega \simeq \boldsymbol{R}\boldsymbol{C_f}\boldsymbol{R}^T. \tag{7}$$

Notice that criticality is independent of translation of the camera.

Two cameras are said to have the same **viewing direction** if their optical axes are parallel or anti-parallel.

Proposition 5.2. *Given Π_∞ and known skew, aspect ratio and principal point, then a motion is critical if and only if there is only one viewing direction.*

Proof. Choose coordinates in which camera 1 has orientation $\boldsymbol{R}_1 = \boldsymbol{I}$. Suppose a motion is critical. According to (6) and (7), this implies that $\boldsymbol{C_f} \simeq \mathrm{diag}(1, 1, 1+\mu) = \boldsymbol{I} + \mu \mathbf{e}_3 \mathbf{e}_3^T$ for some $\mu > -1$. For camera 2, let $\boldsymbol{R}_2 = [\,\mathbf{r}_1\, \mathbf{r}_2\, \mathbf{r}_3\,]$ and apply (7),

$$\boldsymbol{I} + \mu \mathbf{r}_3 \mathbf{r}_3^T \simeq \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \nu \end{bmatrix}, \quad \text{for some } \nu > 0.$$

This implies that $\mathbf{r}_3 = \pm\mathbf{e}_3$, and in turn, $\boldsymbol{R}_2 = \begin{pmatrix} * & * & 0 \\ * & * & 0 \\ 0 & 0 & \pm 1 \end{pmatrix}$ which is equivalent to a fixed viewing direction of the camera. Conversely, suppose the viewing direction is fixed, which means that $\boldsymbol{R}_i = \begin{pmatrix} * & * & 0 \\ * & * & 0 \\ 0 & 0 & \pm 1 \end{pmatrix}$ for $i > 1$. Then, it is not possible to disambiguate between any of the potential absolute conics in the pencil $\boldsymbol{C}_{\boldsymbol{f}}(\mu) \simeq \boldsymbol{I} + \mu\mathbf{e}_3\mathbf{e}_3^T$, since $\boldsymbol{R}_i\boldsymbol{C}_{\boldsymbol{f}}\boldsymbol{R}_i^T = \boldsymbol{C}_{\boldsymbol{f}}$.

■

Potential absolute conics outside Π_∞
Assume we have a critical motion $(\boldsymbol{R}_i, \mathbf{t}_i)_{i=1}^m$ with the false dual absolute conic Ω_f^*. If the supporting plane for Ω_f^* is Π_∞, the critical motion is described by Proposition 5.2, so assume that Ω_f^* is outside Π_∞. As in the proof of Proposition 5.1, one can assume without loss generality that a Euclidean coordinate system has been chosen such that Ω_f^* has supporting plane $z = 0$, and matrix coordinates

$$\Omega_f^* = \begin{bmatrix} d_1 & 0 & 0 & 0 \\ 0 & d_2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & d_4 \end{bmatrix}.$$

The image of Ω_f^* is according to (4),

$$\boldsymbol{P}_i\Omega_f^*\boldsymbol{P}_i^T \simeq \omega_i^* \Leftrightarrow \boldsymbol{R}_i\boldsymbol{C}_{\boldsymbol{f}}\boldsymbol{R}_i^T \simeq \omega_i^*, \text{ where } \boldsymbol{C}_{\boldsymbol{f}} = \begin{bmatrix} d_1 & & \\ & d_2 & \\ & & 0 \end{bmatrix} + d_4\mathbf{t}_i\mathbf{t}_i^T. \tag{8}$$

A necessary condition for degeneracy is that $\boldsymbol{R}_i$ should diagonalize $\boldsymbol{C}_{\boldsymbol{f}}$ to the form (6), i.e. the matrix $\boldsymbol{C}_{\boldsymbol{f}}$ must have two equal eigenvalues. As it is always possible to find an orthogonal matrix that diagonalizes a real, symmetric matrix [5], all we need to do is to find out precisely when $\boldsymbol{C}_{\boldsymbol{f}}$ has two equal eigenvalues. Lemma A.1 in the Appendix characterizes matrices of this form.

Applying the lemma to $\boldsymbol{C}_{\boldsymbol{f}}$ in (8), with $\sigma_1 = d_1$, $\sigma_2 = d_2$, $\sigma_3 = 0$ and $\rho = d_4$, results in the following cases:

(i) If $d_1 \neq d_2$, then

  a. $t_1 = 0$ and $t_2^2 d_1 d_4 + t_3^2(d_1 - d_2)d_4 = (d_1 - d_2)d_1$, *or*
  b. $t_2 = 0$ and $t_1^2 d_2 d_4 + t_3^2(d_2 - d_1)d_4 = (d_2 - d_1)d_2$.

  These equations describe a motion on two planar conics for which the supporting planes are orthogonal. On the first plane, the conic is an ellipse, while on the other the conic is a hyperbola (depending on whether $d_1 > d_2$ or vice versa), see Figure 3.

(ii) If $d_1 = d_2$, then $t_1 = t_2 = 0$ and t_3 arbitrary.

Notice that the second alternative in case (ii) of Lemma A.1 does not occur, since it implies $\mathbf{t}^T\mathbf{e}_3 = 0$, making $\boldsymbol{C}_{\boldsymbol{f}}$ rank-deficient. Also, case (iii) is impossible, since $\sigma_3 = 0 = \sigma_1 = \sigma_2$.

It remains to find the rotations that diagonalize $\boldsymbol{C}_{\boldsymbol{f}}$. Since rotations around the optical axis are irrelevant, only the direction of the optical axis is significant. Suppose the optical axis is parameterized by the camera centre $\mathbf{t}$ and a direction $\mathbf{d}$, i.e. $\{\mathbf{t} + \lambda\mathbf{d} | \lambda \in \mathbb{R}\}$. Any point on the axis projects to the principal point,

$$[\boldsymbol{R} \,|\, -\boldsymbol{R}\mathbf{t}] \begin{bmatrix} \mathbf{t} + \lambda\mathbf{d} \\ 1 \end{bmatrix} \simeq \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \Leftrightarrow \mathbf{d} \simeq \boldsymbol{R}^T \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

The direction $\mathbf{d}$ should equal the third row of $\boldsymbol{R}$, which corresponds to the eigenvector of the single eigenvalue of $\boldsymbol{C}_{\boldsymbol{f}}$. Regarding the proof of Lemma A.1, it is not hard to see that the eigenvectors are $\mathbf{v} \simeq (0, t_2\, d_1, t_3\,(d_1 - d_2))$ and $\mathbf{v} \simeq (t_1\, d_2, 0, t_3\,(d_1 - d_2))$ in the two sub-cases in (i) above.

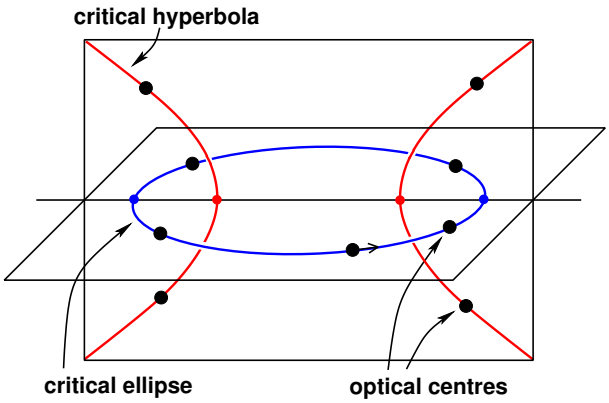


Figure 3: Two orthogonal planes, where one plane contains an ellipse and the other contains a hyperbola.

Geometrically, this means that the optical axis must be tangent to the conic at each position, as illustrated in Figure 4(b). Similarly in (ii), it is easy to derive that $\mathbf{v} = (0, 0, \pm 1)$, which means that the optical axis should be tangent to the translation direction, cf. Figure 4(c). An exceptional case is when $\boldsymbol{C}_f$ has a triple eigenvalue, because then any rotation is possible. However, according to Proposition 5.1, it occurs only for twisted pairs. To summarize, we have proven the following.

Proposition 5.3. *Given known intrinsic parameters except for focal lengths, a motion is critical w.r.t. affine reconstruction if and only if the motion consists of (i) rotations with at most two distinct centres (twisted pair ambiguity), or (ii) motion on two conics*[1] *(one ellipse and one hyperbola) whose supporting planes are orthogonal and where the optical axis is tangent to the conic at each position, or (iii) translation along the optical axis, with arbitrary rotations around the optical axis.*

The motions are illustrated in Figure 4. In case (i) and (ii), the ambiguity of the reconstruction is twofold, as there is only one false absolute conic, whereas in case (iii) there is a one-parameter family of potential planes at infinity (all planes z=constant). Case (iii) can be seen as a special case of the critical motion in Proposition 5.2, which also has a single viewing direction, but arbitrary translations.

5.2.2 Two images

For two cameras, projective geometry is encapsulated in the 7 degrees of freedom in the fundamental matrix, and Euclidean geometry in the 5 degrees of freedom in the essential matrix. Hence, from two projective images we might hope to estimate Euclidean structure plus two additional calibration parameters. Hartley [10, 11] gave a method for the case where the only unknown calibration parameters are the focal lengths of the two cameras. This was later elaborated by Newsam et. al. [28], Zeller and Faugeras [41] and Bougnoux [4]. All of these methods are Kruppa-based. We will derive the critical motions for this case based on the results of the previous sections.

Proposition 5.4. *Given zero skew, unit aspect ratio, principal point at the origin, but unknown focal lengths for two cameras, then a motion (in addition to twisted pair) is critical if and only if (i) the optical axes of the two cameras intersect or (ii) the plane containing the optical axis of camera 1 and camera centre 2, is orthogonal to the plane containing optical axis of camera 2 and camera centre 1.*

[1] The actual critical motion is the conics minus the two points where the ellipse intersects the plane $z = 0$, since the image ω^* is non-proper at these points.

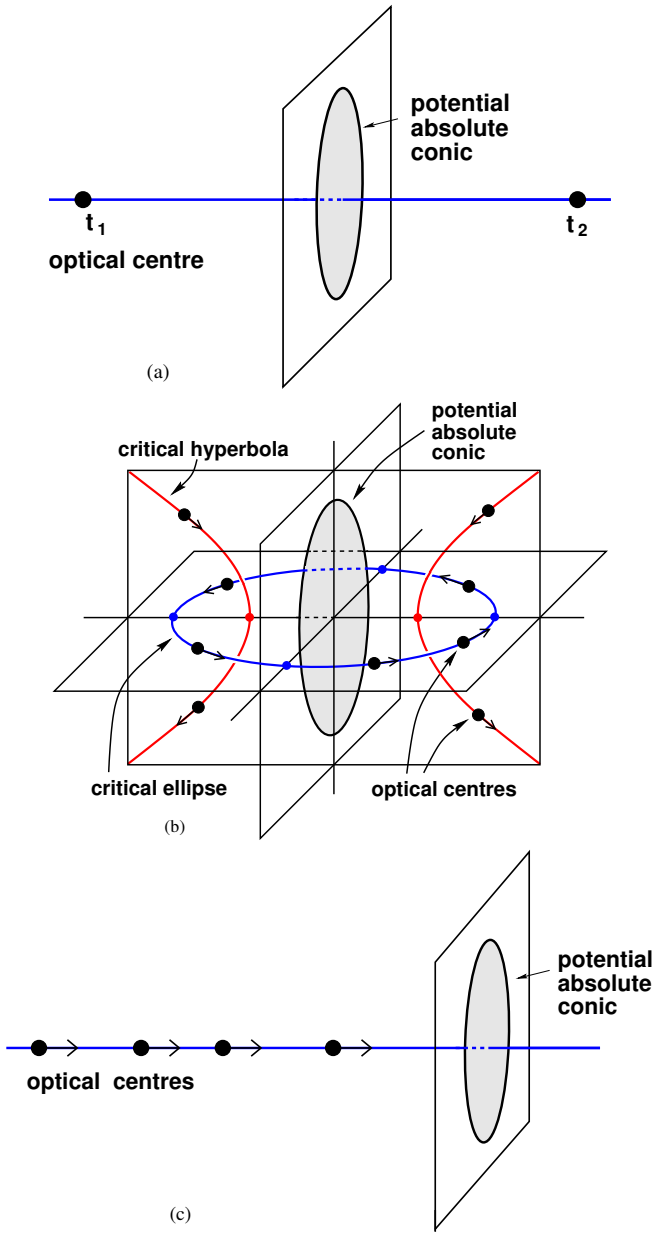


Figure 4: Critical motions for unknown focal lengths: (a) A motion with two fixed centres. (b) A planar motion on an ellipse and a hyperbola. (c) Translation along the optical axis. See Proposition 5.3.

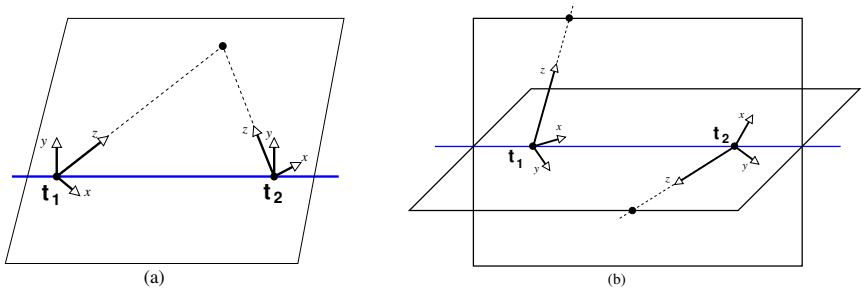


Figure 5: Critical configurations for two cameras: (a) Intersecting optical axes. (b) Orthogonal optical axis planes. See Proposition 5.4.

Proof. Cf. [28]. Suppose a motion is critical. Regarding Proposition 5.2, we see that if there is only one viewing direction, the optical axes are parallel and intersect at infinity, leading to (i) above. Examining the three possibilities in Proposition 5.3, we see that the first one is the twisted pair solution. The second one, either both cameras lie on the same conic (and hence their axes are coplanar and intersect) or one lies on the hyperbola, the other on the ellipse (in which case their optical axes lie in orthogonal planes) leading to (ii). Conversely, given any two cameras with intersecting or orthogonal-plane optical axes, it is possible to fit (a one-parameter family of) conics through the camera centres, tangential to the optical axes. ■

The two critical camera configurations are shown in Figure 5.

5.3 Known skew and aspect ratio

Consider the image $\omega^* = \boldsymbol{K}\boldsymbol{K}^T$ of Ω_∞^*. Inserting the parameterization of $\boldsymbol{K}$ in (2) into its dual ω, it turns out that $\omega_{12} = -\frac{s}{f^2\gamma}$. Since f and γ never vanish, requiring that the skew vanishes is equivalent to $\omega_{12} = 0$. The constraint can also be expressed in envelope form using $\omega^* \simeq \omega^{-1}$,

$$\omega_{12} = 0 \text{ or dually } \omega_{12}^*\omega_{33}^* - \omega_{13}^*\omega_{23}^* = 0. \tag{9}$$

If in addition to zero skew, unit aspect ratio is required in $\boldsymbol{K}$, it is equivalent to $\omega_{11} = \omega_{22}$. This follows from the fact that $\omega_{11} = \frac{1}{f^2}$ and $\omega_{22} = \frac{1}{f^2\gamma}$. The constraint can also be transfered to ω^*,

$$\omega_{11} - \omega_{22} = 0 \text{ or dually } \omega_{33}^*(\omega_{22}^* - \omega_{11}^*) + \omega_{13}^{*2} - \omega_{23}^{*2} = 0. \tag{10}$$

Analyzing the above constraints on ω in locus form, results in the following proposition when the plane at infinity Π_∞ is known.

Proposition 5.5. *Given Π_∞, a motion is critical with respect to zero skew and unit aspect ratio if and only if there are at most two viewing directions.*

Proof. For each image we have the two auto-calibration constraints (9), (10) with ω given by (7). Choose 3D coordinates in which the first camera has orientation $\boldsymbol{R}_1 = \boldsymbol{I}$. The image 1 constraints become simply $\boldsymbol{C}_{11} - \boldsymbol{C}_{22} = \boldsymbol{C}_{12} = 0$, so we can parameterize $\boldsymbol{C}_f$ with $\boldsymbol{C}_{11}$, $\boldsymbol{C}_{12}$ and $\boldsymbol{C}_{13}$. Given a subsequent image 2, represent its orientation $\boldsymbol{R}_2$ by a quaternion $\mathbf{q} = (q_0, q_1, q_2, q_3)$, evaluate its two auto-calibration constraints, and eliminate $\boldsymbol{C}_{11}$ between them to give:

$$(q_0^2+q_3^2)(q_1^2+q_2^2)\left((q_0q_1+q_2q_3)\boldsymbol{C}_{13} + (q_0q_2-q_1q_3)\boldsymbol{C}_{23}\right) = 0$$

One of the 3 factors must vanish. If the first vanishes the motion is an optical axis rotation, $q_1^2+q_2^2 = 0$. If the second vanishes it is a 180° flip about an axis orthogonal to the optical one,

Auto-calibration constraint	**Critical motions**	**Reconstruction ambiguity**
Known calibration	twisted pair duality	projective
Unknown focal length	(i) optical axis rotation	affine
but otherwise known	(ii) motion on two planar conics	projective
calibration	(iii) optical axis translation	projective
Unknown focal length	(i) intersecting optical axes	projective
(two images only)	(ii) orthogonal optical axis planes	projective
Zero skew and	(i) two viewing directions	affine
unit aspect ratio	(ii) complicated algebraic variety	projective

Table 1: Summary of critical motions in auto-calibration.

$q_0^2+q_3^2=0$. In both cases the viewing direction remains unchanged and no additional constraint is enforced on $\boldsymbol{C_f}$. Finally, if the third factor vanishes, solving for $\boldsymbol{C_f}$ in terms of $\mathbf{q}$ gives a linear family of solutions of the form

$$\boldsymbol{C_f} \simeq \alpha\,\boldsymbol{I} + \beta\,(\mathbf{o}_1\,\mathbf{o}_2^T + \mathbf{o}_2\,\mathbf{o}_1^T) \tag{11}$$

where $\mathbf{o}_1 = (0,0,1)^T$ and $\mathbf{o}_2 =$ (the third row of $\boldsymbol{R}_2(\mathbf{q})$) are the two viewing directions and (α,β) are arbitrary parameters. Conversely, given any potential AC $\boldsymbol{C_f} \not\simeq \boldsymbol{I}$, there is always exactly one pair of real viewing directions $\mathbf{o}_1, \mathbf{o}_2$ that make $\boldsymbol{C_f}$ critical under (11). The linear family $\alpha'\,\boldsymbol{I}+\beta'\,\boldsymbol{C_f}$ contains three rank 2 members, one for each eigenvalue λ of $\boldsymbol{C_f}$ (with $\beta'/\alpha' = -\lambda$). Explicit calculation shows that each rank 2 member can be decomposed uniquely (up to sign) into a pair of viewing direction vectors $\mathbf{o}_1, \mathbf{o}_2$ supporting (11), but only the pair corresponding to the middle eigenvalue is real. (Coincident eigenvalues correspond to coincident viewing directions and can be ignored). Hence, no potential AC $\boldsymbol{C_f}$ can be critical for three or more real directions simultaneously. ■

For potential absolute conics outside Π_∞ things are more complicated. For each image, there are two auto-calibration constraints. So in order to single out the true absolute conic (which has 8 degrees of freedom), at least 4 images are necessary. For a given Ω_f^* the polynomial constraints in (9) and (10) determine a variety in the space of rigid motions. We currently know of no easy geometrical interpretation of this manifold.

It is easy to see that given a critical camera motion, the ambiguity is not resolved by rotation around the camera's optical axis.

5.4 Summary

A summary of the critical motions for auto-calibration under the auto-calibration constraints studied is given in Table 1. The reconstruction ambiguity is classified as projective if the plane at infinity cannot be uniquely recovered, and affine if it is possible. As mentioned earlier, the twisted pair duality is not a true critical motion, since the positive-depth constraint can always resolve the ambiguity.

6 Particular Motions

Some critical motions occur frequently in practice. In this section, a selection of them is analyzed in more detail.

6.1 Pure rotation

In the case of a stationary camera performing arbitrary rotations, no 3D reconstruction is possible. There always exist many potential absolute conics outside Π_∞.

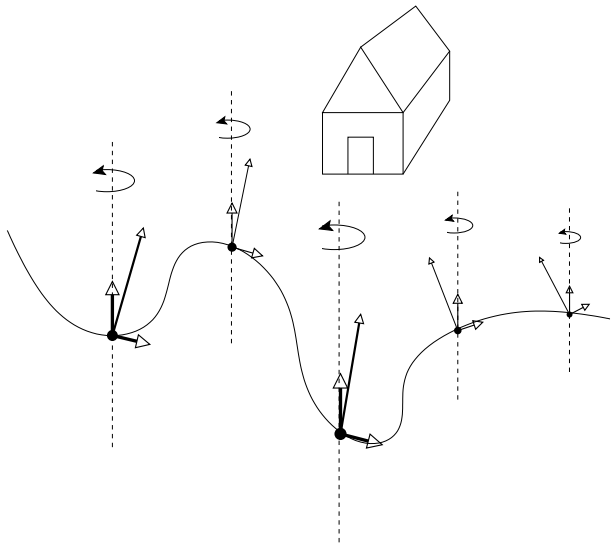


Figure 6: Rotations around the vertical axes with arbitrary translations.

However, it is still possible to recover the internal camera calibration, provided there are no potential absolute conics on Π_∞, cf. [37]. Proposition 5.2 and Proposition 5.5, regarding critical motions and potential ACs on Π_∞ tells us when such auto-calibration is possible for a purely rotating camera.

6.2 Pure translation

If a sequence of movements only consists of arbitrary translations and no rotations, all proper, virtual conics on Π_∞ are potential absolute conics. Still, one could hope to recover the plane at infinity correctly, and thus get an affine reconstruction.

Proposition 6.1. *Let $(\mathbf{t}_i)_{i=1}^m$ be a general sequence of translations, where m is sufficiently large. Then, the motion is*

(i) always critical w.r.t. affine reconstruction under the constraints zero skew and unit aspect ratio.

(ii) not critical w.r.t. affine reconstruction under the constraints zero skew, unit aspect ratio and vanishing principal point.

Proof. (i) We need to show that there exists a potential DAC Ω_f^* outside Π_∞, which is valid for all $(\mathbf{t}_i)_{i=1}^m$. Choose a coordinate system such that $\boldsymbol{P}_i = \begin{bmatrix}\boldsymbol{I} \mid -\mathbf{t}_i\end{bmatrix}$. Then for instance $\Omega_f^* = \mathrm{diag}(1,1,0,1)$ is a potential DAC (multiply $\boldsymbol{P}_i\Omega_f^*\boldsymbol{P}_i^T$ to get ω^* and check that it fulfills (9) and (10)). (ii) follows directly from Proposition 5.3. ∎

Note that translating only along the optical axis in case (ii) above results in a critical motion.

6.3 Parallel axis rotations

Sequences of rotations around parallel axes with arbitrary translations are interesting in several aspects. They occur frequently in practice and are one of the major degeneracies for auto-calibration with constant intrinsic parameters [36, 43]. See Figure 6.

It follows directly from Proposition 5.5 that given zero skew, unit aspect ratio and general rotation angles, the fixed-axis motion is not critical unless it is around the optical axis. If we

further add the vanishing principal point constraint, the optical axis remains critical according to Proposition 5.2. If we know only that the skew vanishes, we have the following proposition.

Proposition 6.2. *Let $(\boldsymbol{R}_i, \mathbf{t}_i)_{i=1}^m$ be a general motion whose rotations are all about parallel axes, where $\boldsymbol{R}_1 = \boldsymbol{I}$ and m is sufficiently large. Given Π_∞, the motion is critical w.r.t. zero skew if and only if the rotation is around one of the following axes:*

(i) $(0, *, *)$ *or* $(*, 0, *)$,

(ii) $(1, 1, 0)$ *or* $(1, -1, 0)$,

where each $*$ denotes an arbitrary real number.

Proof. Let $\boldsymbol{C}_f$ denote a false AC on Π_∞. The zero skew constraint in (9) using the parameterization in (7) gives $\boldsymbol{C}_{12} = 0$ for camera 1. An arbitrary rotation around a fixed axis (q_1, q_2, q_3) can be parameterized by $\lambda(q_1, q_2, q_3)$, $\lambda \in \mathbb{R}$. Inserting this into the zero skew constraint in (9) yields a polynomial in $\mathbb{R}[\lambda]$. Since λ can be arbitrary all coefficients of the polynomial must vanish. The solutions to the system of vanishing coefficients are the ones given above. ∎

Some of these critical axes may be resolved by requiring that the camera calibration should be constant. In [37], it is shown that parallel axis rotations under constant intrinsic parameters are always critical and give rise to the following pencil of potential absolute conics:

$$\boldsymbol{C}_f(\mu) = \boldsymbol{I} + \mu[q_1, q_2, q_3][q_1, q_2, q_3]^T. \tag{12}$$

Combining constant intrinsic parameters, and some a priori known values of the intrinsic parameters, some of the critical axes are still critical.

Corollary 6.1. *Let $(\boldsymbol{R}_i, \mathbf{t}_i)_{i=1}^m$ be a general motion with parallel axis rotations, where $\boldsymbol{R}_1 = \boldsymbol{I}$ and m is sufficiently large. Given Π_∞, and constant intrinsic parameters, the following axes are the only ones still critical:*

(i) $(0, *, *)$ *and* $(*, 0, *)$ *w.r.t. zero skew,*

(ii) $(0, 0, 1)$ *w.r.t. zero skew and unit aspect ratio,*

(iii) $(0, 0, 1)$ *w.r.t. an internally calibrated camera except for focal length.*

Proof. (i) Using the potential ACs in (12) in the proof of Proposition 6.2, one finds that the only critical axes remaining under the zero skew constraint are $(0, *, *)$ and $(*, 0, *)$. (ii) and (iii) are proved analogously. ∎

7 Experiments

In practice, a motion is never exactly degenerate due to measurement noise and modeling discrepancies. However, if the motion is close to a critical manifold it is likely that the reconstructed parameters will be inaccurately estimated. To illustrate the typical effects of critical motions, we have included some simple synthetic experiments for case of two cameras with unknown focal lengths but other intrinsic parameters known. We focus on the question of how far from critical the two cameras must be to give reasonable estimates of focal length and 3D Euclidean structure [20]. The experimental setup is as follows: two unit focal length perspective cameras view 25 points distributed uniformly within the unit sphere. The camera centres are placed at $(-2, -2, 0)$ and $(2, -2, 0)$ and their optical axes intersect at the origin, similar to the setup in Figure 5(a). Independent Gaussian noise of 1 pixel standard deviation is added to each image point in the 512×512 images.

In the experiment, the elevation angles are varied, upwards for the left camera and downwards for the right one, so that their optical axes are skewed and no longer meet. For each pose, the

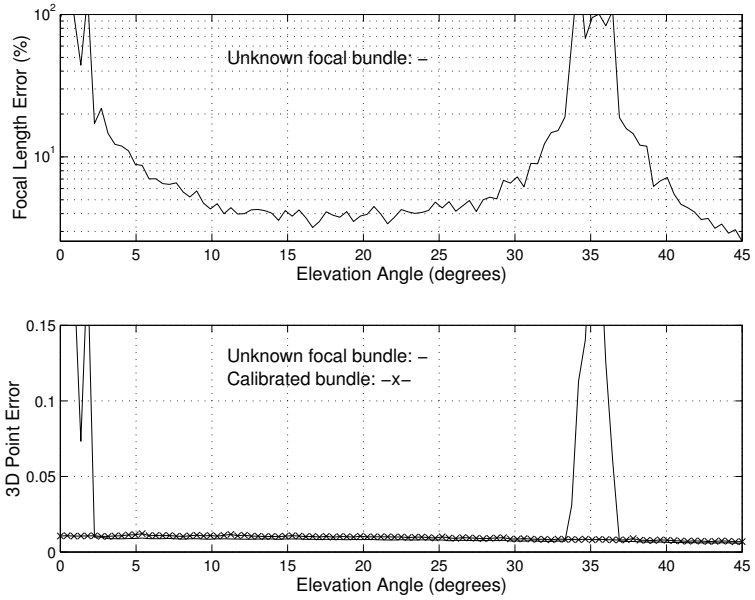


Figure 7: Relative errors vs. camera elevation for two cameras.

projective structure and the fundamental matrix are estimated by a projective bundle adjustment that minimises the image distance between the measured and reprojected points [3]. Then, the focal lengths are computed analytically with Bougnoux' method [4]. For comparison, a calibrated bundle adjustment with *known* focal lengths is also applied to the same data. The resulting 3D error is calculated by Euclidean alignment of the true and reconstructed point sets.

Figure 7 shows the resulting root mean square errors over 100 trials as a function of elevation angle. At zero elevation, the two optical axes intersect at the origin. This is a critical configuration according to Proposition 5.4. A second critical configuration occurs when the epipolar planes of the optical axes become orthogonal at around 35° elevation. Both of these criticalities are clearly visible in both graphs. For geometries more than about 5-10° from criticality, the focal lengths can be recovered quite accurately and the resulting Euclidean 3D structure is very similar to the optimal 3D structure obtained with *known* calibration.

8 Conclusion

In this paper, the critical motions in auto-calibration under several auto-calibration constraints have been derived. The various constraints on the intrinsic parameters have been expressed as subgroup conditions on the 3×3 upper triangular camera matrices. With this type of condition, we showed that the critical motions are independent of the specific values of the intrinsic parameters.

It is important to be aware of the critical motions when trying to auto-calibrate a camera. Additional scene or motion constraints may help to resolve the ambiguity, but clearly the best way to avoid degeneracies is to use motions that are "far" from critical. Some synthetic experiments have been performed that give some practical insight to the numerical conditioning of near-critical and critical stereo configurations.

Acknowledgments

This work was supported by the European Union under Esprit project LTR-21914 CUMULI. We would like to thank Sven Spanne for constructing the proof of Lemma A.1.

Appendix

Lemma A.1. *Let $\boldsymbol{A}$ be a real, symmetric 3×3 matrix of the form*

$$\boldsymbol{A} = \sigma_1 \mathbf{e}_1 \mathbf{e}_1^T + \sigma_2 \mathbf{e}_2 \mathbf{e}_2^T + \sigma_3 \mathbf{e}_3 \mathbf{e}_3^T + \rho \mathbf{t}\mathbf{t}^T,$$

where $\mathbf{e}_1 = (1,0,0)$, $\mathbf{e}_2 = (0,1,0)$, $\mathbf{e}_3 = (0,0,1)$, $\mathbf{t}$ a non-zero real 3 vector and ρ a non-zero real scalar. Let σ_1,σ_2 and σ_3 be given real scalars. Then, necessary and sufficient conditions on ($\mathbf{t}$,ρ) for $\boldsymbol{A}$ to have two equal eigenvalues can be divided into three cases:

(i) If $\sigma_1 \neq \sigma_2 \neq \sigma_3$, then $\mathbf{t}^T \mathbf{e}_i = 0$ for at least one i (where $i = 1, 2$ or 3). Furthermore, ρ can take the values:

$$\rho = \frac{(\sigma_i - \sigma_j)(\sigma_i - \sigma_k)}{t_j^2(\sigma_i - \sigma_k) + t_k^2(\sigma_i - \sigma_j)},$$

for any i for which $\mathbf{t}^T \mathbf{e}_i = 0$ (where $j \neq k \neq i$).

(ii) If $\sigma_i = \sigma_j \neq \sigma_k$, then

a. $\mathbf{t}^T \mathbf{e}_i = \mathbf{t}^T \mathbf{e}_j = 0$ and ρ arbitrary, or

b. $\mathbf{t}^T \mathbf{e}_k = 0$ and $\rho = \frac{\sigma_k - \sigma_i}{t_i^2 + t_j^2}$.

(iii) If $\sigma_1 = \sigma_2 = \sigma_3$, then $\mathbf{t}$ and ρ arbitrary.

Proof. It follows from the Spectral Theorem [5] that if $\boldsymbol{A}$ is real and symmetric with two equal eigenvalues μ, then there is a third eigenvector $\mathbf{v}$ and a scalar ν such that $\boldsymbol{A} = \mu \boldsymbol{I} + \nu \mathbf{v}\mathbf{v}^T$. (The eigenvalue corresponding to $\mathbf{v}$ is $\mu + \nu$). This gives

$$\sigma_1 \mathbf{e}_1 \mathbf{e}_1^T + \sigma_2 \mathbf{e}_2 \mathbf{e}_2^T + \sigma_3 \mathbf{e}_3 \mathbf{e}_3^T + \rho \mathbf{t}\mathbf{t}^T - \nu \mathbf{v}\mathbf{v}^T - \mu \boldsymbol{I} = 0.$$

Multiplying this matrix equation with $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$, results in three vector equations,

$$\begin{aligned} (\sigma_1 - \mu)\mathbf{e}_1 + \rho t_1 \mathbf{t} - \nu v_1 \mathbf{v} &= 0 \\ (\sigma_2 - \mu)\mathbf{e}_2 + \rho t_2 \mathbf{t} - \nu v_2 \mathbf{v} &= 0 \\ (\sigma_3 - \mu)\mathbf{e}_3 + \rho t_3 \mathbf{t} - \nu v_3 \mathbf{v} &= 0. \end{aligned} \tag{13}$$

To prove (i), assume $\sigma_1 \neq \sigma_2 \neq \sigma_3$. The orthogonal bases $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$ are linearly independent and cannot all be linear combinations of $\mathbf{t}$ and $\mathbf{v}$, so one of $\sigma_i - \mu$ must vanish, and thereby exactly one. Suppose $\mu = \sigma_1$. Then,

$$\rho t_1 \mathbf{t} - \nu v_1 \mathbf{v} = 0.$$

If one of the coefficients is non-zero, then $\mathbf{t}$ and $\mathbf{v}$ would be linearly dependent. However, this is impossible because $\mathbf{e}_2$ and $\mathbf{e}_3$ are linearly independent and $\sigma_2 - \mu \neq 0$, $\sigma_3 - \mu \neq 0$. Analogously, $\rho \neq 0$ because otherwise $\mathbf{e}_2$ and $\mathbf{e}_3$ would be linearly dependent according to (13). Therefore $\mathbf{t}^T \mathbf{e}_1 = 0$.

If $\mathbf{t}$ is orthogonal to $\mathbf{e}_1$ and $\mu = \sigma_1$, some easy calculations yield that ρ must be chosen as

$$\rho = \frac{(\sigma_1 - \sigma_2)(\sigma_1 - \sigma_3)}{t_2^2(\sigma_1 - \sigma_3) + t_3^2(\sigma_1 - \sigma_2)},$$

which is also sufficient.

When two or three of σ_i ($i = 1, 2, 3$) are equal, similar arguments can be used to deduce (ii) and (iii).

■

References

[1] Armstrong, M., A. Zisserman, and R. Hartley: 1996, 'Self-Calibration from Image Triplets'. In: *European Conf. Computer Vision*. Cambridge, UK, pp. 3–16.

[2] Åström, K. and A. Heyden: 1996, 'Euclidean Reconstruction from Constant Intrinsic Parameters'. In: *Int. Conf. Pattern Recognition*. Vienna, Austria, pp. 339–343.

[3] Atkinson, K. B.: 1996, *Close Range Photogrammetry and Machine Vision*. Whittles Publishing.

[4] Bougnoux, S.: 1998, 'From Projective to Euclidean Space Under any Practical Situation, a Criticism of Self-Calibration'. In: *Int. Conf. Computer Vision*. Mumbai, India, pp. 790–796.

[5] Cullen, C. G.: 1966, *Matrices and Linear Transformations*. Addison-Wesley Publishing Company.

[6] Faugeras, O.: 1993, *Three-Dimensional Computer Vision*. MIT Press, Cambridge, Mass.

[7] Faugeras, O., Q.-T. Luong, and S. Maybank: 1992, 'Camera self-calibration: Theory and experiments'. In: *European Conf. Computer Vision*. Santa Margherita Liguere, Italy, pp. 321–334.

[8] Faugeras, O. D.: 1992, 'What can be seen in three dimensions with an uncalibrated stereo rig?'. In: *European Conf. Computer Vision*. Santa Margherita Liguere, Italy, pp. 563–578.

[9] Golub, G. and C. Van Loan: 1989, *Matrix Computation*. The Johns Hopkins University Press, Baltimore.

[10] Hartley, R.: 1992, 'Estimation of Relative Camera Positions for Uncalibrated Cameras'. In: *European Conf. Computer Vision*. Santa Margherita Liguere, Italy, pp. 579–587.

[11] Hartley, R.: 1993, 'Extraction of Focal Lengths from the Fundamental Matrix'. Unpublished manuscript.

[12] Hartley, R., R. Gupta, and T. Chang: 1992, 'Stereo from uncalibrated cameras'. In: *Conf. Computer Vision and Pattern Recognition*. Champaign, USA, pp. 761–764.

[13] Helmholtz, H.: 1867, *Handbuch der Physiologischen Optik*. Leibzig: Voss.

[14] Heyden, A. and K. Åström: 1997, 'Euclidean Reconstruction from Image Sequences with Varying and Unknown Focal Length and Principal Point'. In: *Conf. Computer Vision and Pattern Recognition*. San Juan, Puerto Rico, pp. 438–443.

[15] Heyden, A. and K. Åström: 1998, 'Minimal Conditions on Intrinsic Parameters for Euclidean Reconstruction'. In: *Asian Conf. Computer Vision*, Vol. II. Hong Kong, pp. 169–176.

[16] Heyden, A. and K. Åström: 1999, 'Flexible Calibration: Minimal Cases for Auto-calibration'. In: *Int. Conf. Computer Vision*. Kerkyra, Greece, pp. 350–355.

[17] Heyden, A. and G. Sparr: 1999, 'Reconstruction from Calibrated Cameras - A New Proof of the Kruppa Demazure Theorem'. *Journal of Mathematical Imaging and Vision* **10**(2), 123–142.

[18] Horn, B. K. P.: 1987, 'Motion Fields Are Hardly Ever Ambiguous'. *Int. Journal Computer Vision* **1**(3), 239–258.

[19] Kahl, F.: 1999, 'Critical Motions and Ambiguous Euclidean Reconstructions in Auto-Calibration'. In: *Int. Conf. Computer Vision*. Kerkyra, Greece, pp. 469–475.

[20] Kahl, F. and B. Triggs: 1999, 'Critical Motions in Euclidean Structure from Motion'. In: *Conf. Computer Vision and Pattern Recognition*. Fort Collins, USA, pp. 366–372.

[21] Krames, J.: 1941, 'Zur Ermittlung eines Objectes aus zwei Perspectiven (Ein Beitrag zur Theorie der gefährlichen Örter).'. *Monatch. Math. Phys.* **49**, 327–354.

[22] Kruppa, E.: 1913, 'Zur Ermittlung eines Objektes aus zwei Perspektiven mit innerer Orientierung'. *Sitz-Ber. Akad. Wiss., Wien, math. naturw. Kl. Abt* **IIa**(122), 1939–1948.

[23] Longuet-Higgins, H. C.: 1988, 'Multiple Interpretations of A Pair of Images of A Surface'. *Phil. Trans. Royal Soc. London A* **418**, 1–15.

[24] Maybank, S.: 1993, *Theory of Reconstruction from Image Motion*. Springer-Verlag, Berlin, Heidelberg, New York.

[25] Maybank, S. and A. Shashua: 1998, 'Ambiguity in Reconstruction from Images of Six Points'. In: *Int. Conf. Computer Vision*. Mumbai, India, pp. 703–708.

[26] Maybank, S. J. and O. D. Faugeras: 1992, 'A theory of self calibration of a moving camera'. *Int. Journal Computer Vision* **8**(2), 123–151.

[27] Negahdaripour, S.: 1990, 'Multiple Interpretations of the Shape and Motion of Objects from Two Perspective Images'. *IEEE Trans. Pattern Analysis and Machine Intelligence* **12**(11), 1025–1039.

[28] Newsam, G., D. Huynh, M. Brooks, and H.-P. Pan: 1996, 'Recovering Unknown Focal Lengths in Self-Calibration: An Essentially Linear Algorithm and Degenerate Configurations'. In: *Int. Arch. Photogrammetry & Remote Sensing*, Vol. XXXI-B3. Vienna, Austria, pp. 575–80.

[29] Pollefeys, M., R. Koch, and L. Van Gool: 1998, 'Self-Calibration and Metric Reconstruction in spite of Varying and Unknown Internal Camera Parameters'. In: *Int. Conf. Computer Vision*. Mumbai, India, pp. 90–95.

[30] Pollefeys, M., L. Van Gool, and M. Oosterlinck: 1996a, 'The Modulus Constraint: A New Constraint for Self-Calibration'. In: *Int. Conf. Pattern Recognition*. Vienna, Austria, pp. 349–353.

[31] Pollefeys, M., L. Van Gool, and M. Proesmans: 1996b, 'Euclidean 3D Reconstruction from Image Sequences with Variable Focal Lengths'. In: *European Conf. Computer Vision*. Cambridge, UK, pp. 31–42.

[32] Schröter, H.: 1880, *Theorie der Oberflächen zweiter Ordnung und der Raumkurve dritter Ordnung als Erzeugnisse projektvischer Gebilde*. Leibzig: Teubner.

[33] Semple, J. G. and G. T. Kneebone: 1952, *Algebraic Projective Geometry*. Clarendon Press, Oxford.

[34] Spain, B.: 1960, *Analytical Quadrics*. Pergamon Press.

[35] Sparr, G.: 1991, 'An algebraic-analytic method for affine shapes of point configurations'. In: *Scandinavian Conf. on Image Analysis*. Aalborg, Denmark, pp. 274–281.

[36] Sturm, P.: 1997a, 'Critical motion sequences for monocular self-calibration and uncalibrated Euclidean reconstruction'. In: *Conf. Computer Vision and Pattern Recognition*. San Juan, Puerto Rico, pp. 1100–1105.

[37] Sturm, P.: 1997b, 'Vision 3D Non Calibrée: Contributions à la Reconstruction Projective et Étude des Mouvements Critiques pour l'Auto-Calibrage'. Ph.D. thesis, Institut National Polytechnique de Grenoble.

[38] Sturm, P.: 1999, 'Critical Motion Sequences for the Self-Calibration of Cameras and Stereo Systems with Variable Focal Length'. In: *British Machine Vision Conf.* Nottingham, UK, pp. 63–72.

[39] Triggs, B.: 1997, 'Autocalibration and the Absolute Quadric'. In: *Conf. Computer Vision and Pattern Recognition*. San Juan, Puerto Rico, pp. 609–614.

[40] Van Gool, L., T. Moons, M. Proesmans, and M. Van Diest: 1994, 'Affine reconstruction from perspective image pairs obtained by a translating camera'. In: *Int. Conf. Pattern Recognition*. Jerusalem, Israel, pp. 290–294.

[41] Zeller, C. and O. Faugeras: 1996, 'Camera Self-Calibration from Video Sequences: the Kruppa Equations Revisited'. Technical Report 2793, INRIA, Sophia-Antipolis, France.

[42] Zisserman, A., P. Beardsley, and I. Reid: 1995, 'Metric Calibration of a Stereo Rig'. In: *IEEE Workshop on Representation of Visual Scenes*. Cambridge Ma, USA, pp. 93–110.

[43] Zisserman, A., D. Liebowitz, and M. Armstrong: 1998, 'Resolving Ambiguities in Auto-Calibration'. *Phil. Trans. Royal Soc. London A* **356**(1740), 1193–1211.

Le Calcul de Pose : de nouvelles méthodes matricielles

Camera Pose Revisited : new linear algorithms

Marc-André Ameller Long Quan Bill Triggs

INRIA Rhône-Alpes, 655, av. de l'Europe
38334 St. Ismier cedex, France.
prénom.nom@inria.fr

Résumé

Estimer la pose d'une caméra signifie calculer sa position et son oritentation, à l'aide de ses paramètres internes, ainsi que de la connaissance de la position de points 3D de référence et de leurs projections dans l'image. On va brièvement passer en revue différentes méthodes existantes pour le calcul de la pose, puis on introduira 4 nouveaux algorithmes. Tous ces algorithmes sont basés sur l'algèbre linéaire. Le premier est basé sur le calcul de valeur propre d'une matrice 5×5. Elle permet de résoudre le problème de pose avec 3 points et donc, donne plusieurs solutions. Les 3 autres algorithmes, qui permettent le calcul de la pose à partir de quatre points, donne une unique solution, qui est le noyau d'une matrice. Ce noyau est calculé à l'aide d'une SVD. Ces techniques sont basées sur les matrices de résultants : la méthode 24×24 est une méthode de résultant de Macaulay, et les méthodes 12×12 et 9×9 sont des versions compressées, obtenues après élimination gaussienne. Un des avantages de ces méthodes est leur simplicité. En particulier, les coefficients des matrices ne sont que des fonctions simples des données. Les expériences numériques donnent une comparaison entre les nouveaux algorithmes et ceux déja existant.

Mots clés : Calibration, Estimation de la pose, Résolution de systèmes de polynômes, Matrices de résultant, Résection.

Abstract

Camera pose estimation is the problem of determining the position and orientation of an internally calibrated camera from known 3D reference points and their images. We briefly survey existing methods for pose estimation, then introduce four new algorithms based on efficient matrix computations. The first is based on eigendecomposition of a 5×5 matrix and returns the four intrinsic solutions to the problem of pose from 3 points. The remaining three methods give a unique linear solution from four points by SVD null space computation on resultant matrices. The 24×24 method is the raw resultant matrix, and the 12×12 and 9×9 methods are compressed versions of this obtained by Gaussian elimination with pivoting on constant entries. All of these methods are simple to implement. In particular, the matrix entries are simple functions of the input data. Numerical experiments are given comparing the performance of the new algorithms with several existing methods.

Keywords : Calibration, Pose Estimation, Resection, Polynomial Solving, Resultant Matrices.

1 Introduction

L'estimation de la pose d'une caméra consiste à déterminer la position et l'orientation de la caméra à partir de la calibration de celle-ci, ainsi que de la donnée de coordonnées de points de référence et de la position respective de leurs projections sur l'image. Ce problème est aussi appelé *resection* par les photogramètres, et a été bien étudié dans le passé. Avec 3 points, le problème de pose possède dans le cas général 4 solutions. De nombreuses méthodes sont connues pour calculer ces solutions. La plus ancienne est probablement due à Lagrange en 1795. Voir [20, 7, 8, 9, 17]. Fischler & Bolles [7] ont donné une méthode devenue populaire en vision par ordinateur, lorsqu'ils ont introduit la méthode RANSAC pour la détection des valeurs abérrantes dans les données initiales. Haralick *et al* [9] a montré différentes variations, nouvelles et anciennes, de la méthode basique utilisant 3 points, et étudié les différentes stabilités suivants la manière de procéder aux substitutions et éliminations. Pour les problèmes redondants, des méthodes itératives ont été développées [13, 21, 3]. Des méthodes ont aussi été développées pour le calcul de pose à partir de correspondances de lignes et d'autres primitives. [10, 4, 1, 14, 12].
La méthode de 3 points possède des solutions multiples. Si l'on veut obtenir une solution unique, il faut ajouter des

Ce papier fut publié dans les actes de *Reconnaissance des Formas et Intelligence Artificielle* 2002.

données. Par exemple, si l'on ajoute un quatrième point de référence, en général, il existe une solution unique. Aussi, dans certains cas dégénérés, même un nombre infini de points ne donne pas de solution unique. Ces *configurations critiques* sont connues précisément. Voir [18, 20] pour les détails. Brièvement, ces configurations correspondent au cas où la caméra est sur une cubique twisté (l'horoptère) dans l'espace, qui est tracé sur un cylindre (Le cylindre dangereux). On peut noter les cas suivants qui sont dégénérés : (1) tous les points sont à l'infini (la translation de la caméra ne peut être estimée); (2) une droite et un cercle, avec la droite orthogonale au plan du cercle. Ce dernier cas est particulièrement ennuyeux, pour le calcul de pose à partir de 3 points, ou à partir de 4 points coplanaires et formant un rectangle, lorsque la caméra est dans le proche voisinage des points. On montrera les effets de ces cas dégénérés dans quelques expériences.

L'article est motivé par le fait qu'il n'y ait que peu de méthodes donnant directement la solution unique du problème de pose dans le cas de données redondantes. Des familles d'algorithmes linéaires sont présentées dans [16, 19]. Malheureusement, ces méthodes possèdent les mêmes inconvénients que les méthodes algébriques en ce sens que les coefficients des matrices utilisées sont compliqués, extraits de polynômes de degré 4, ce qui alourdit considérablement l'implémentation. Dans cet article, on propose : (1) un nouvel algorithme pour le calcul de pose à partir de 3 points, basé sur le calcul des valeurs propres d'une matrice 5×5 obtenue par la méthode de résultant de Bezout-Cayley-Dixon; (2) Trois algorithmes linéaires pour le calcul de pose à partir de 4 points, basé sur le calcul du noyau de matrices 9×9, 12×12 et 24×24, dont les coefficients sont simples.

L'article est organisé comme suis. Dans la section 2, on reprend les bases géométriques du calcul de pose. La section 3 passe en revue une méthode pour résoudre linéairement des systèmes de polynômes. Dans la section 4, on présente une méthode de valeurs propres pour la méthode de 3 points. Les 3 nouveaux algorithmes linéaires sont présentés dans la section 5. La section 6 donne quelques premiers résultats d'expérimentation qui comparent ancienne et nouvelles méthodes, et ceci sur des données simulées. Enfin, la section 7 résume les contributions et donne quelques conclusions.

2 Géométrie pour le calcul de pose à partir de points

Étant donnée une caméra calibrée centrée en $\mathbf{c}$ et des correspondances entre des points de référence 3D $\mathbf{p}_i$ et leurs images $\mathbf{u}_i$. Chaque paire de correspondances de points donne une contrainte sur les distances inconnues entre les points de référence et le centre de la caméra, $x_i = ||\mathbf{p}_i - \mathbf{c}||$. Les contraintes s'écrivent (cf. Figure 1) :

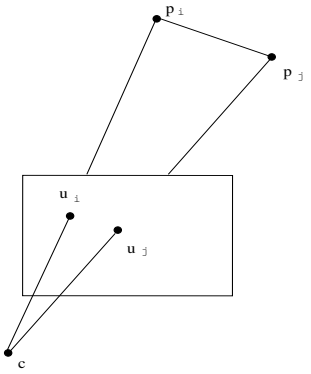


FIG. 1 – *La détermination de la géométrie de base pour le calcul de pose à partir de paires de correspondances de points* $\mathbf{p}_i \leftrightarrow \mathbf{u}_i$ *et* $\mathbf{p}_j \leftrightarrow \mathbf{u}_j$ *entre des points 3D et leurs projections dans les images.*

$$P_{ij}(x_i,x_j) \ :\equiv: {x_i}^2 + {x_j}^2 + c_{ij}\, x_i\, x_j - {d_{ij}}^2 \ :=: 0 \tag{1}$$

$$c_{ij} \ :\equiv: -2\, \cos\theta_{ij} \tag{2}$$

où $d_{ij} = ||\mathbf{p}_i - \mathbf{p}_j||$ est la distance connue entre les points de référence numéro i et j, et θ_{ij} l'angle entre les demi-droites $[\mathbf{cp}_i)$ et $[\mathbf{cp}_j)$. Le cosinus de cet angle peut aisément être déduit de la position des projections des points dans l'image et de la matrice $\mathbf{K}$ de calibration de la caméra de la manière suivante :

$$\cos\theta_{ij} \ :=: \frac{\mathbf{u}_i^T\, \mathbf{C}\, \mathbf{u}_j}{\sqrt{(\mathbf{u}_i^T\, \mathbf{C}\, \mathbf{u}_i)(\mathbf{u}_j^T\, \mathbf{C}\, \mathbf{u}_j)}},$$

où $\mathbf{C} = (\mathbf{K}\, \mathbf{K}^T)^{-1}$.

Pour $n = 3$, on obtient le système de polynômes suivant.

$$\begin{cases} P_{12}(x_1,x_2) = 0, \\ P_{13}(x_1,x_3) = 0, \\ P_{23}(x_2,x_3) = 0 \end{cases}$$

où les trois inconnues sont les distances x_1,x_2,x_3. Le théorème de Bezout [2] borne le nombre de solutions du système à $8 = 2^3$ solutions. Cependant, n'ayant que des termes en des puissances paires des x_i, le système est invariant par la transformation $x_i \mapsto -x_i$. Ainsi, les solutions peuvent être classées par paires de solutions opposées. En utilisant le résultant de Sylvester 2 fois, on peut éliminer les variables x_2 et x_3, de manière à obtenir un polynôme de degré 8 en x_1, dont tout les termes

ont un degré pair, c'est-à-dire un polynôme de degré 4 en $x = x_1^2$:.

$$g(x) = a_4\,x^4 + a_3\,x^3 + a_2\,x^2 + a_1\,x + a_0 = 0.$$

Cette équation possède au plus 4 solutions en x et peut-être résolue de manière explicite. Comme les x_i sont positifs, $x_1 = \sqrt{x}$.

Dans la méthode, et pour les autres qui vont suivre, on calcule les distances x_i afin d'estimer les coordonnées des points de référence 3D dans un repère attaché et centré sur la caméra : $\tilde{\mathbf{p}}_i = x_i\,\mathbf{K}^{-1}\mathbf{u}_i$. Pour trouver la pose de la caméra, il suffit alors d'estimer le mouvement rigide qui aligne le mieux les points dans leurs repères respectifs. Les centres de gravité des deux nuages de points donnent la translation. La rotation s'en déduit facilement par la méthode des quaternions ou par SVD [11, 6].

3 Les méthodes issues de l'algèbre linéaire

Les matrices de résultant L'algèbre linéaire est un outil puissant pour manipuler les polynômes [15, 5, 2]. Un polynôme $P_i(x) \;:=: \sum_\alpha c_{\alpha,i}\,x^\alpha$ en les variables $x \;:=: (x_1,\ldots,x_n)$ est une somme finie de *coefficients* $c_{\alpha,i}$ multipliés par des *monômes* $x^\alpha \;:=: x_1^{\alpha_1}\cdot x_2^{\alpha_2}\cdot\ldots\cdot x_n^{\alpha_n}$, avec $\alpha \;:=: (\alpha_1,\ldots,\alpha_n) \in \mathbb{Z}^n$ qui est appelé multi-index ou vecteur exposant. L'idée clé est de regarder les polynômes comme produits de vecteurs lignes de coefficients par des vecteurs colonnes de monômes. De manière similaire, on peut voir un système de polynôme comme une matrice de coefficients que multiplie un vecteur colonne de monômes. La structure interne des polynômes n'apparaît ainsi que de manière implicite. C'est une manière de "sur-paramètrer" le problème, mais qui a l'avantage d'être linéaire.

Considérons un système de polynômes quelconque :

$$\begin{cases} P_1(x_1,\ldots,x_n) &=& 0 \\ & \vdots & \\ P_q(x_1,\ldots,x_n) &=& 0, \end{cases}$$

On va résoudre ce système par linéarisation. Pour cela, on va choisir des polynômes $(Q_j)_{j\in\{1,\ldots,n\}}$ de l'idéal engendré par les P_i, de sorte que n soit plus grand que le nombre de monômes distincts présents dans l'écriture des Q_j. Avec les polynômes Q_j, on peut appliquer le procédé expliqué ci-dessus, c'est-à-dire écrire le système de polynômes $\{Q_j = 0, j \in \{1,\ldots,n\}$ sous la forme d'un produit d'une matrice de coefficients par un vecteur colonne de monôme :

$$\mathbf{M}\mathbf{v} = 0.$$

Comme les polynômes Q_j sont choisit en plus grand nombre que les monômes, la matrice $\mathbf{M}$ possède plus de ligne que de colonnes. La matrice $\mathbf{M}$ s'appelle matrice résultante du système de polynômes. Dans les cas favorables, le système possède une unique solution, et le noyau de la matrice résultante est de dimension 1. Dans ce cas, l'unique solution du système de polynômes est facile à extraire. Les polynômes permettant la construction de la matrice résultante, peuvent être soit trouvés à la main, soit en utilisant une méthode automatique [15]. Bien que ces constructions automatiques donnent des solutions convenables, en général, elles ne garantissent pas la minimalité de la taille de la matrice obtenue. Dans notre cas, la matrice a été construite à la main.

La Méthode de Bezout-Cayley-Dixon Une autre technique pour résoudre les systèmes de polynômes est la méthode de Bezout-Cayley-Dixon [15]. Considérons un système de polynômes général :

$$\begin{cases} P_1(x_1,\ldots,x_n) &=& 0 \\ & \vdots & \\ P_{n+1}(x_1,\ldots,x_n) &=& 0. \end{cases}$$

Introduisons les nouvelles variables $y_1,\ldots,y_n$ et construisons la matrice :

$$\mathbf{M} = \begin{pmatrix} P_1(Z_0) & P_1(Z_1) & \ldots & P_1(Z_n) \\ \vdots & \vdots & & \vdots \\ P_{n+1}(Z_0) & P_{n+1}(Z_1) & \ldots & P_{n+1}(Z_n) \end{pmatrix}.$$

Où, $Z_i = (y_1,\ldots,y_i,x_{i+1},\ldots,x_n)$, avec $i \in \{0,\ldots,n\}$. Dans chaque colonne, on convertit un x de plus en un y (Le résultat dépend en général de l'ordre des variables qu'on choisit). Si $(x_1,\ldots,x_n)$ est une solution du système, la première colonne s'annule et ainsi $\det(\mathbf{M}) = 0$. En plus, le déterminant est divisible par $(x_1-y_1)\ldots(x_n-y_n)$ parce que $P_i(\ldots,x_i,\ldots) - P_i(\ldots,y_i,\ldots)$ est divisible par $(x_i - y_i)$ (s'annule si $y_i = x_i$). Si nous posons :

$$P(x,y) = \frac{\det \mathbf{M}}{\prod_{i=1}^n (x_i - y_i)}$$

$P(x,y)$ est bien un polynôme, et nous avons :

$$P(x,y) \;:=: \sum c_{\alpha,\beta} x^\alpha y^\beta \;:=: \mathbf{w}^T\mathbf{C}\,\mathbf{v}$$

où $\mathbf{v} = (\ldots x^\alpha \ldots)^T$ et $\mathbf{w} = (\ldots y^\beta \ldots)^T$ sont des vecteurs de monômes et $\mathbf{C}$ est une matrice de coefficients $c_{\alpha,\beta}$. Résoudre le système polynomial se réduit à la résolution du système linéaire

$$\mathbf{C}\cdot\mathbf{v} = 0.$$

En effet, comme les y peuvent prendre des valeurs arbitraires, chaque monôme y^α est linéairement indépendant de tous les autres.

4 L'algorithme de vecteur propre pour la pose de 3 points

Nous appliquons maintenant la méthode de Bézout-Cayley-Dixon au calcul de pose à partir de 3 points. Pour cela, on divise chacun des P_{ij} par x_1^2. On pose ensuite :

$$u_1 = 1/x1,\, u_2 = x_2/x_1,\, u_3 = x_3/x_1.$$

On obtient ainsi 3 polynômes Q_{12}, Q_{13}, Q_{23} en les variables u_1, u_2, et u_3. On considère maintenant ces polynômes comme étant des polynômes en u_2,u_3 à coefficients en u_1 et c_{ij},d_{ij}, et on applique la méthode ci-dessus. Ceci donne :

$$\mathbf{w}^T \mathbf{M} \mathbf{v} = 0, \tag{3}$$

avec :

$$\mathbf{v} = \begin{pmatrix} 1 \\ y_2 \\ y_3 \\ y_2\, y_3 \\ y_2^2 \end{pmatrix} \quad \mathbf{w} = \begin{pmatrix} 1 \\ u_2 \\ u_3 \\ u_2^2 \\ u_3\, u_3 \end{pmatrix}$$

Et, $\mathbf{M} = \mathbf{M}_1 + \mathbf{M}_2 u_1^2$, où :

$$\mathbf{M}_2 = \begin{pmatrix} -d_{23}\, c_{12}\, c_{13} & -d_{13}\, c_{12}\, c_{23} + c_{13}\, (d_{12} - d_{23}) & c_{12}\, (-d_{23} + d_{13}) & -d_{23} + d_{13} + d_{12} & -d_{13}\, c_{23} \\ c_{13}\, (d_{12} - d_{23}) & -d_{13}\, c_{23} & -d_{23} + d_{13} + d_{12} & 0 & 0 \\ -d_{12}\, c_{13}\, c_{23} + c_{12}\, (d_{13} - d_{23}) & -d_{23} + d_{13} + d_{12} & -d_{12}\, c_{23} & 0 & 0 \\ -d_{12}\, c_{23} & 0 & 0 & 0 & 0 \\ -d_{23} + d_{13} + d_{12} & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{M}_1 = \begin{pmatrix} 0 & c_{12}\, c_{23} - c_{13} & -c_{12} & -2 & c_{23} \\ -c_{13} & c_{23} - c_{12}\, c_{13} & -2 & -c_{12} & 0 \\ -c_{12} + c_{13}\, c_{23} & -2 + c_{13}\, c_{12}\, c_{23} & c_{23} - c_{12}\, c_{13} & -c_{13} & c_{13}\, c_{23} \\ c_{23} & c_{12}\, c_{23} & 0 & 0 & c_{23} \\ -2 & -c_{12} & -c_{13} & -c_{23} & 0 \end{pmatrix}$$

Résoudre l'équation (3) signifie trouver u_1 tel que :

$$(u_1^{-2}\mathbf{I} + \mathbf{M}_1^{-1}\mathbf{M}_2)\mathbf{v} = 0,$$

puisque $\mathbf{M}_1$ est génériquement non-singulière. Mais ceci est équivalent à trouver les valeurs propres de la matrice $\mathbf{M}_1^{-1}\mathbf{M}_2$. Le calcul de $x_1 = 1/u_1$ se ramène donc a un calcul de valeurs propres.

5 Algorithme linéaire pour la pose de 4 points

Maintenant, nous appliquons les méthodes de résultant au problème de pose. Pour 4 points, le système d'entrée possède 6 équations (1) à 4 variables : $\{P_{ij}(x_i,x_j) := : 0 \mid 1 \le i < j \le 4\}$. Chaque solution de ce système satisfait donc :

$$\{\, x_i\, P_{jk}(x_j,x_k) = 0 \,|\, i,j < k = 1,2,3,4\}$$

Ce nouveau système contient 24 polynômes en 24 monômes : 4 de la forme x_i^3, 12 de la forme $x_i^2\, x_j\, x_k$ et 4 de la forme $x_i\, x_j\, x_k$ (où i,j,k sont distincts), et finallement 4 x_i. Il s'avère que (si les entrées c_{ij} et d_{ij} sont correctes) la matrice résultante correspondante 24×24 a génériquemenet un noyau de dimension 1, qui contient les deux solutions algébriques possibles x et $-x$, mais aussi la solution trivialle $x = 0$ qui provient de la multiplication de P_{jk} par x_i. Mais il est facile de choisir la solution désirée, puisque les profondeurs x_i doivent être positives. En détail, la matrice 24×24 et la liste des monômes étiquettant ses colonnes sont sur la figure 2.

Les coefficients de la matrice sont de très simples fonctions des données d'entrée. Une fois que le vecteur de monômes $\mathbf{v}$ du noyau est trouvé par SVD de la matrice, la solution pour les profondeurs x peut être calculée comme suit. Nous pouvons simplement prendre quelques ratios fixé des éléments, comme

$$(x_1,x_2,x_3,x_4) \;:=:\; \left(\sqrt{\frac{\mathbf{v}_1}{\mathbf{v}_{21}}}, \sqrt{\frac{\mathbf{v}_2}{\mathbf{v}_{22}}}, \sqrt{\frac{\mathbf{v}_3}{\mathbf{v}_{23}}}, \sqrt{\frac{\mathbf{v}_4}{\mathbf{v}_{24}}}\right).$$

Une méthode quelque peu plus appropriée est calculer le facteur d'échelle à l'aide des composantes les plus grandes du vecteur solution. En effet, en procédant ainsi, on utilise les composantes qui présentent le moins de risque d'être affectées par le bruit, ce qui donnera le meilleur facteur d'échelle possible. On procède ensuite de même pour le calcul des x_i à partir du facteur d'échelle.

Génériquement, si les coefficients $c_{i,j}$ et $d_{i,j}$ ne correspondent pas à une géométrie 3D cohérente, la matrice va être de rang maximal. Donc le déterminant de la matrice 24×24 est un multiple (non-trivial) du polynôme résultant du système.

5.1 Les méthodes linéaires 12×12 et 9×9

Une matrice résultante peut être réduite en une matrice plus petite par élimination Gaussienne symbolique partielle. Ici, le fait que la matrice d'entrée est très creuse et a un grand nombre de constantes d'entrée qui peuvent être utilisées comme pivots reconnus comme stables, permet l'élimination symbolique pour aller assez loin avant que les coefficients ne deviennent trop compliqués pour une implémentation convenable et/ou que le conditionnement numérique soit perdu. C'est un compromis entre la complexité des entrées et la taille de la matrice résultante. Mais l'élimination ne doit pas aller trop loin pour ne pas

$$
\left(\begin{array}{cccc|cccccccccccc|cccc|cccc}
1 & 0 & 0 & 0 & c_{12} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & d_{12}^2 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & c_{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & d_{12}^2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_{12} & 0 & 0 & 0 & 0 & 0 & d_{12}^2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_{12} & 0 & 0 & 0 & 0 & 0 & d_{12}^2 \\
\hline
1 & 0 & 0 & 0 & 0 & c_{13} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & d_{13}^2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & c_{13} & 0 & 0 & 0 & 0 & d_{13}^2 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & c_{13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & d_{13}^2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & c_{13} & 0 & 0 & 0 & 0 & d_{13}^2 \\
\hline
1 & 0 & 0 & 0 & 0 & 0 & c_{14} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & d_{14}^2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & c_{14} & 0 & 0 & 0 & d_{14}^2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & c_{14} & 0 & 0 & 0 & d_{14}^2 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_{14} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & d_{14}^2 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & c_{23} & 0 & 0 & 0 & d_{23}^2 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & c_{23} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & d_{23}^2 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & c_{23} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & d_{23}^2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_{23} & 0 & 0 & 0 & d_{23}^2 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & c_{24} & 0 & 0 & 0 & d_{24}^2 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & c_{24} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & d_{24}^2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & c_{24} & 0 & 0 & d_{24}^2 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & c_{24} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & d_{24}^2 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & c_{34} & 0 & 0 & 0 & d_{34}^2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_{34} & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & d_{34}^2 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & c_{34} & 0 & 0 & d_{34}^2 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & c_{34} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & d_{34}^2
\end{array}\right)
\left(\begin{array}{c}
x_1^3 \\ x_2^3 \\ x_3^3 \\ x_4^3 \\ \hline
x_1^2\,x_2 \\ x_1^2\,x_3 \\ x_1^2\,x_4 \\ x_2^2\,x_1 \\ x_2^2\,x_3 \\ x_2^2\,x_4 \\ x_3^2\,x_2 \\ x_3^2\,x_1 \\ x_3^2\,x_4 \\ x_4^2\,x_2 \\ x_4^2\,x_3 \\ x_4^2\,x_1 \\ \hline
x_1\,x_2\,x_3 \\ x_1\,x_2\,x_4 \\ x_1\,x_3\,x_4 \\ x_2\,x_3\,x_4 \\ \hline
-x_1 \\ -x_2 \\ -x_3 \\ -x_4
\end{array}\right) = 0
$$

FIG. 2 – *Le problème linéarisé*

rendre trop difficile l'extraction de la solution à partir des monômes restants. De plus, la réduction change la taille de l'erreur effective, de sorte que cela peut affecter la précision des résultats.

Dans ce cas, nous expérimentons avec les réductions 12×12 et 9×9. A chaque étape de la réduction Gaussienne nous choisissons un pivot constant de sorte à ne pas avoir de division polynomiale, et permutons les lignes en conséquence. La version 12×12 élimine tous les termes avec des x_1, x_2 ou x_3 au carré ou au cube. En éliminant aussi les termes x_4^2 et x_4^3 cela donne une matrice 8×8 en:

$$(x_1\,x_2\,x_3, \ldots, x_2\,x_3\,x_4,\, x_1, \ldots, x_4)$$

. Mais nous jugeons les coefficients beaucoup trop complexes, et en éliminant le terme x_4^3 cela rend la récupération de l'échelle de la solution plus difficile, aussi nous préferrons laisser x_4^3 et avoir une matrice résultante 9×9. Les matrices 12×12 et 9×9 sont un peu trop grandes pour être présentées ici, mais peuvent être demandées aux auteurs.

6 Résultats expérimentaux

Cette section présente un comparatif entre des expériences testant le nouvel algorithme développé ici et, quelques algorithmes plus anciens. Les méthodes portent les noms abrégés comme suit :

4pt 3×3	L'algorithme des 4 points présenté dans [16].
3pt	L'élimination classique basée sur la méthode des 3 points [16].
3pt 5×5	La méthode 5×5 des 3 points du système propre.
4pt 9×9	La méthode 9×9 pour 4 points.
4pt 12×12	La méthode 12×12 pour 4 points.
4pt 24×24	La méthode 24×24 pour 4 points.

Nous avons donc testé une variante de chaque algorithme qui réordonne heuristiquement les points d'entrée de sorte que les points de référence qui sont utilisés par l'algorithme soient aussi dispersés que possible dans l'image.

Dans toutes les expériences, les points sont projetés avec une distance focale de 1024 pixels dans une image 512×512. Un bruit gaussien est additionné, avec un sigma par défaut de 1 pixel.

Chaque point des graphs représente 200 tirages de points 3D généré au hasard dans un nuage gaussien de déviation standard 1 unité, vu par environ 5 unités plus loin. Cependant, dans l'expérience sur les singularités, il y a eu 500 essais par point et seul le bruit a varié entre les essais, pas les points 3D. Dans l'expérience sur la coplanarité, le nuage est d'abord projeté sur un plan. Les résultats représentent les erreurs médianes, relatives à la taille totale de la translation et à la rotation de 1 radian. Le taux d'échec a été mesuré plutôt arbitrairement, comme le pourcentage d'essais total où, soit l'erreur de rotation, soit l'erreur de translation était supérieur à 0.5 unités.

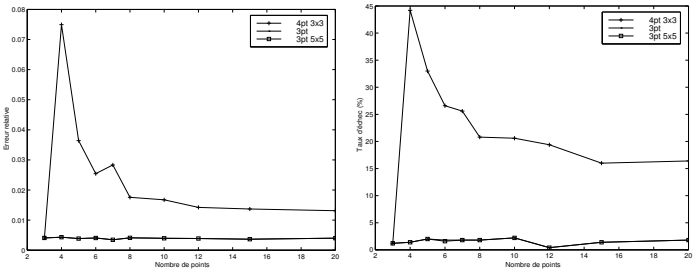


FIG. 3 – *Erreur de translation relative et taux d'échec versus le nombre de points d'entrée.*

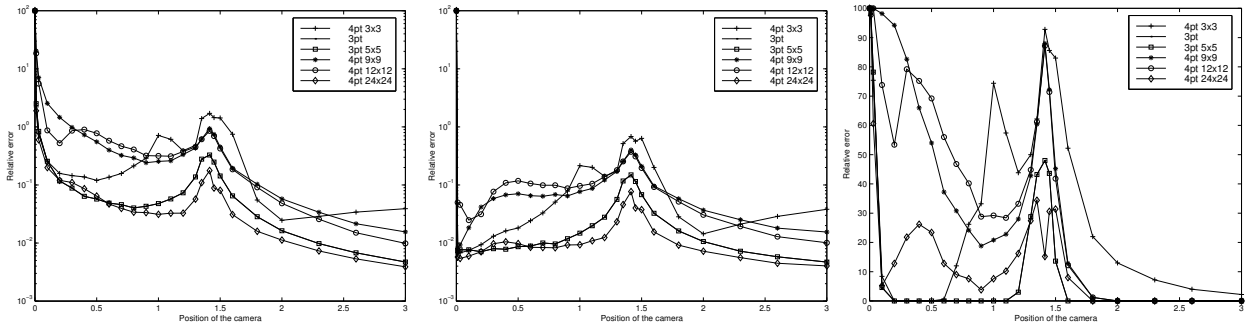


FIG. 4 – *Translation relative et erreur de rotation et taux d'échec quand la caméra bouge d'une configuration critique à une autre pour le problème de pose, à la position parametre 0 et* $\sqrt{2}$.

La figure 10 montre le comportement des différentes méthodes avec 4 points d'entrée, quand le niveau de bruit augmente. La figure 3 montre le comportement quand des points supplémentaires sont additionnés à l'ensemble de données, pour les méthodes qui peuvent gérer des points supplémentaires.

Comme mentionné dans l'introduction, le problème de pose possède quelques cas singuliers qui cause souvent des problèmes dans la pratique. La figure 4 montre les résultats des erreurs et taux d'échec pour une telle configuration. Les données sont 4 points coplanaires dans un carré $[-1,1] \times [-1,1]$ et la caméra commence à la 'position=0', en un point singulier (pour la méthode de 4 points) directement au-dessus de leur centre. La caméra bouge ensuite parallèlement au plan des quatres points selon une droite, en se dirigeant vers un des angles du carré. A la 'position= $\sqrt{2}$' unités elle croise le coté du cylindre circulaire vertical qui correspond aux 4 points de donnée, où une autre singularité se produit.

La principale conclusion de ces expériences est que — en se basant sur le fait qu'elles retournent des solutions possibles multiples qui peuvent ne pas convenir — l'algorithme des 3-points surpasse significativement tous les algorithmes de 4 points linéaires. Même si les méthodes linéaires utilisent plus de données et intègrent des redondances, leurs erreurs relatives et leurs taux d'échec sont souvent 2 à 10 fois supérieurs à ceux des méthodes 3 points. La normalisation des données conventionnelles ne semble pas aider ici, mais des méthodes plus sophistiquées sont possibles.

Les points coplanaires ne sont pas un cas singulier pour aucune des méthodes testées ici, et d'une manière générale leur performances sont similaires au cas non-coplanaires, excepté que l'avantage de performance des méthodes 3 points est décroissant. En augmentant le nombre de points pour les algorithmes linéaires cela améliore légèrement les résultats, mais pas suffisament pour égaler les méthodes des trois points. Pour les méthodes qui traitent les points asymétriquement, en choisissant des points image éparpillés comme points de base, cela semble améliorer les résultat en moyenne. Il existe sûrement de meilleures heuristiques que le choix de points éparpillés pour trouver des configurations de base stable. En particulier, avec de nombreux points dans un nuage gaussien, il y a une légère tendance, en choisissant des points éparpillés, de choisir des configurations proche de la dégénéréscence 'camera au dessus du cercle de points', qui réduit la stabilité moyenne.

L'élimination traditionelle et la nouvelle méthode, basé sur un calcul de vecteur propre, pour le calcul de pose à partir de 3 points ont des performances très proches dans tous les cas testés, la méthode du vecteur propre ayant peut être une toute petite avance.

La performance des méthodes linéaires 24×24, 12×12 et 9×9 est ainsi similaire, avec 24×24 ayant un léger avantage dans la précision globale, mais étant significativement plus lent que la méthode 9×9. Ce ne sont que des tendances statistiques — dans chaque problème il est très difficile de prévoir laquelle va le mieux fonctionner.

Les erreurs, et spécialement les taux d'échec, de toutes les méthodes sont significativement plus élevées que ce que l'on voudrait, spécialement pour les méthodes de 4 points. Ceci est en partie du à la génération alléatoire des données qui est souvent proche d'une configuration particulière.

Nous avons aussi effectué des tests avec des méthodes robustes sur l'algorithme algébrique, et l'algorithme linéaire 24×24. Nous avons comparé ces résultats avec une "méthode combinée" qui calcule la solution avec l'un et l'autre des algorithmes puis choisit celui ayant le moins de valeurs abérrantes entre les deux méthodes sur 300 échantillons aléatoires sur 30 points aléatoires. L'algo-

rithme RANSAC pour la méthode combinée a moitié moins d'itérations que les autres méthodes pour obtenir la même efficacité en temps. Le nombre de valeurs abérrantes est estimé à partir d'un seuil fixé sur les erreurs de reprojection. Nous utilisons 1, 3 et 5 pixels comme seuils.

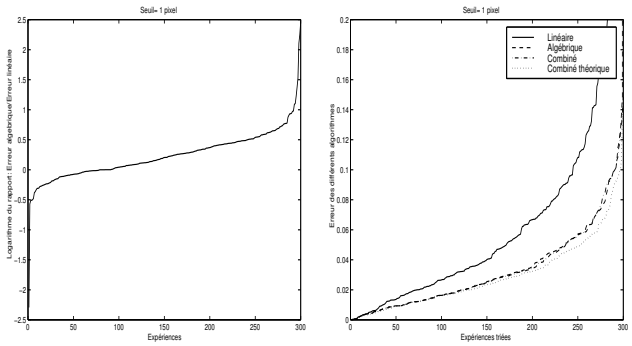


FIG. 5 – *Comparaison des 3 méthodes avec 1 pixel comme seuil de reprojection.*

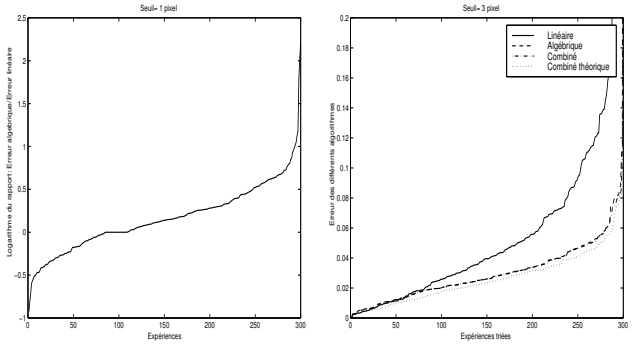


FIG. 6 – *Comparaison des 3 méthodes avec 3 pixels comme seuil de reprojection.*

Nous avons remarqué que l'algorithme combiné est un tout petit peu plus robuste (voir Figure 6). Nous avons effectué 7500 tests pour chacune des trois méthodes, ensuite nous avons trié les erreurs résiduelles et finallement zoomé sur les 100 pires tests de chaque méthode. L'algorithme combiné réduit la zone d'instabilité de l'estimation sur les méthodes individuelles ayant différentes configurations instables.

Les résultats avec les expériences sur des images réelles avec des données de calibration connues sont présentés à la Figure 9.

7 Conclusions

Nous avons présenté une nouvelle méthode pour l'estimation de pose à partir de 3 points, basé sur un calcul de vecteur propre, et trois nouveaux algorithmes linéaires

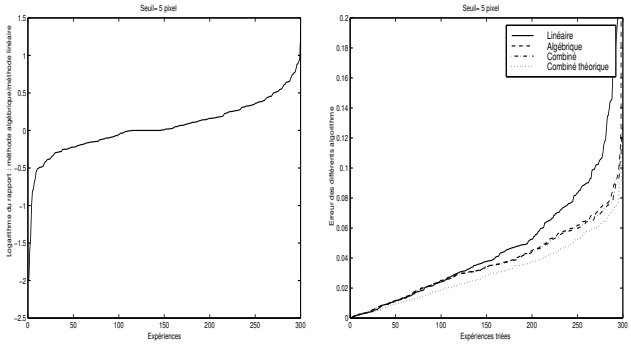


FIG. 7 – *Comparaison des 3 méthodes avec 5 pixels comme seuil de reprojection.*

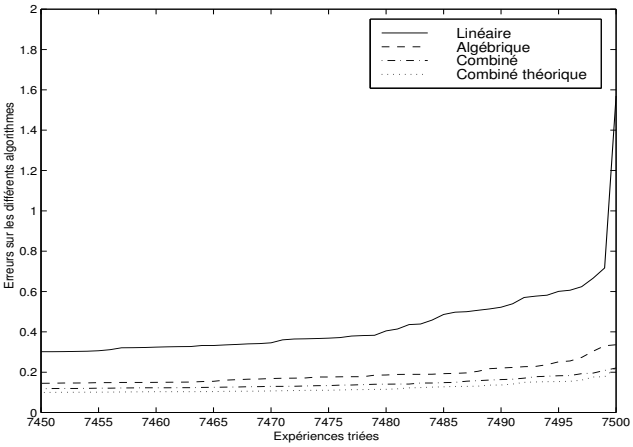


FIG. 8 – *La robustesse de la méthode combiné.*

pour l'estimation de pose à partir de 4 points. Le principal avantage des algorithmes linéaires est qu'ils génèrent une solution unique. Aucune des méthodes ne dégénère pour des points (génériques) coplanaires. Toutes les méthodes développées ici sont faciles à implémenter dans le sens où leur matrices sont des fonctions relativement simples des coordonnées d'entrée. La méthode du vecteur propre surpasse légèrement l'élimination traditionnelle et est recommandée pour être utilisée en applications. Les méthodes linéaires à 4 points demandent des méthodes de normalisation des données plus sophistiquées pour être plus pratique.

Références

[1] H.H. Chen. Pose determination from line-to-plane correspondence: Existence condition and closed-form solutions. In *Proceedings of the 3rd International Conference on Computer Vision, Osaka, Japan*, pages 374–378, 1990.

[2] D.Cox, J.Little, and D.O'Shea. *Using Algebraic Geometry*. Springer, 1998.

FIG. 9 – *Les points réprojetés sur l'image originale par (a) la méthode algébrique (b) la méthode linéaire* 24×24 *(c) la méthode linéaire non linéairement optimisée* 24×24.

[3] D. Dementhon and L.S. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15(1/2):123–141, 1995.

[4] M. Dhome, M. Richetin, J.T. Lapresté, and G. Rives. Determination of the attitude of 3D objects from a single perspective view. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1265–1278, December 1989.

[5] I.Z. Emeris. A general solver based on sparse resultants: Numerical issues and kinematic applications. Technical Report RR-3110, INRIA, 1997.

[6] O. Faugeras and M. Hebert. The representation, recognition, and locating of 3D objects. *The International Journal of Robotics Research*, 5:27–52, 1986.

[7] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Graphics and Image Processing*, 24(6):381 – 395, June 1981.

[8] W. Förstner. Reliability analysis of parameter estimation in linear models with applications to mensuration problems in computer vision. *Computer Vision, Graphics and Image Processing*, 40:273–310, 1987.

[9] R.M. Haralick, C. Lee, K. Ottenberg, and M. Nölle. Analysis and solutions of the three point perspective pose estimation problem. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Maui, Hawaii, USA*, pages 592–598, 1991.

[10] R. Horaud, B. Conio, O. Leboulleux, and B. Lacolle. An analytic solution for the perspective 4-point problem. *Computer Vision, Graphics and Image Processing*, 47:33–44, 1989.

[11] B.K.P. Horn. Closed form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

[12] Y. Liu, T.S. Huang, and O.D. Faugeras. Determination of camera location from 2D to 3D line and point. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 12(1):28–37, January 1990.

[13] D. Lowe. Fitting parameterized three-dimensional models to images. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441–450, May 1991.

[14] D.G. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, Norwell, Massachusets, 1985.

[15] B. Mourrain and V.Y. Pan. Multivariate polynomials, duality and structured matrices. Technical report, INRIA Sophia-Antipolis, October 1998.

[16] L. Quan and Z.D. Lan. Linear n-point camera pose determination. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 21(8):774–780, August 1999.

[17] C.C. Slama, editor. *Manual of Photogrammetry, Fourth Edition*. American Society of Photogrammetry and Remote Sensing, Falls Church, Virginia, USA, 1980.

[18] E.H. Thompson. Space resection: Failure cases. *Photogrammetric Record*, X(27):201–204, 1966.

[19] B. Triggs. Camera pose and calibration from 4 or 5 known 3d points. In *Proceedings of the 7th International Conference on Computer Vision, Kerkyra, Greece*, pages 278–284, Kerkyra, Greece, September 1999.

[20] B.P. Wrobel. Minimum solutions for orientation. In *Proc. of the Workshop on Calibration and Orientation of Cameras in Computer Vision, Washington D.C., USA*. Springer-Verlag, August 1992.

[21] J.S.C. Yuan. A general phogrammetric solution for the determining object position and orientation. IEEE *Transactions on Robotics and Automation*, 5(2):129–142, April 1989.

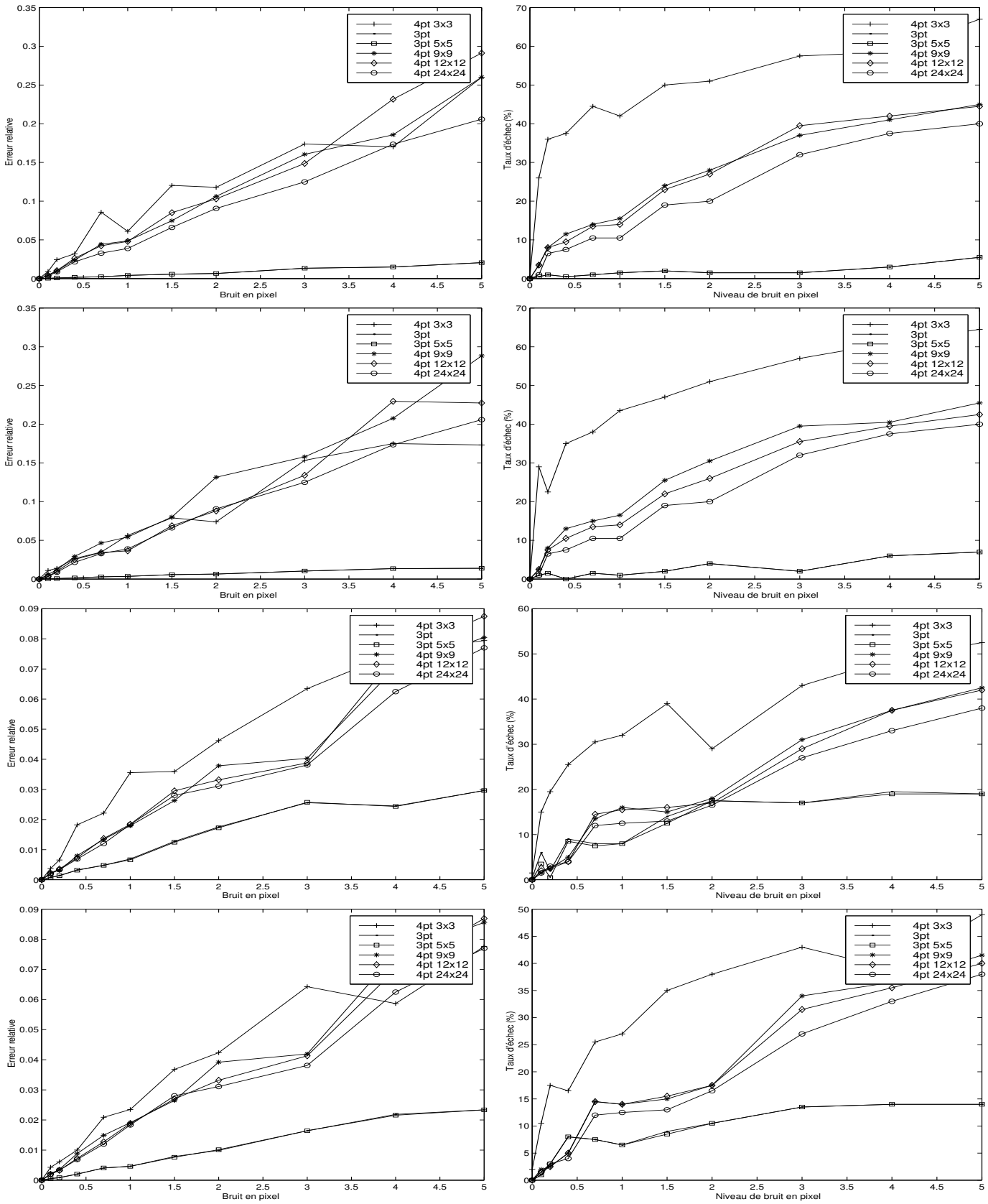


FIG. 10 – *L'erreur de translation relative et le taux d'échec versus le niveau de bruit en pixels, pour 4 points. Les deux premières lignes sont pour les données non coplanaires, les deux suivantes pour les données coplanaires. Les première et troisième lignes utilisent l'ordre des points d'entrée, la seconde et la quatrième réordonnent heuristiquement les points pour une meilleure stabilité avant de lancer les méthodes de pose. La tendance de l'erreur de rotation est très proche de celle de la translation, et n'est pas montrée par manque de place.*

Plane + Parallax, Tensors and Factorization

Bill Triggs
INRIA Rhône-Alpes,
655 avenue de l'Europe, 38330 Montbonnot, France.
Bill.Triggs@inrialpes.fr
http://www.inrialpes.fr/movi/people/Triggs

Abstract

We study the special form that the general multi-image tensor formalism takes under the plane + parallax decomposition, including matching tensors and constraints, closure and depth recovery relations, and inter-tensor consistency constraints. Plane + parallax alignment greatly simplifies the algebra, and uncovers the underlying geometric content. We relate plane + parallax to the geometry of translating, calibrated cameras, and introduce a new parallax-factorizing projective reconstruction method based on this. Initial plane + parallax alignment reduces the problem to a single rank-one factorization of a matrix of rescaled parallaxes into a vector of projection centres and a vector of projective heights above the reference plane. The method extends to 3D lines represented by via-points and 3D planes represented by homographies.

Keywords: Plane + parallax, matching tensors, projective reconstruction, factorization, structure from motion.

1 Introduction

This paper studies the special forms that matching tensors take under the plane + parallax decomposition, and uses this to develop a new projective reconstruction method based on rank-1 parallax factorization. The main advantage of the plane + parallax analysis is that it greatly simplifies the usually rather opaque matching tensor algebra, and clarifies the way in which the tensors encode the underlying 3D camera geometry. The new plane + parallax factorizing reconstruction method appears to be even stabler than standard projective factorization, especially for near-planar scenes. It is a one-step, closed form, multi-point, multi-image factorization for projective structure, and in this sense improves on existing minimal-configuration and iterative depth recovery plane + parallax SFM methods [19,4,23,22,45]. As with standard projective factorization [37], it can be extended to handle 3D lines (via points) and planes (homographies) alongside 3D points.

Matching tensors [29,8,36] are the image signature of the camera geometry. Given several perspective images of the same scene taken from different viewpoints, the 3D camera geometry is encoded by a set of 3×4 homogeneous camera projection matrices. These depend on the chosen 3D coordinate system, but the dependence can be eliminated algebraically to give four series of multi-image **tensors** (multi-index arrays of components), each interconnecting 2–4 images. The different

Published in European Conference on Computer Vision, 2000.
This work was supported by European Union LTR project CUMULI. I would like to thank A. Zisserman and P. Anandan for comments.

images of a 3D feature are constrained by multilinear **matching relations** with the tensors as coefficients. These relations can be used to estimate the tensors from an initial set of correspondences, and the tensors then constrain the search for further correspondences. The tensors implicitly characterize the relative projective camera geometry, so they are a useful starting point for 3D reconstruction. Unfortunately, they are highly redundant, obeying a series of complicated internal self-consistency constraints whose general form is known but too complex to use easily, except in the simplest cases [36,5,17,6].

On the other hand, a camera is simply a device for recording incoming light in various directions at the camera's optical centre. Any two cameras with the same centre are equivalent in the sense that — modulo field-of-view and resolution constraints which we ignore for now — they see exactly the same set of incoming light rays. So their images can be warped into one another by a 1-1 mapping (for projective cameras, a 2D homography). Anything that can be done using one of the images can equally well be done using the other, if necessary by pre-warping to make them identical.

From this point of view, it is clear that the camera centres are the essence of the *3D* camera geometry. Changing the camera orientations or calibrations while leaving the centres fixed amounts to a 'trivial' change of image coordinates, which can be undone at any time by homographic (un)warping. In particular, the algebraic structure (degeneracy, number of solutions, *etc*.) of the matching constraints, tensors and consistency relations — and *a fortiori* that of any visual reconstruction based on these — is essentially a 3D matter, and hence depends *only* on the camera centres.

It follows that much of the complexity of the matching relations is only apparent. At bottom, the geometry is simply that of a configuration of 3D points (the camera centres). But the inclusion of arbitrary calibration-orientation homographies everywhere in the formulae makes the algebra appear much more complicated than need be. One of the main motivations for this work was to study the matching tensors and relations in a case — that of projective plane + parallax alignment — where most of the arbitrariness due to the homographies has been removed, so that the underlying geometry shows up much more clearly.

The observation that the camera centres lie at the heart of the projective camera geometry is by no means new. It is the basis of Carlsson's 'duality' between 3D points and cameras (*i.e*. centres) [2,43,3,10], and of Heyden & Åström's closely related 'reduced tensor' approach [13,14,15,17]. The growing geometry tradition in the plane + parallax literature [19,23,22,4,45] is also particularly relevant here.

Organization: §2 introduces our plane + parallax representation and shows how it applies to the basic feature types; §3 displays the matching tensors and constraints in the plane + parallax representation; §4 discusses tensor scaling, redundancy and consistency; §5 considers the tensor closure and depth recovery relations under plane + parallax ; §6 introduces the new parallax factorizing projective reconstruction method; §7 shows some initial experimental results; and §8 concludes.

Notation: Bold italic '$\boldsymbol{x}$' denotes 3-vectors, bold sans-serif 'x' 4-vectors, upper case '$\boldsymbol{H}, \mathsf{H}$' matrices, Greek '$\lambda, \mu$' scalars (*e.g*. homogeneous scale factors). We use homogeneous coordinates for 3D points x and image points $\boldsymbol{x}$, but usually inhomogeneous ones $\boldsymbol{c}$ for projection centres $\mathsf{c} = \begin{pmatrix} \boldsymbol{c} \\ 1 \end{pmatrix}$. We use P for 3×4 camera projection matrices, $\boldsymbol{e}$ for epipoles. 3D points $\mathsf{x} = \begin{pmatrix} \boldsymbol{x} \\ w \end{pmatrix}$ are parametrized by a point $\boldsymbol{x}$ on the reference plane and a 'projective height' w above it. $\wedge$ denotes cross-product, $[-]_\times$ the associated 3×3 skew matrix $[\boldsymbol{x}]_\times \boldsymbol{y} = \boldsymbol{x} \wedge \boldsymbol{y}$, and $[\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}]$ the triple product.

2 The Plane + Parallax Representation

Data analysis is often simplified by working in terms of small corrections against a reference model. Image analysis is no exception. In **plane + parallax**, the reference model is a real or virtual **reference plane** whose points are held fixed throughout the image sequence by image warping (see, *e.g.* [24, 33, 19] and their references). The reference is often a perceptually dominant plane in the scene such as the ground plane. Points that lie above the plane are not exactly fixed, but their motion can be expressed as a residual **parallax** with respect to the plane. The parallax is often much smaller than the uncorrected image motion, particularly when the camera motion is mainly rotational. This simplifies feature extraction and matching. For each projection centre, alignment implicitly defines a unique reference orientation and calibration, and in this sense entirely cancels any orientation and calibration variations. Moreover, the residual parallaxes directly encode useful structural information about the size of the camera translation and the distance of the point above the plane. So alignment can be viewed as a way of focusing on the essential *3D* geometry — the camera centres and 3D points — by eliminating the 'nuisance variables' associated with orientation and calibration. The 'purity' of the parallax signal greatly simplifies many geometric computations. In particular, we will see that it dramatically simplifies the otherwise rather cumbersome algebra of the matching tensors and relations (*c.f.* also [19, 22, 4]).

The rest of this section describes our "plane at infinity + parallax" representation. It is projectively equivalent to the more common "ground plane + parallax" representation (*e.g.* [19, 42]), but has algebraic advantages — simpler formulae for scale factors, and the link to translating cameras — that will be discussed below.

Coordinate frame: We suppose given a 3D **reference plane** with a predefined projective coordinate system, and a 3D **reference point** not on the plane. The plane may be real or virtual, explicit or implicit. The plane coordinates might derive from an image or be defined by features on the plane. The reference point might be a 3D point, a projection centre, or arbitrary. We adopt a projective 3D coordinate system that places the reference point at the 3D origin $(0\ 0\ 0\ 1)^\top$, and the reference plane at infinity in standard position (*i.e.* its reference coordinates coincide with the usual coordinates on the plane at infinity). Examining the possible residual 4×4 homographies shows that this fixes the 3D projective frame up to a single global scale factor. If $\mathsf{H} = \left(\begin{smallmatrix} \boldsymbol{A} & \boldsymbol{t} \\ \boldsymbol{b}^\top & \lambda \end{smallmatrix}\right)$, then the constraint that H fixes each point $\left(\begin{smallmatrix} \boldsymbol{x} \\ 0 \end{smallmatrix}\right)$ on the reference plane implies that $\boldsymbol{A} = \mu \boldsymbol{I}$ and $\boldsymbol{b} = \boldsymbol{0}$, and the constraint that H fixes the origin $\left(\begin{smallmatrix} \boldsymbol{0} \\ 1 \end{smallmatrix}\right)$ implies that $\boldsymbol{t} = \boldsymbol{0}$. So $\mathsf{H} = \left(\begin{smallmatrix} \mu \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \lambda \end{smallmatrix}\right)$, which is a global scaling by μ/λ.

3D points: 3D points are represented as linear combinations of the reference point/origin and a point on the reference plane:

$$\mathsf{x} \equiv \begin{pmatrix} \boldsymbol{x} \\ w \end{pmatrix} = \begin{pmatrix} \boldsymbol{x} \\ 0 \end{pmatrix} + w \begin{pmatrix} \boldsymbol{0} \\ 1 \end{pmatrix} \tag{1}$$

$\boldsymbol{x}$ is the intersection with the reference plane, of the line through x and the origin. w is called x's **projective height** above the plane. $w = 0$ is the reference plane, $w = \infty$ the origin. w depends on the normalization convention for $\boldsymbol{x}$. If the reference plane is made finite ($z = 0$) by interchanging z and w coordinates, w becomes the vertical height above the plane. But in our projective, plane-at-infinity based frame with affine normalization $\boldsymbol{x}_z = 1$, w is the inverse z-distance (or with spherical normalization $\|\boldsymbol{x}\| = 1$, the inverse "Euclidean" distance) of x from the origin.

Camera matrices: Plane + parallax aligned cameras fix the image of the reference plane, so their

leading 3×3 submatrix is the identity. They are parametrized simply by their projection centres:

$$\mathsf{P} \;=\; \begin{pmatrix} u\boldsymbol{I}_{3\times 3} & -\boldsymbol{c} \end{pmatrix} \qquad \text{with projection centre} \qquad \mathsf{c} \;=\; \begin{pmatrix} \boldsymbol{c} \\ u \end{pmatrix} \tag{2}$$

Hence, any 3D point can be viewed as a plane + parallax aligned camera and vice versa. But, whereas points often lie on or near the reference plane ($w \to 0$), cameras centred on the plane ($u \to 0$) are too singular to be useful — they project the entire 3D scene to their centre point $\boldsymbol{c}$.

We will break the nominal symmetry between points and cameras. Points will be treated projectively, as general homogeneous 4-component quantities with arbitrary height component w. But camera centres $\mathsf{c} = \left(\begin{smallmatrix} \boldsymbol{c} \\ u \end{smallmatrix}\right)$ will be assumed to lie outside the reference plane and scaled affinely ($u \to 1$), so that they and their camera matrices $\mathsf{P} = \begin{pmatrix} u\boldsymbol{I} & -\boldsymbol{c} \end{pmatrix}$ are parametrized by their *in*homogeneous centre 3-vector $\boldsymbol{c}$ alone.

This asymmetry is critical to our approach. Our coordinate frame and reconstruction methods are essentially projective and are most naturally expressed in homogeneous coordinates. Conversely, scaling u to 1 freezes the scales of the projection matrices, and everywhere that matching tensors are used, it converts formulae that would be bilinear or worse in the $\boldsymbol{c}$'s and u's, to ones that are merely *linear* in the $\boldsymbol{c}$'s. This greatly simplifies the tensor estimation process compared to the general unaligned case. The representation becomes singular for cameras near the reference plane, but that is not too much of a restriction in practice. In any case it *had* to happen — no minimal linear representation can be globally valid, as the general redundant tensor one is.

Point projection: In image i, the image $\mathsf{P}_i\,\mathsf{x}_p$ of a 3D point $\mathsf{x}_p = \left(\begin{smallmatrix} \boldsymbol{x}_p \\ w_p \end{smallmatrix}\right)$ is displaced linearly from its reference image[1] $\boldsymbol{x}_p$ towards the centre of projection $\boldsymbol{c}_i$, in proportion to its height w_p :

$$\lambda_{ip}\boldsymbol{x}_{ip} \;=\; \mathsf{P}_i\,\mathsf{x}_p \;=\; \begin{pmatrix} \boldsymbol{I} & -\boldsymbol{c}_i \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_p \\ w_p \end{pmatrix} \;=\; \boldsymbol{x}_p - w_p\,\boldsymbol{c}_i \tag{3}$$

Here λ_{ip} is a **projective depth** [32,37] — an initially-unknown projective scale factor that compensates for the loss of the scale information in $\mathsf{P}_i\,\mathsf{x}_p$ when $\boldsymbol{x}_{ip}$ is measured in its image. Although the homogeneous rescaling freedom of x_p makes them individually arbitrary, the combined projective depths of a 3D point — or more precisely its vector of **rescaled image points** $(\lambda_{ip}\boldsymbol{x}_{ip})_{i=1\ldots m}$ — implicitly define its 3D structure. This is similar to the general projective case, except that in plane + parallax the projection matrix scale freedom is already frozen: while the homogeneous scale factors of x_p and $\boldsymbol{x}_{ip}$ are arbitrary, $\boldsymbol{c}_i$ has a fixed scale linked to its camera's position.

Why not a ground plane?: In many applications, the reference plane is nearby. Pushing it out to infinity forces a deeply projective 3D frame. It might seem preferable to use a finite reference plane, as in, *e.g.* [19,42]. For example, interchanging the z and w coordinates puts the plane at $z = 0$, the origin at the vertical infinity $(0\ 0\ 1\ 0)^\top$, and (modulo Euclidean coordinates on the plane itself) creates an obviously rectilinear 3D coordinate system, where $\boldsymbol{x}$ gives the ground coordinates and w the vertical height above the plane. However, a finite reference plane would hide a valuable insight that is obvious from (2): *The plane + parallax aligned camera geometry is projectively equivalent to translating calibrated cameras.* Any algorithm that works for these works for projective plane + parallax, and (up to a 3D projectivity!) vice versa. Although not new (see, *e.g.* [14]), this analogy deserves to be better known. It provides simple algebra and geometric intuition that were very helpful during this work. It explicitly realizes — albeit in a weak, projectively distorted sense, with the

[1] The origin/reference point need not coincide with a physical camera, but can still be viewed as a **reference camera** $\mathsf{P}_0 = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \end{pmatrix}$, projecting 3D points x_p to their reference images $\boldsymbol{x}_p$.

reference plane mapped to infinity — the suggestion that plane + parallax alignment cancels the orientation and calibration, leaving only the translation [22].

3D Lines: Any 3D line L can be parametrized by a homogeneous 6-tuple of Plücker coordinates $(\boldsymbol{l},\boldsymbol{z})$ where: (*i*) $\boldsymbol{l}$ is a line 3-vector — L's projection from the origin onto the reference plane; (*ii*) $\boldsymbol{z}$ is a point 3-vector — L's intersection with the reference plane; (*iii*) $\boldsymbol{z}$ lies on $\boldsymbol{l}$, $\boldsymbol{l}\cdot\boldsymbol{z} = 0$ (this is the Plücker constraint); (*iv*) the relative scaling of $\boldsymbol{l}$ and $\boldsymbol{z}$ is fixed and gives L's 'steepness': lines on the plane have $\boldsymbol{z} \to \boldsymbol{0}$, while the ray from the origin to $\boldsymbol{z}$ has $\boldsymbol{l} \to \boldsymbol{0}$. This parametrization of L relates to the usual 3D projective Plücker (4×4 skew rank 2 matrix) representations as follows:

$$\mathsf{L}^* = \begin{pmatrix} [\boldsymbol{l}]_\times & \boldsymbol{z}^\top \\ -\boldsymbol{z} & 0 \end{pmatrix} \quad \begin{matrix}\text{contravariant}\\ \text{form}\end{matrix} \qquad \mathsf{L}_* = \begin{pmatrix} [\boldsymbol{z}]_\times & \boldsymbol{l}^\top \\ -\boldsymbol{l} & 0 \end{pmatrix} \quad \begin{matrix}\text{covariant}\\ \text{form}\end{matrix} \tag{4}$$

The line from $\binom{\boldsymbol{x}}{w}$ to $\binom{\boldsymbol{y}}{v}$ is $(\boldsymbol{l},\boldsymbol{z}) = (\boldsymbol{x}\wedge\boldsymbol{y}, w\boldsymbol{y} - v\boldsymbol{x})$. A 3D point $\mathsf{x} = \binom{\boldsymbol{x}}{w}$ lies on L iff $\mathsf{L}_*\mathsf{x} = \binom{w\boldsymbol{l}+\boldsymbol{z}\wedge\boldsymbol{x}}{\boldsymbol{l}\cdot\boldsymbol{x}} = \mathsf{0}$. In a camera at c_i, L projects to:

$$\mu_i \boldsymbol{l}_i = \boldsymbol{l} + \boldsymbol{z}\wedge\boldsymbol{c}_i \tag{5}$$

This vanishes if c_i lies on L.

Displacements and epipoles: Given two cameras with centres $\mathsf{c}_i = \binom{\boldsymbol{c}_i}{1}$ and $\mathsf{c}_j = \binom{\boldsymbol{c}_j}{1}$, the **3D displacement vector** between their two centres is $\boldsymbol{c}_{ij} = \boldsymbol{c}_i - \boldsymbol{c}_j$. The scale of $\boldsymbol{c}_{ij}$ is meaningful, encoding the relative 3D camera position. Forgetting this scale factor gives the **epipole** $\boldsymbol{e}_{ij}$ — the 2D projective point at which the ray from c_j to c_i crosses the reference plane:

$$\boldsymbol{e}_{ij} \simeq \boldsymbol{c}_{ij} \equiv \boldsymbol{c}_i - \boldsymbol{c}_j \tag{6}$$

We will see below that it is really the inter-camera displacements $\boldsymbol{c}_{ij}$ and not the epipoles $\boldsymbol{e}_{ij}$ that appear in tensor formulae. Correct relative scalings are essential for geometric coherence, but precisely because of this they are also straightforward to estimate. Once found, the displacements $\boldsymbol{c}_{ij}$ amount to a reconstruction of the plane + parallax aligned camera geometry. To find a corresponding set of camera centres, simply fix the 3D coordinates of one centre (or function of the centres) arbitrarily, and the rest follow immediately by adding displacement vectors.

Parallax: Subtracting two point or line projection equations (3, 5) gives the following important **parallax equations**:

$$\lambda_i \boldsymbol{x}_i - \lambda_j \boldsymbol{x}_j = -w\,\boldsymbol{c}_{ij} \tag{7}$$

$$\mu_i \boldsymbol{l}_i - \mu_j \boldsymbol{l}_j = \boldsymbol{z}\wedge\boldsymbol{c}_{ij} \tag{8}$$

Given the correct projective depths λ,μ, the relative parallax caused by a camera displacement is proportional to the displacement vector. The RHS of (7) already suggests the possibility of factoring a multi-image, multi-point matrix of rescaled parallaxes into (w) and $(\boldsymbol{c}_{ij})$ matrices. Results equivalent to (7) appear in [19,22], albeit with more complicated scale factors owing to the use of different projective frames.

Equation (7) has a trivial interpretation in terms of 3D displacements. For a point $\mathsf{x} = \binom{\boldsymbol{x}}{w}$ above the reference plane, scaling to $w = 1$ gives projection equations $\lambda_i\boldsymbol{x}_i = \mathsf{P}_i\mathsf{x} = \boldsymbol{x} - \boldsymbol{c}_i$, so $\lambda_i\boldsymbol{x}_i$ is the 3D displacement vector from c_i to x. (7) just says that the sum of displacements around the 3D triangle $\overline{\mathsf{c}_i\,\mathsf{c}_j\,\mathsf{x}}$ vanishes. On the reference plane, this entails the alignment of the 2D points $\boldsymbol{x}_i, \boldsymbol{x}_j$ and $\boldsymbol{e}_{ij}$ (along the line of intersection of the 3D plane of $\overline{\mathsf{c}_i\,\mathsf{c}_j\,\mathsf{x}}$ with the reference plane — see fig. 1),

and hence the vanishing of the triple product $[\boldsymbol{x}_i, \boldsymbol{e}_{ij}, \boldsymbol{x}_j] = 0$. However the 3D information in the relative scale factors is more explicit in (7).

3D Planes: The 3D plane $\mathsf{p} = (\boldsymbol{n}^\top\ d)$ has equation $\mathsf{p}\cdot\mathsf{x} = \boldsymbol{n}\cdot\boldsymbol{x} + d\,w = 0$. It intersects the reference plane in the line $\boldsymbol{n}\cdot\boldsymbol{x} = 0$. The relative scaling of $\boldsymbol{n}$ and d gives the 'steepness' of the 3D plane: $\boldsymbol{n} = \boldsymbol{0}$ for the reference plane, $d = 0$ for planes through the origin. A point $\boldsymbol{x}_j$ in image j back-projects to the 3D point $\mathsf{B}_j\boldsymbol{x}_j$ on p, which induces an image j to image i homography $\boldsymbol{H}_{ij}$, where:

$$\mathsf{B}_j(\mathsf{p}) \equiv \begin{pmatrix} \boldsymbol{I} - \boldsymbol{c}_j\boldsymbol{n}^\top/(\boldsymbol{n}\cdot\boldsymbol{c}_j + d) \\ -\boldsymbol{n}^\top/(\boldsymbol{n}\cdot\boldsymbol{c}_j + d) \end{pmatrix} \qquad \boldsymbol{H}_{ij}(\mathsf{p}) = \mathsf{P}_i\,\mathsf{B}_j = \boldsymbol{I} + \frac{\boldsymbol{c}_{ij}\,\boldsymbol{n}^\top}{\boldsymbol{n}\cdot\boldsymbol{c}_j + d} \tag{9}$$

For any i, j and any p, this fixes the epipole $\boldsymbol{e}_{ij}$ and each point on the intersection line $\boldsymbol{n}\cdot\boldsymbol{x} = 0$. $\boldsymbol{H}_{ij}$ is actually a **planar homology** [30, 11] — it has a double eigenvalue corresponding to the points on the fixed line.

Any chosen plane $\mathsf{p} = (\boldsymbol{n}^\top\ d)$ can be made the reference plane by applying a 3D homography H and compensating image homographies $\boldsymbol{H}_i$:

$$\mathsf{H} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{n}^\top/d & 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{n}^\top/d & 1 \end{pmatrix}^{-1} \qquad \boldsymbol{H}_i = \boldsymbol{I} - \frac{\boldsymbol{c}_i\,\boldsymbol{n}^\top}{\boldsymbol{n}\cdot\boldsymbol{c}_i + d} = \left(\boldsymbol{I} + \frac{\boldsymbol{c}_i\,\boldsymbol{n}^\top}{d}\right)^{-1}$$

Reference positions $\boldsymbol{x}$ are unchanged, projective heights are warped by an affinity $w \rightarrow w + \boldsymbol{n}\cdot\boldsymbol{x}/d$, and camera centres are rescaled $\boldsymbol{c} \rightarrow \frac{d}{\boldsymbol{n}\cdot\boldsymbol{c}+d}\,\boldsymbol{c}$ (infinitely, if they lie on the plane p):

$$\mathsf{x} = \begin{pmatrix} \boldsymbol{x} \\ w \end{pmatrix} \longrightarrow \mathsf{H}\mathsf{x} = \begin{pmatrix} \boldsymbol{x} \\ w + \boldsymbol{n}\cdot\boldsymbol{x}/d \end{pmatrix} \tag{10}$$

$$\mathsf{p} = \begin{pmatrix} \boldsymbol{n}^\top & d \end{pmatrix} \longrightarrow \mathsf{p}\,\mathsf{H}^{-1} = (\boldsymbol{0}\quad d) \tag{11}$$

$$\mathsf{P}_i = \begin{pmatrix} \boldsymbol{I} & -\boldsymbol{c}_i \end{pmatrix} \longrightarrow \boldsymbol{H}_i\,\mathsf{P}_i\,\mathsf{H}^{-1} = \begin{pmatrix} \boldsymbol{I} & \frac{-d\,\boldsymbol{c}_i}{\boldsymbol{n}\cdot\boldsymbol{c}_i + d} \end{pmatrix} \tag{12}$$

When p is the true plane at infinity, the 3D frame becomes affine and the aligned camera motion becomes truly translational.

Given multiple planes p_k and images i, and choosing some fixed base image 0, the 3 columns of each $\boldsymbol{H}_{i0}(\mathsf{p}_k)$ can be viewed as three point vectors and incorporated into the rank-one factorization method below to reconstruct the $\boldsymbol{c}_{i0}$ and $\boldsymbol{n}_k^\top/(\boldsymbol{n}_k\cdot\boldsymbol{c}_0 + d_k)$. Consistent normalizations for the different $\boldsymbol{H}_{i0}$ are required. If $\boldsymbol{e}_{i0}$ is known, the correct normalization can be recovered from $[\boldsymbol{e}_{i0}]_\times\boldsymbol{H}_{i0} = [\boldsymbol{e}_{i0}]_\times$. This amounts to the point depth recovery equation (19) below applied to the columns of $\boldsymbol{H}_{i0}$ and $\boldsymbol{H}_{00} = \boldsymbol{I}$. Alternatively, $\boldsymbol{H}_{i0} = \boldsymbol{I} + \ldots$ has two repeated unit eigenvalues, and the right (left) eigenvectors of the remaining eigenvalue are $\boldsymbol{e}_{i0}$ ($\boldsymbol{n}^\top$). This allows the normalization, epipole and plane normal to be recovered from an estimated $\boldsymbol{H}_{i0}$. Less compact rank 4 factorization methods also exist, based on writing $\boldsymbol{H}_{i0}$ as a 9-vector, linear in the components of $\boldsymbol{I}$ and either $\boldsymbol{c}_{i0}$ or $\boldsymbol{n}_k$ [28, 44, 45].

Carlsson duality: Above we gave the plane + parallax correspondence between 3D points and (the projection centres of aligned) cameras [19, 22]:

$$\mathsf{x} = \begin{pmatrix} \boldsymbol{x} \\ w \end{pmatrix} \quad \longleftrightarrow \quad \mathsf{P} = \begin{pmatrix} w\boldsymbol{I} & -\boldsymbol{x} \end{pmatrix}$$

Carlsson [2, 3] (see also [13, 14, 43, 10, 42]) defined a related but more 'twisted' duality mapping based on the alignment of a projective basis rather than a plane:

$$\mathsf{x} = \begin{pmatrix} \boldsymbol{x} \\ w \end{pmatrix} \quad \longleftrightarrow \quad \mathsf{P} = \begin{pmatrix} 1/x & & & -1/w \\ & 1/y & & -1/w \\ & & 1/z & -1/w \end{pmatrix} \simeq \begin{pmatrix} x & & \\ & y & \\ & & z \end{pmatrix}^{-1} \begin{pmatrix} w\boldsymbol{I} & -\boldsymbol{x} \end{pmatrix}$$

Provided that x, y, z are non-zero, the two mappings differ only by an image homography. Plane + parallax aligns a 3D plane pointwise, thus forcing the image $-\boldsymbol{x}$ of the origin to depend on the projection centre. Carlsson aligns a 3D projective basis, fixing the image of the origin and just 3 points on the plane (and incidentally introducing potentially troublesome singularities for projection centres on the x, y and z coordinate planes, as well as on the $w = 0$ one). In either case the point-camera "duality" (isomorphism would be a better description) allows some or all points to be treated as cameras and vice versa. This has been a fruitful approach for generating new algorithms [2,43, 3,10,42,19,22,4]. All of the below formulae can be dualized, with the proviso that camera centres should avoid the reference plane and be affinely normalized, while points need not and must be treated homogeneously.

3 Matching Tensors and Constraints in Plane + Parallax

Matching Tensors: The matching tensors for aligned projections are very simple functions of the scaled epipoles / projection centre displacements. From a tensorial point of view[2], the simplest way to derive them is to take the homography-epipole decompositions of the generic matching tensors [29,8,36], and substitute identity matrices for the homographies:

$$\begin{aligned}
\boldsymbol{c}_{12} &= \boldsymbol{c}_1 - \boldsymbol{c}_2 && \text{displacement from } \mathtt{c}_2 \text{ to } \mathtt{c}_1 \\
\boldsymbol{F}_{12} &= [\,\boldsymbol{c}_{12}\,]_\times = [\,\boldsymbol{c}_1 - \boldsymbol{c}_2\,]_\times && \text{image 1-2 fundamental matrix} \\
\boldsymbol{T}_1^{23} &= \boldsymbol{I}_1^2 \otimes \boldsymbol{c}_{13} - \boldsymbol{c}_{12} \otimes \boldsymbol{I}_1^3 && \text{image 1-2-3 trifocal tensor} \\
\boldsymbol{Q}^{A_1A_2A_3A_4} &= \textstyle\sum_{i=1}^{3} (-1)^{i-1} \boldsymbol{\epsilon}^{A_1 \ldots \hat{A}_i \ldots A_4} \cdot \boldsymbol{c}_{i4}^{A_i} && \text{image 1-2-3-4 quadrifocal tensor}
\end{aligned}$$

The plane + parallax fundamental matrix and trifocal tensor have also been studied in [22,4]. The use of affine scaling $u_i \to 1$ for the centres $\mathtt{c}_i = \left(\begin{smallmatrix} \boldsymbol{c}_i \\ u_i \end{smallmatrix}\right)$ is essential here, otherwise $\boldsymbol{T}$ is bilinear and $\boldsymbol{Q}$ quadrilinear in $\boldsymbol{c}, u$.

Modulo scaling, $\boldsymbol{c}_{12}$ is the epipole $\boldsymbol{e}_{12}$ — the intersection of the ray from $\mathtt{c}_2$ to $\mathtt{c}_1$ with the reference plane. Coherent relative scaling of the terms of the trifocal and quadrifocal tensor sums is indispensable here, as in most other multi-term tensor relations. But for this very reason, the correct scales can be found using these relations. As discussed above, the correctly scaled $\boldsymbol{c}_{ij}$'s characterize the relative 3D camera geometry very explicitly, as a network of 3D displacement vectors. It is actually rather misleading to think in terms of epipolar points on the reference plane: the $\boldsymbol{c}_{ij}$ are neither estimated (*e.g.* from the trifocal tensor) nor used (*e.g.* for reconstruction) like that, and treating their scale factors as arbitrary only confuses the issue.

Matching constraints: The first few matching relations simplify as follows:

$$[\,\boldsymbol{x}_1, \boldsymbol{c}_{12}, \boldsymbol{x}_2\,] = 0 \qquad \text{epipolar point} \qquad (13)$$

$$(\boldsymbol{x}_1 \wedge \boldsymbol{x}_2)\,(\boldsymbol{c}_{13} \wedge \boldsymbol{x}_3)^\top - (\boldsymbol{c}_{12} \wedge \boldsymbol{x}_2)\,(\boldsymbol{x}_1 \wedge \boldsymbol{x}_3)^\top = \boldsymbol{0} \qquad \text{trifocal point} \qquad (14)$$

$$(\boldsymbol{l}_1 \wedge \boldsymbol{l}_2)\,(\boldsymbol{l}_3 \cdot \boldsymbol{c}_{13}) - (\boldsymbol{l}_2 \cdot \boldsymbol{c}_{12})\,(\boldsymbol{l}_1 \wedge \boldsymbol{l}_3) = \boldsymbol{0} \qquad \text{trifocal line} \qquad (15)$$

$$(\boldsymbol{l}_2 \cdot \boldsymbol{x}_1)\,(\boldsymbol{l}_3 \cdot \boldsymbol{c}_{13}) - (\boldsymbol{l}_2 \cdot \boldsymbol{c}_{12})\,(\boldsymbol{l}_3 \cdot \boldsymbol{x}_1) = 0 \qquad \text{trifocal point-line} \qquad (16)$$

$$\begin{aligned}(\boldsymbol{l}_2 \wedge \boldsymbol{l}_3)\,(\boldsymbol{l}_1 \cdot \boldsymbol{c}_{14}) + (\boldsymbol{l}_3 \wedge \boldsymbol{l}_1)\,(\boldsymbol{l}_2 \cdot \boldsymbol{c}_{24}) \\ + (\boldsymbol{l}_1 \wedge \boldsymbol{l}_2)\,(\boldsymbol{l}_3 \cdot \boldsymbol{c}_{34}) = \boldsymbol{0}\end{aligned} \qquad \text{quadrifocal 3-line} \qquad (17)$$

[2] There is no space here to display the general projective tensor analogues of the plane + parallax expressions given here and below — see [35].

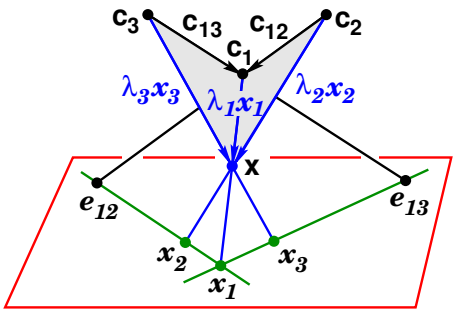


Figure 1: The geometry of the trifocal constraint.

Equation (16) is the primitive trifocal constraint. Given three images $\boldsymbol{x}_{i|i=1\ldots3}$ of a 3D point X, and arbitrary image lines $\boldsymbol{l}_2, \boldsymbol{l}_3$ through $\boldsymbol{x}_2, \boldsymbol{x}_3$, (16) asserts that the 3D optical ray of $\boldsymbol{x}_1$ meets the 3D optical planes of $\boldsymbol{l}_2, \boldsymbol{l}_3$ in a common 3D point (X). The tri- and quadrifocal 3-line constraints (16,17) both require that the optical planes of $\boldsymbol{l}_1, \boldsymbol{l}_2, \boldsymbol{l}_3$ intersect in a common 3D line. The quadrifocal 4-point constraint is straightforward but too long to give here.

The trifocal point constraint contains [29,35,22,4] two epipolar constraints in the form $\boldsymbol{x} \wedge \boldsymbol{x}' \simeq \boldsymbol{c} \wedge \boldsymbol{x}'$, plus a proportionality-of-scale relation for these parallel 3-vectors:

$$(\boldsymbol{x}_1 \wedge \boldsymbol{x}_2) \,:\, (\boldsymbol{c}_{12} \wedge \boldsymbol{x}_2) \;=\; (\boldsymbol{x}_1 \wedge \boldsymbol{x}_3) \,:\, (\boldsymbol{c}_{13} \wedge \boldsymbol{x}_3) \tag{18}$$

The homogeneous scale factors of the $\boldsymbol{x}$'s cancel. This equation essentially says that $\boldsymbol{x}_3$ must progress from $\boldsymbol{e}_{13}$ to $\boldsymbol{x}_1$ in step with $\boldsymbol{x}_2$ as it progresses from $\boldsymbol{e}_{12}$ to $\boldsymbol{x}_1$ (and both in step with X as it progresses from $\mathtt{C}_1$ to $\boldsymbol{x}_1$ on the plane — see fig. 1). In terms of 3D displacement vectors $\boldsymbol{c}$ and $\lambda\boldsymbol{x}$ (or if the figure is projected generically into another image), the ratio on the LHS of (18) is 1, being the ratio of two different methods of calculating the area of the triangle $\overline{\mathtt{C}_1\,\mathtt{C}_2\,\mathtt{X}}$. Similarly for the RHS with $\overline{\mathtt{C}_1\,\mathtt{C}_3\,\mathtt{X}}$. Both sides involve X, hence the lock-step.

Replacing the lines in the line constraints (15,16,17) with corresponding tangents to iso-intensity contours gives ***tensor brightness constraints*** on the normal flow at a point. The Hanna-Okamoto-Stein-Shashua brightness constraint (16) predominates for small, mostly-translational image displacements like residual parallaxes [7,31]. But for more general displacements, the 3 line constraints give additional information.

4 Redundancy, Scaling and Consistency

A major advantage of homography-epipole parametrizations is the extent to which they eliminate the redundancy that often makes the general tensor representation rather cumbersome. With plane + parallax against a fixed reference plane, the redundancy can be entirely eliminated. The aligned m camera geometry has $3m-4$ d.o.f.: the positions of the centres modulo an arbitrary choice of origin and a global scaling. These degrees of freedom are explicitly parametrized by, *e.g.*, the displacements $\boldsymbol{c}_{i1\,|\,i=2\ldots m}$, again modulo global rescaling. The remaining displacements can be found from $\boldsymbol{c}_{ij} = \boldsymbol{c}_i - \boldsymbol{c}_j = \boldsymbol{c}_{i1} - \boldsymbol{c}_{j1}$, and all of the matching tensors are simple linear functions of these. Conversely, the matching constraints are linear in the tensors and hence in the basic displacements $\boldsymbol{c}_{i1}$, so the complete vector of basic displacements *with the correct relative scaling* can be estimated linearly from image correspondences. These properties clearly simplify reconstruction. They are possible only because plane + parallax is a *local* representation — unlike the general, redundant tensor framework,

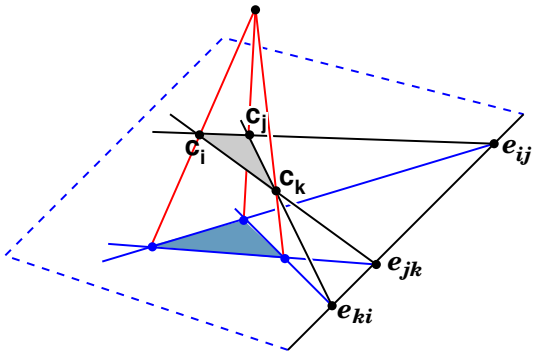


Figure 2: The various image projections of each triplet of 3D points and/or camera centres are in Desargues correspondence [22,4].

it becomes singular whenever a camera approaches the reference plane. However, the domain of validity is large enough for most real applications.

Consistency relations: As above, if they are parametrized by an independent set of inter-centre displacements, individual matching tensors in plane + parallax have no remaining internal consistency constraints and can be estimated linearly. The *inter*-tensor consistency constraints reduce to various more or less involved ways of enforcing the coincidence of versions of the same inter-camera displacement vector $\boldsymbol{c}_{ij}$ derived from different tensors, and the vanishing of cyclic sums of displacements:

$$\boldsymbol{c}_{ji} \wedge \boldsymbol{c}_{ij} \;=\; \boldsymbol{0} \qquad \boldsymbol{c}_{ij} \wedge (\boldsymbol{c}_{ij})' \;=\; \boldsymbol{0} \qquad \boldsymbol{c}_{ij} + \boldsymbol{c}_{jk} + \boldsymbol{c}_{kl} + \ldots + \boldsymbol{c}_{mi} \;=\; \boldsymbol{0}$$

In particular, each **cyclic triplet** of non-coincident epipoles is not only *aligned*, but has a *unique* consistent relative scaling $\boldsymbol{c}_{ij} \;\equiv\; \lambda_{ij}\, \boldsymbol{e}_{ij}$:

$$[\,\boldsymbol{e}_{ij}, \boldsymbol{e}_{jk}, \boldsymbol{e}_{ki}\,] \;=\; 0 \qquad \Longleftrightarrow \qquad \boldsymbol{c}_{ij} + \boldsymbol{c}_{jk} + \boldsymbol{c}_{ki} \;=\; \boldsymbol{0}$$

This and similar cyclic sums can be used to linearly recover the missing displacement scales. However, this fails if the 3D camera centres are aligned: the three epipoles coincide, so the vanishing of their cyclic sum still leaves 1 d.o.f. of relative scaling freedom. This corresponds to the well-known singularity of many fundamental matrix based reconstruction and transfer methods for aligned centres [40]. Trifocal or observation (depth recovery) based methods [32,37] must be used to recover the missing scale factors in this case.

The cyclic triplet relations essentially encode the coplanarity of triplets of optical centres. All three epipoles lie on the line of intersection of this plane with the reference plane. Also, the three images of any fourth point or camera centre form a Desargues theorem configuration with the three epipoles (see fig. 2). A multi-camera geometry induces multiple, intricately interlocking Desargues configurations — the reference plane 'signature' of its coherent 3D geometry.

5 Depth Recovery and Closure Relations

Closure relations: In the general projective case, the **closure relations** are the bilinear constraints between the (correctly scaled) matching tensors and the projection matrices, that express the fact that the former are functions of the latter [35,40]. **Closure based reconstruction** [38,40] uses this to

recover the projection matrices linearly from the matching tensors. In plane + parallax, the closure relations trivialize to identities of the form $\boldsymbol{c}_{ij} \wedge (\boldsymbol{c}_i - \boldsymbol{c}_j) = \boldsymbol{0}$ (since $\boldsymbol{c}_{ij} = \boldsymbol{c}_i - \boldsymbol{c}_j$). Closure based reconstruction just reads off a consistent set of $\boldsymbol{c}_i$'s from these linear constraints, with an arbitrary choice of origin and global scaling. $\boldsymbol{c}_i \equiv \boldsymbol{c}_{i1}$ is one such solution.

Depth recovery relations: Attaching the projection matrices in the closure relations to a 3D point gives **depth recovery relations** linking the matching tensors to correctly scaled image points [35, 32, 40]. These are used, *e.g.* for projective depth (scale factor) recovery in factorization based projective reconstruction [32, 37]. For plane + parallax registered points and lines with unknown relative scales, the first few depth recovery relations reduce to:

$$\boldsymbol{c}_{ij} \wedge (\lambda_i \boldsymbol{x}_i - \lambda_j \boldsymbol{x}_j) = \boldsymbol{0} \qquad \text{epipolar} \tag{19}$$

$$\boldsymbol{c}_{ij} \; (\lambda_k \boldsymbol{x}_k - \lambda_i \boldsymbol{x}_i)^\top - (\lambda_j \boldsymbol{x}_j - \lambda_i \boldsymbol{x}_i) \; (\boldsymbol{c}_{ik})^\top = \boldsymbol{0} \qquad \text{trifocal} \tag{20}$$

$$(\mu_i \boldsymbol{l}_i - \mu_j \boldsymbol{l}_j) \cdot \boldsymbol{c}_{ij} = 0 \qquad \text{line} \tag{21}$$

These follow immediately from the parallax equations (7,8). As before, the trifocal point relations contain two epipolar ones, plus an additional relative vector scaling proportionality: $(\lambda_i \boldsymbol{x}_i - \lambda_j \boldsymbol{x}_j) : \boldsymbol{c}_{ij} = (\lambda_i \boldsymbol{x}_i - \lambda_k \boldsymbol{x}_k) : \boldsymbol{c}_{ik}$. See fig. 1.

6 Reconstruction by Parallax Factorization

Now consider factorization based projective reconstruction under plane + parallax. Recall the general projective factorization reconstruction method [32, 37]: m cameras with 3×4 camera matrices $\mathsf{P}_{i \,|\, i=1\ldots m}$ view n 3D points $\mathsf{x}_{p \,|\, p=1\ldots n}$ to produce mn image points $\lambda_{ip} \boldsymbol{x}_{ip} = \mathsf{P}_i \mathsf{x}_p$. These projection equations can be gathered into a $3m \times n$ matrix:

$$\begin{pmatrix} \lambda_{11}\boldsymbol{x}_{11} & \cdots & \lambda_{1n}\boldsymbol{x}_{1n} \\ \vdots & \ddots & \vdots \\ \lambda_{m1}\boldsymbol{x}_{m1} & \cdots & \lambda_{mn}\boldsymbol{x}_{mn} \end{pmatrix} = \begin{pmatrix} \mathsf{P}_1 \\ \vdots \\ \mathsf{P}_m \end{pmatrix} \begin{pmatrix} \mathsf{x}_1 & \cdots & \mathsf{x}_n \end{pmatrix} \tag{22}$$

So the $(\lambda \boldsymbol{x})$ matrix factorizes into rank 4 factors. Any such factorization amounts to a projective reconstruction: the freedom is exactly a 4×4 projective change of coordinates H, with $\mathsf{x}_p \rightarrow \mathsf{H}\,\mathsf{x}_p$ and $\mathsf{P}_i \rightarrow \mathsf{P}_i\,\mathsf{H}^{-1}$. With noisy data the factorization is not exact, but we can use a numerical method such as truncated SVD to combine the measurements and estimate an approximate factorization and structure. To implement this with image measurements, we need to recover the unknown projective depths (scale factors) λ_{ip}. For this we use matching tensor based depth recovery relations such as $\boldsymbol{F}_{ij} \, (\lambda_{jp} \boldsymbol{x}_{jp}) = \boldsymbol{e}_{ji} \wedge (\lambda_{ip} \boldsymbol{x}_{ip})$ [35, 32, 37]. Rescaling the image points amounts to an implicit projective reconstruction, which the factorization consolidates and concretizes. For other factorization based SFM methods, see (among others) [34, 27, 18, 25, 26].

Plane + parallax point factorization: The general rank 4 method continues to work under plane + parallax with aligned points $\boldsymbol{x}_{ip}$, but in this case a more efficient rank 1 method exists, that exploits the special form of the aligned projection matrices:

1. Align the mn image points to the reference plane and (as for the general-case factorization) estimate their scale factors λ_{ip} by chaining together a network of plane + parallax depth recovery relations (19) or (20).

2. Choose a set of arbitrary weights ρ_i with $\sum_{i=1}^m \rho_i = 1$. We will work in a 3D frame based at the weighted average of the projection centres: *i.e.* $\bar{\boldsymbol{c}} = \sum_{i=1}^m \rho_i \boldsymbol{c}_i$ will be set to $\boldsymbol{0}$. For the experiments we work in an average-of-centres frame $\rho_i = \frac{1}{m}$. Alternatively, we could choose some image j as a base image, $\rho_i = \delta_{ij}$.

3. Calculate the weighted mean of the rescaled images of each 3D point, and their residual parallaxes relative to this in each image. The theoretical values are given for reference, based on (3) and our choice of frame $\bar{\boldsymbol{c}} \rightarrow \boldsymbol{0}$:

$$\bar{\boldsymbol{x}}_p \equiv \sum_{i=1}^m \rho_i \, (\lambda_{ip} \boldsymbol{x}_{ip}) \qquad \approx \qquad \boldsymbol{x}_p - w_p \bar{\boldsymbol{c}} \qquad \longrightarrow \qquad \boldsymbol{x}_p \tag{23}$$

$$\boldsymbol{\delta x}_{ip} \equiv \lambda_{ip} \boldsymbol{x}_{ip} - \bar{\boldsymbol{x}}_p \qquad \approx \qquad -(\boldsymbol{c}_i - \bar{\boldsymbol{c}}) \, w_p \qquad \longrightarrow \qquad -\boldsymbol{c}_i \, w_p \tag{24}$$

4. Factorize the combined residual parallax matrix to rank 1, to give the projection centres $\boldsymbol{c}_i$ and point depths w_p, with their correct relative scales:

$$\begin{pmatrix} \boldsymbol{\delta x}_{11} & \dots & \boldsymbol{\delta x}_{1n} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\delta x}_{m1} & \dots & \boldsymbol{\delta x}_{mn} \end{pmatrix} \approx \begin{pmatrix} -\boldsymbol{c}_1 \\ \vdots \\ -\boldsymbol{c}_m \end{pmatrix} \begin{pmatrix} w_1 & \dots & w_n \end{pmatrix} \tag{25}$$

   The ambiguity in the factorization is a single global scaling $\boldsymbol{c}_i \rightarrow \mu \boldsymbol{c}_i$, $w_p \rightarrow w_p/\mu$ (the length scale of the scene).

5. The final reconstructions are $\mathsf{P}_i = \begin{pmatrix} \boldsymbol{I} & -\boldsymbol{c}_i \end{pmatrix}$ and $\mathsf{x}_p = \begin{pmatrix} \bar{\boldsymbol{x}}_p \\ w_p \end{pmatrix}$.

This process requires the initial plane + parallax alignment, and estimates of the epipoles for projective depth recovery. It returns the 3D structure and camera centres in a projective frame that places the reference plane at infinity and the origin at the weighted average of camera centres.

With affine coordinates on the reference plane, the heights w_p reduce to inverse depths $1/z_p$ (w.r.t. the projectively distorted frame). Several existing factorization based SFM methods try to cancel the camera rotation and then factor the resulting translational motion into something like (inverse depth)·(translation), *e.g.* [12,25,21]. Owing to perspective effects, this is usually only achieved approximately, which leads to an iterative method. Here we require additional knowledge — a known, alignable reference plane and known epipoles for depth recovery — and we recover only projective structure, but this allows us to achieve exact results from perspective images with a single non-iterative rank 1 factorization. It would be interesting to investigate the relationships between our method and [25,26,21], but we have not yet done so.

Line Factorization: As in the general projective case, lines can be integrated into the point factorization method using via points. Each line is parametrized by choosing two arbitrary (but well-spaced) points on it in one image. The corresponding points on other images of the line are found by epipolar or trifocal point transfer, and the 3D via points are reconstructed using factorization. It turns out that the transfer process automatically gives the correct scale factor (depth) for the via points:

$$\boldsymbol{x}_i \equiv -\frac{\boldsymbol{l}_i \wedge (\boldsymbol{F}_{ij} \boldsymbol{x}_j)}{\boldsymbol{l}_i \cdot \boldsymbol{e}_{ji}} \quad \begin{matrix} \text{general} \\ \text{case} \end{matrix} \qquad \boldsymbol{x}_j \equiv \boldsymbol{x}_i + \frac{\boldsymbol{l}_j \cdot \boldsymbol{x}_i}{\boldsymbol{l}_j \cdot \boldsymbol{e}_{ij}} \boldsymbol{e}_{ij} \quad \begin{matrix} \text{plane + parallax} \\ \text{case} \end{matrix} \tag{26}$$

Under plane + parallax, all images $\boldsymbol{z} \wedge \boldsymbol{c}_i + \boldsymbol{l}$ of a line $(\boldsymbol{l}, \boldsymbol{z})$ intersect in a common point $\boldsymbol{z}$. If we estimate this first, only one additional via point is needed for the line.

Plane factorization: As mentioned in §2, inter-image homographies $\boldsymbol{H}_{i0}$ induced by 3D planes against a fixed base image 0 can also be incorporated in the above factorization, simply by treating their three columns as three separate point 3-vectors. Under plane + parallax, once they are scaled correctly as in §2, the homographies take the form (9). Averaging over i as above gives an $\bar{\boldsymbol{H}}_{i0}$ of the same form, with $\boldsymbol{c}_{i0}$ replaced by $\bar{\boldsymbol{c}} - \boldsymbol{c}_0 \;\rightarrow\; -\boldsymbol{c}_0$. So the corresponding "homography parallaxes" $\boldsymbol{\delta H}_{i0} = \frac{\boldsymbol{c}_i \boldsymbol{n}^\top}{\boldsymbol{n}\cdot\boldsymbol{c}_0+d}$ factor as for points, with $\frac{\boldsymbol{n}^\top}{\boldsymbol{n}\cdot\boldsymbol{c}_0+d}$ in place of w_p. Alternatively, if $\boldsymbol{c}_0$ is taken as origin and the $\boldsymbol{\delta}$'s are measured against image 0, $\boldsymbol{I}$ rather than $\bar{\boldsymbol{H}}_{i0}$ is subtracted.

Optimality properties: Ideally, we would like our structure and motion estimates to be optimal in some sense. For point estimators like maximum likelihood or MAP, this amounts to globally minimizing a measure of the (robustified, covariance-weighted) total squared image error, perhaps with overfitting penalties, *etc*. Unfortunately — as with all general closed-form projective SFM methods that we are aware of, and notwithstanding its excellent performance in practice — plane + parallax factorization uses an algebraically simple but statistically suboptimal error model. Little can be done about this, beyond using the method to initialize an iterative nonlinear refinement procedure (*e.g*. bundle adjustment). As in other estimation problems, it is safest to refine the results after each stage of the process, to ensure that the input to the next stage is as accurate and as outlier-free as possible. But even if the aligning homographies are refined in this way before being used (*c.f*. [9, 1, 11]), the projective centering and factorization steps are usually suboptimal because the projective rescaling $\lambda_{ip} \neq 1$ skews the statistical weighting of the input points. In more detail, by pre-weighting the image data matrix before factorization, affine factorization [34] can be generalized to give optimal results under an image error model as general as a per-image covariance times a per-3D-point weight[3]. But this is no longer optimal in projective factorization: even if the input errors are uniform, rescaling by the non-constant factors λ_{ip} distorts the underlying error model. In the plane + parallax case, the image rectification step further distorts the error model whenever there is non-negligible camera rotation. In spite of this, our experiments suggest that plane + parallax factorization gives near-optimal results in practice.

7 Experiments

Figure 3 compares the performance of the plane + parallax point factorization method described above, with conventional projective factorization using fundamental matrix depth recovery [32, 37], and also with projective bundle adjustment initialized from the plane + parallax solution. Cameras about 5 radii from the centre look inwards at a synthetic spherical point cloud cut by a reference plane. Half the points (but at least 4) lie on the plane, the rest are uniformly distributed in the sphere. The image size is 512×512, the focal length 1000 pixels. The cameras are uniformly spaced around a 90° arc centred on the origin. The default number of views is 4, points 20, Gaussian image noise 1 pixel. In the scene flatness experiment, the point cloud is progressively flattened onto the plane. The geometry is strong except under strong flattening and for small numbers of points.

The main conclusions are that plane + parallax factorization is somewhat more accurate than

[3] *I.e*. image point $\boldsymbol{x}_{ip}$ has covariance $\rho_p \boldsymbol{C}_i$, where $\boldsymbol{C}_i$ is a fixed covariance matrix for image i and ρ_p a fixed weight for 3D point p. Under this error model, factoring the weighted data matrix $(\rho_p^{-1/2} \boldsymbol{C}_i^{-1/2} \boldsymbol{x}_{ip})$ into weighted camera matrices $\boldsymbol{C}^{-1/2}\,\mathsf{P}_i$ and 3D point vectors $\rho_p^{-1/2}\,\mathsf{x}_p$ gives statistically optimal results. *Side note:* For typical images at least 90–95% of the image energy is in edge-like rather than corner-like structures ("the aperture problem"). So assuming that the (residual) camera rotations are small, an error model that permitted each 3D point to have its own highly anisotropic covariance matrix would usually be more appropriate than a per-image covariance. Irani & Anandan [20] go some way towards this by introducing an initial reduction based on a higher rank factorization of transposed weighted point vectors.

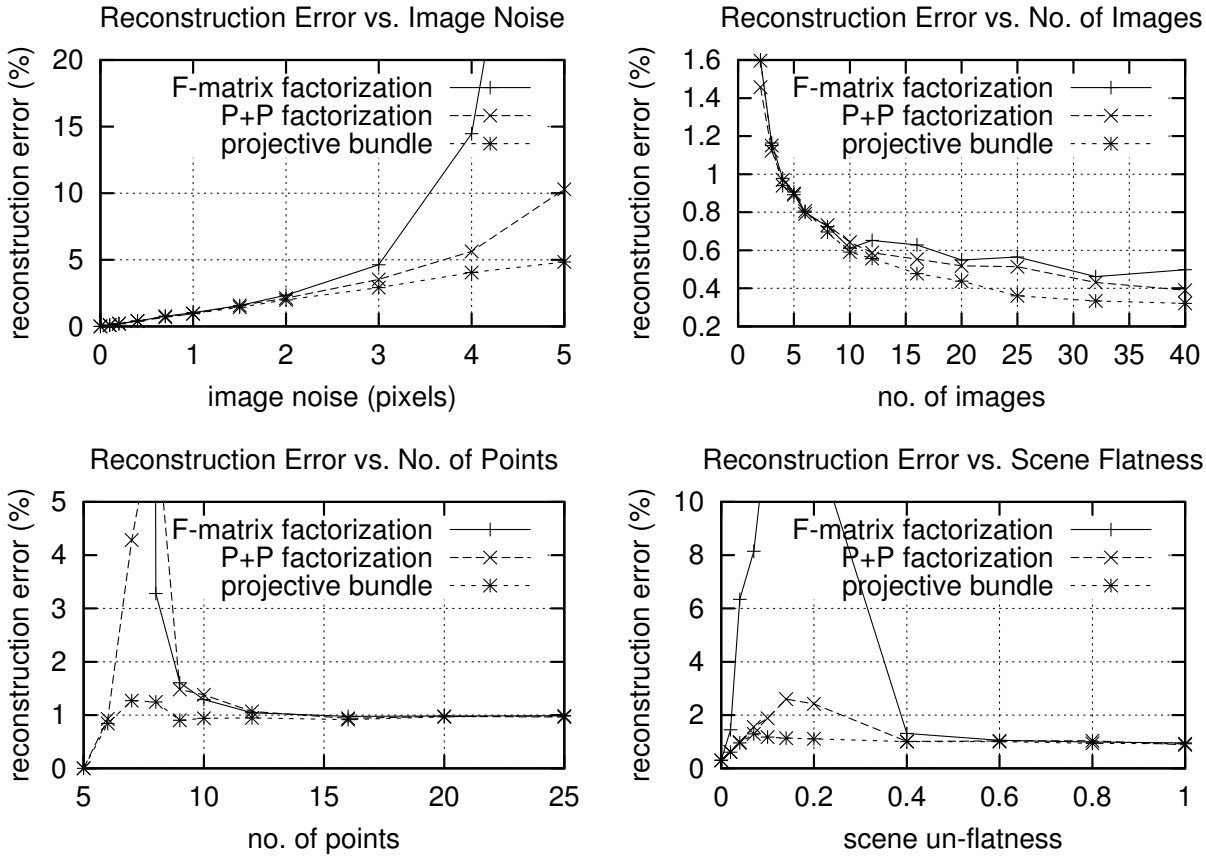


Figure 3: A comparison of 3D reconstruction errors for plane + parallax SFM factorization, fundamental matrix based projective factorization [32, 37], and projective bundle adjustment.

the standard fundamental matrix method, particularly for near planar scenes and linear fundamental matrix estimates, and often not far from optimal. In principle this was to be expected given that plane + parallax applies additional scene constraints (known coplanarity of some of the observed points). However, additional processing steps are involved (plane alignment, point centring), so it was not clear *a priori* how effectively the coplanarity constraints could be used. In fact, the two factorizations have very similar average reprojection errors in all the experiments reported here, which suggests that the additional processing introduces very little bias. The plane + parallax method's greater stability is confirmed by the fact that its factorization matrix is consistently a little better conditioned than that of the fundamental matrix method (*i.e*. the ratio of the smallest structure to the largest noise singular value is larger).

8 Summary

Plane + parallax alignment greatly simplifies multi-image projective geometry, reducing matching tensors and constraints, closure, depth recovery and inter-tensor consistency relations to fairly simple functions of the (correctly scaled!) epipoles. Choosing projective plane + parallax coordinates with

the reference plane at infinity helps this process by providing a (weak, projective) sense in which reference plane alignment cancels out precisely the camera rotation and calibration changes. This suggests a fruitful analogy with the case of translating calibrated cameras and a simple interpretation of plane + parallax geometry in terms of 3D displacement vectors.

The simplified parallax formula allows exact projective reconstruction by a simple rank-one (centre of projection)·(height) factorization. Like the general projective factorization method [32,37], an initial scale recovery step based on estimated epipoles is needed. When the required reference plane is available, the new method appears to perform at least as well as the general method, and significantly better in the case of near-planar scenes. Lines and homography matrices can be integrated into the point-based method, as in the general case.

Future work: We are still testing the plane + parallax factorization and refinements are possible. It would be interesting to relate it theoretically to affine factorization [34], and also to Oliensis's family of bias-corrected rotation-cancelling multiframe factorization methods [25,26]. Bias correction might be useful here too, although our centred data is probably less biased than the key frames of [25,26].

The analogy with translating cameras is open for exploration, and more generally, the idea of using a projective choice of 3D and image frames to get closer to a situation with a simple, special-case calibrated method, thus giving a simplified *projective* one. *E.g.* we find that suitable projective rectification of the images often makes affine factorization [34] much more accurate as a *projective* reconstruction method.

One can also consider autocalibration in the plane + parallax framework. It is easy to derive analogues of [41] (if only structure on the reference plane is used), or [16,39] (if the off-plane parallaxes are used as well). But so far this has not lead to any valuable simplifications or insights. Reference plane alignment distorts the camera calibrations, so the aligning homographies can not (immediately) be eliminated from the problem.

References

[1] D. Capel and A. Zisserman. Automated mosaicing with super-resolution zoom. In *Int. Conf. Computer Vision & Pattern Recognition*, pages 885–891, June 1998.

[2] S. Carlsson. Duality of reconstruction and positioning from projective views. In P. Anandan, editor, *IEEE Workshop on Representation of Visual Scenes*. IEEE Press, 1995.

[3] S. Carlsson and D. Weinshall. Dual computation of projective shape and camera positions from multiple images. *Int. J. Computer Vision*, 27(3):227–241, May 1998.

[4] A. Criminisi, I. Reid, and A. Zisserman. Duality, rigidity and planar parallax. In *European Conf. Computer Vision*, pages 846–861. Springer-Verlag, 1998.

[5] O. Faugeras and B. Mourrain. On the geometry and algebra of the point and line correspondences between n images. In *Int. Conf. Computer Vision*, pages 951–6, 1995.

[6] O. Faugeras and T. Papadopoulo. Grassmann-Cayley algebra for modeling systems of cameras and the algebraic equations of the manifold of trifocal tensors. *Transactions of the Royal Society A*, 1998.

[7] K. Hanna and N. Okamoto. Combining stereo and motion analysis for direct estimation of scene structure. In *Int. Conf. Computer Vision & Pattern Recognition*, pages 357–65, 1993.

[8] R. Hartley. Lines and points in three views and the trifocal tensor. *Int. J. Computer Vision*, 22(2):125–140, 1997.

[9] R. Hartley. Self calibration of stationary cameras. *Int. J. Computer Vision*, 22(1):5–23, 1997.

[10] R. Hartley and G. Debunne. Dualizing scene reconstruction algorithms. In R. Koch and L. Van Gool, editors, *Workshop on 3D Structure from Multiple Images of Large-scale Environments SMILE'98*, pages 14–31. Springer-Verlag, 1998.

[11] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[12] D. Heeger and A. Jepson. Subspace methods for recovering rigid motion I: Algorithm and implementation. *Int. J. Computer Vision*, 7:95–117, 1992.

[13] A. Heyden. Reconstruction from image sequences by means of relative depths. In E. Grimson, editor, *Int. Conf. Computer Vision*, pages 1058–63, Cambridge, MA, June 1995.

[14] A. Heyden and K. Åström. A canonical framework for sequences of images. In *IEEE Workshop on Representations of Visual Scenes*, Cambridge, MA, June 1995.

[15] A. Heyden and K. Åström. Algebraic varieties in multiple view geometry. In *European Conf. Computer Vision*, pages 671–682. Springer-Verlag, 1996.

[16] A. Heyden and K. Åström. Euclidean reconstruction from constant intrinsic parameters. In *Int. Conf. Pattern Recognition*, pages 339–43, Vienna, 1996.

[17] A. Heyden and K. Åström. Algebraic properties of multilinear constraints. *Mathematical Methods in the Applied Sciences*, 20:1135–1162, 1997.

[18] A. Heyden, R. Berthilsson, and G. Sparr. An iterative factorization method for projective structure and motion from image sequences. *Image & Vision Computing*, 17(5), 1999.

[19] M. Irani and P. Anadan. Parallax geometry of pairs of points for 3D scene analysis. In *European Conf. Computer Vision*, pages 17–30. Springer-Verlag, 1996.

[20] M. Irani and P. Anadan. Factorization with uncertainty. In *European Conf. Computer Vision*. Springer-Verlag, 2000.

[21] M. Irani, P. Anadan, and M. Cohen. Direct recovery of planar-parallax from multiple frames. In *Vision Algorithms: Theory and Practice*. Springer-Verlag, 2000.

[22] M. Irani, P. Anadan, and D. Weinshall. From reference frames to reference planes: Multi-view parallax geometry and applications. In *European Conf. Computer Vision*, pages 829–845. Springer-Verlag, 1998.

[23] M. Irani and P. Anandan. A unified approach to moving object detection in 2D and 3D scenes. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 20(6):577–589, June 1998.

[24] R. Kumar, P. Anandan, M. Irani, J. Bergen, and K. Hanna. Representation of scenes from collections of images. In *IEEE Workshop on Representations of Visual Scenes*, pages 10–17, June 1995.

[25] J. Oliensis. Multiframe structure from motion in perspective. In *IEEE Workshop on Representation of Visual Scenes*, pages 77–84, June 1995.

[26] J. Oliensis and Y. Genc. Fast algorithms for projective multi-frame structure from motion. In *Int. Conf. Computer Vision*, pages 536–542, Corfu, Greece, 1999.

[27] C.J. Poelman and T. Kanade. A parapersective factorization method for shape and motion recovery. In *European Conf. Computer Vision*, pages 97–108, Stockholm, 1994. Springer-Verlag.

[28] A. Shashua and S. Avidan. The rank 4 constraint in multiple ($\geq$ 3) view geometry. In *European Conf. Computer Vision*, pages 196–206, Cambridge, 1996.

[29] A. Shashua and M. Werman. On the trilinear tensor of three perspective views and its underlying geometry. In *Int. Conf. Computer Vision*, Boston, MA, June 1995.

[30] C.E. Springer. *Geometry and Analysis of Projective Spaces*. Freeman, 1964.

[31] G. Stein and A. Shashua. Model-based brightness constraints: On direct estimation of structure and motion. In *Int. Conf. Computer Vision & Pattern Recognition*, pages 400–406, 1997.

[32] P. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *European Conf. Computer Vision*, pages 709–20, Cambridge, U.K., 1996. Springer-Verlag.

[33] R. Szeliski and S.B. Kang. Direct methods for visual scene reconstruction. In *IEEE Workshop on Representation of Visual Scenes*, pages 26–33, June 1995.

[34] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *Int. J. Computer Vision*, 9(2):137–54, 1992.

[35] B. Triggs. The geometry of projective reconstruction I: Matching constraints and the joint image. Unpublished. (Submitted to *IJCV* in 1995).

[36] B. Triggs. Matching constraints and the joint image. In E. Grimson, editor, *Int. Conf. Computer Vision*, pages 338–43, Cambridge, MA, June 1995.

[37] B. Triggs. Factorization methods for projective structure and motion. In *Int. Conf. Computer Vision & Pattern Recognition*, pages 845–51, San Francisco, CA, 1996.

[38] B. Triggs. Linear projective reconstruction from matching tensors. In *British Machine Vision Conference*, pages 665–74, Edinburgh, September 1996.

[39] B. Triggs. Autocalibration and the absolute quadric. In *Int. Conf. Computer Vision & Pattern Recognition*, Puerto Rico, 1997.

[40] B. Triggs. Linear projective reconstruction from matching tensors. *Image & Vision Computing*, 15(8):617–26, August 1997.

[41] B. Triggs. Autocalibration from planar scenes. In *European Conf. Computer Vision*, pages I 89–105, Freiburg, June 1998.

[42] D. Weinshall, P. Anandan, and M. Irani. From ordinal to euclidean reconstruction with partial scene calibration. In R. Koch and L. Van Gool, editors, *3D Structure from Multiple Images of Large-scale Environments SMILE'98*, pages 208–223. Springer-Verlag, 1998.

[43] D. Weinshall, M. Werman, and A. Shashua. Shape tensors for efficient and learnable indexing. In *IEEE Workshop on Representation of Visual Scenes*, pages 58–65, June 1995.

[44] L. Zelnik-Manor and M. Irani. Multi-frame alignment of planes. In *Int. Conf. Computer Vision & Pattern Recognition*, pages 151–156, 1999.

[45] L. Zelnik-Manor and M. Irani. Multi-view subspace constraints on homographies. In *Int. Conf. Computer Vision*, pages 710–715, 1999.

Bundle Adjustment — A Modern Synthesis

Bill Triggs[1], Philip McLauchlan[2], Richard Hartley[3] and Andrew Fitzgibbon[4]

[1] INRIA Rhône-Alpes, 655 avenue de l'Europe, 38330 Montbonnot, France.
Bill.Triggs@inrialpes.fr ◇ *http://www.inrialpes.fr/movi/people/Triggs*

[2] School of Electrical Engineering, Information Technology & Mathematics
University of Surrey, Guildford, GU2 5XH, U.K.
P.McLauchlan@ee.surrey.ac.uk ◇ *http://www.ee.surrey.ac.uk/Personal/P.McLauchlan*

[3] General Electric CRD, Schenectady, NY, 12301
hartley@crd.ge.com

[4] Dept of Engineering Science, University of Oxford, 19 Parks Road, OX1 3PJ, U.K.
awf@robots.ox.ac.uk ◇ *http://www.robots.ox.ac.uk/ awf*

Abstract

This paper is a survey of the theory and methods of photogrammetric bundle adjustment, aimed at potential implementors in the computer vision community. Bundle adjustment is the problem of refining a visual reconstruction to produce jointly optimal structure and viewing parameter estimates. Topics covered include: the choice of cost function and robustness; numerical optimization including sparse Newton methods, linearly convergent approximations, updating and recursive methods; gauge (datum) invariance; and quality control. The theory is developed for general robust cost functions rather than restricting attention to traditional nonlinear least squares.

Keywords: Bundle Adjustment, Scene Reconstruction, Gauge Freedom, Sparse Matrices, Optimization.

1 Introduction

This paper is a survey of the theory and methods of bundle adjustment aimed at the computer vision community, and more especially at potential implementors who already know a little about bundle methods. Most of the results appeared long ago in the photogrammetry and geodesy literatures, but many seem to be little known in vision, where they are gradually being reinvented. By providing an accessible modern synthesis, we hope to forestall some of this duplication of effort, correct some common misconceptions, and speed progress in visual reconstruction by promoting interaction between the vision and photogrammetry communities.

This work was supported in part by the European Commission Esprit LTR project CUMULI (B. Triggs), the UK EPSRC project GR/L34099 (P. McLauchlan), and the Royal Society (A. Fitzgibbon). We would like to thank A. Zisserman, A. Grün and W. Förstner for valuable comments and references.
Appeared in *Vision Algorithms: Theory & Practice*, B. Triggs, A. Zisserman & R. Szeliski (Eds.), Springer-Verlag LNCS 1883, 2000.

Bundle adjustment is the problem of refining a visual reconstruction to produce *jointly optimal* 3D structure and viewing parameter (camera pose and/or calibration) estimates. *Optimal* means that the parameter estimates are found by minimizing some cost function that quantifies the model fitting error, and *jointly* that the solution is simultaneously optimal with respect to both structure and camera variations. The name refers to the 'bundles' of light rays leaving each 3D feature and converging on each camera centre, which are 'adjusted' optimally with respect to both feature and camera positions. Equivalently — unlike *independent model methods*, which merge partial reconstructions without updating their internal structure — all of the structure and camera parameters are adjusted together 'in one bundle'.

Bundle adjustment is really just a large sparse geometric parameter estimation problem, the parameters being the combined 3D feature coordinates, camera poses and calibrations. Almost everything that we will say can be applied to many similar estimation problems in vision, photogrammetry, industrial metrology, surveying and geodesy. Adjustment computations are a major common theme throughout the measurement sciences, and once the basic theory and methods are understood, they are easy to adapt to a wide variety of problems. Adaptation is largely a matter of choosing a numerical optimization scheme that exploits the problem structure and sparsity. We will consider several such schemes below for bundle adjustment.

Classically, bundle adjustment and similar adjustment computations are formulated as nonlinear least squares problems [19,46,100,21,22,69,5,73,109]. The cost function is assumed to be quadratic in the feature reprojection errors, and robustness is provided by explicit outlier screening. Although it is already very flexible, this model is not really general enough. Modern systems often use non-quadratic M-estimator-like distributional models to handle outliers more integrally, and many include additional penalties related to overfitting, model selection and system performance (priors, MDL). For this reason, we will *not* assume a least squares / quadratic cost model. Instead, the cost will be modelled as a sum of opaque contributions from the independent information sources (individual observations, prior distributions, overfitting penalties...). The functional forms of these contributions and their dependence on fixed quantities such as observations will usually be left implicit. This allows many different types of robust and non-robust cost contributions to be incorporated, without unduly cluttering the notation or hiding essential model structure. It fits well with modern sparse optimization methods (cost contributions are usually sparse functions of the parameters) and object-centred software organization, and it avoids many tedious displays of chain-rule results. Implementors are assumed to be capable of choosing appropriate functions and calculating derivatives themselves.

One aim of this paper is to correct a number of misconceptions that seem to be common in the vision literature:

- ***"Optimization / bundle adjustment is slow"*:** Such statements often appear in papers introducing yet another heuristic Structure from Motion (SFM) iteration. The claimed slowness is almost always due to the unthinking use of a general-purpose optimization routine that completely ignores the problem structure and sparseness. Real bundle routines are *much* more efficient than this, and usually considerably more efficient and flexible than the newly suggested method (§6, 7). That is why bundle adjustment remains the dominant structure refinement technique for real applications, after 40 years of research.
- ***"Only linear algebra is required"*:** This is a recent variant of the above, presumably meant to imply that the new technique is especially simple. Virtually all iterative refinement techniques use only linear algebra, and bundle adjustment is simpler than many in that it only solves linear systems: it makes no use of eigen-decomposition or SVD, which are themselves complex iterative methods.

- ***"Any sequence can be used"*:** Many vision workers seem to be very resistant to the idea that reconstruction problems should be planned in advance (§11), and results checked afterwards to verify their reliability (§10). System builders should at least be aware of the basic techniques for this, even if application constraints make it difficult to use them. The extraordinary extent to which weak geometry and lack of redundancy can mask gross errors is too seldom appreciated, *c.f.* [34,50,30,33].

- ***"Point P is reconstructed accurately"*:** In reconstruction, just as there are no absolute references for position, there are none for uncertainty. The 3D coordinate frame is itself uncertain, as it can only be located relative to uncertain reconstructed features or cameras. All other feature and camera uncertainties are expressed relative to the frame and inherit its uncertainty, so statements about them are meaningless until the frame and its uncertainty are specified. Covariances can look completely different in different frames, particularly in object-centred versus camera-centred ones. See §9.

There is a tendency in vision to develop a profusion of *ad hoc* adjustment iterations. Why should you use bundle adjustment rather than one of these methods? :

- **Flexibility:** Bundle adjustment gracefully handles a very wide variety of different 3D feature and camera types (points, lines, curves, surfaces, exotic cameras), scene types (including dynamic and articulated models, scene constraints), information sources (2D features, intensities, 3D information, priors) and error models (including robust ones). It has no problems with missing data.

- **Accuracy:** Bundle adjustment gives precise and easily interpreted results because it uses accurate statistical error models and supports a sound, well-developed quality control methodology.

- **Efficiency:** Mature bundle algorithms are comparatively efficient even on very large problems. They use economical and rapidly convergent numerical methods and make near-optimal use of problem sparseness.

In general, as computer vision reconstruction technology matures, we expect that bundle adjustment will predominate over alternative adjustment methods in much the same way as it has in photogrammetry. We see this as an inevitable consequence of a greater appreciation of optimization (notably, more effective use of problem structure and sparseness), and of systems issues such as quality control and network design.

Coverage: We will touch on a good many aspects of bundle methods. We start by considering the camera projection model and the parametrization of the bundle problem §2, and the choice of error metric or cost function §3. §4 gives a rapid sketch of the optimization theory we will use. §5 discusses the network structure (parameter interactions and characteristic sparseness) of the bundle problem. The following three sections consider three types of implementation strategies for adjustment computations: §6 covers second order Newton-like methods, which are still the most often used adjustment algorithms; §7 covers methods with only first order convergence (most of the *ad hoc* methods are in this class); and §8 discusses solution updating strategies and recursive filtering bundle methods. §9 returns to the theoretical issue of gauge freedom (datum deficiency), including the theory of inner constraints. §10 goes into some detail on quality control methods for monitoring the accuracy and reliability of the parameter estimates. §11 gives some brief hints on network design, *i.e.* how to place your shots to ensure accurate, reliable reconstruction. §12 completes the body of the paper by summarizing the main conclusions and giving some provisional recommendations for methods. There are also several appendices. §A gives a brief historical overview of the development of bundle methods, with literature references. §B gives some technical details of matrix factorization,

updating and covariance calculation methods. §C gives some hints on designing bundle software, and pointers to useful resources on the Internet. The paper ends with a glossary and references.

General references: Cultural differences sometimes make it difficult for vision workers to read the photogrammetry literature. The collection edited by Atkinson [5] and the manual by Karara [69] are both relatively accessible introductions to close-range (rather than aerial) photogrammetry. Other accessible tutorial papers include [46,21,22]. Kraus [73] is probably the most widely used photogrammetry textbook. Brown's early survey of bundle methods [19] is well worth reading. The often-cited manual edited by Slama [100] is now quite dated, although its presentation of bundle adjustment is still relevant. Wolf & Ghiliani [109] is a text devoted to adjustment computations, with an emphasis on surveying. Hartley & Zisserman [62] is an excellent recent textbook covering vision geometry from a computer vision viewpoint. For nonlinear optimization, Fletcher [29] and Gill *et al* [42] are the traditional texts, and Nocedal & Wright [93] is a good modern introduction. For linear least squares, Björck [11] is superlative, and Lawson & Hanson is a good older text. For more general numerical linear algebra, Golub & Van Loan [44] is the standard. Duff *et al* [26] and George & Liu [40] are the standard texts on sparse matrix techniques. We will not discuss initialization methods for bundle adjustment in detail, but appropriate reconstruction methods are plentiful and well-known in the vision community. See, *e.g.*, [62] for references.

Notation: The structure, cameras, *etc.*, being estimated will be parametrized by a single large **state vector** x. In general the state belongs to a nonlinear manifold, but we linearize this locally and work with small linear state displacements denoted $\delta\mathsf{x}$. Observations (*e.g.* measured image features) are denoted $\underline{\mathsf{z}}$. The corresponding predicted values at parameter value x are denoted $\mathsf{z} = \mathsf{z}(\mathsf{x})$, with **residual prediction error** $\triangle\mathsf{z}(\mathsf{x}) \equiv \underline{\mathsf{z}} - \mathsf{z}(\mathsf{x})$. However, observations and prediction errors usually only appear implicitly, through their influence on the **cost function** $\mathsf{f}(\mathsf{x}) = \mathsf{f}(predz(\mathsf{x}))$. The cost function's **gradient** is $\mathsf{g} \equiv \frac{\mathsf{df}}{\mathsf{dx}}$, and its **Hessian** is $\mathsf{H} \equiv \frac{\mathsf{d}^2\mathsf{f}}{\mathsf{dx}^2}$. The **observation-state Jacobian** is $\mathsf{J} \equiv \frac{\mathsf{dz}}{\mathsf{dx}}$. The dimensions of $\delta\mathsf{x}, \delta\mathsf{z}$ are $n_\mathsf{x}, n_\mathsf{z}$.

2 Projection Model and Problem Parametrization

2.1 The Projection Model

We begin the development of bundle adjustment by considering the basic image projection model and the issue of problem parametrization. Visual reconstruction attempts to recover a model of a 3D scene from multiple images. As part of this, it usually also recovers the poses (positions and orientations) of the cameras that took the images, and information about their internal parameters. A simple scene model might be a collection of isolated 3D features, *e.g.*, points, lines, planes, curves, or surface patches. However, far more complicated scene models are possible, involving, *e.g.*, complex objects linked by constraints or articulations, photometry as well as geometry, dynamics, *etc*. One of the great strengths of adjustment computations — and one reason for thinking that they have a considerable future in vision — is their ability to take such complex and heterogeneous models in their stride. Almost any *predictive parametric* model can be handled, *i.e.* any model that *predicts* the values of some known measurements or descriptors on the basis of some continuous *parametric* representation of the world, which is to be estimated from the measurements.

Similarly, many possible camera models exist. Perspective projection is the standard, but the affine and orthographic projections are sometimes useful for distant cameras, and more exotic models such as push-broom and rational polynomial cameras are needed for certain applications [56,63]. In addition to pose (position and orientation), and simple internal parameters such as focal length and

principal point, real cameras also require various types of **additional parameters** to model internal aberrations such as radial distortion [17, 18, 19, 100, 69, 5].

For simplicity, suppose that the scene is modelled by individual static 3D features $\mathbf{X}_p$, $p = 1 \ldots n$, imaged in m shots with camera pose and internal calibration parameters $\mathbf{P}_i$, $i = 1 \ldots m$. There may also be further calibration parameters $\mathbf{C}_c$, $c = 1 \ldots k$, constant across several images (*e.g.*, depending on which of several cameras was used). We are given uncertain measurements $\underline{\boldsymbol{x}}_{ip}$ of some subset of the possible image features $\boldsymbol{x}_{ip}$ (the true image of feature $\mathbf{X}_p$ in image i). For each observation $\underline{\boldsymbol{x}}_{ip}$, we assume that we have a **predictive model** $\boldsymbol{x}_{ip} = \boldsymbol{x}(\mathbf{C}_c, \mathbf{P}_i, \mathbf{X}_p)$ based on the parameters, that can be used to derive a **feature prediction error**:

$$\triangle \boldsymbol{x}_{ip}(\mathbf{C}_c, \mathbf{P}_i, \mathbf{X}_p) \;\equiv\; \underline{\boldsymbol{x}}_{ip} - \boldsymbol{x}(\mathbf{C}_c, \mathbf{P}_i, \mathbf{X}_p) \tag{1}$$

In the case of image observations the predictive model is image projection, but other observation types such as 3D measurements can also be included.

To estimate the unknown 3D feature and camera parameters from the observations, and hence reconstruct the scene, we minimize some measure (discussed in §3) of their total prediction error. Bundle adjustment is the model refinement part of this, starting from given initial parameter estimates (*e.g.*, from some approximate reconstruction method). Hence, it is essentially a matter of optimizing a complicated nonlinear cost function (the total prediction error) over a large nonlinear parameter space (the scene and camera parameters).

We will not go into the analytical forms of the various possible feature and image projection models, as these do not affect the general structure of the adjustment network, and only tend to obscure its central simplicity. We simply stress that the bundle framework is flexible enough to handle almost any desired model. Indeed, there are so many different combinations of features, image projections and measurements, that it is best to regard them as black boxes, capable of giving measurement predictions based on their current parameters. (For optimization, first, and possibly second, derivatives with respect to the parameters are also needed).

For much of the paper we will take quite an abstract view of this situation, collecting the scene and camera parameters to be estimated into a large **state vector** x, and representing the cost (total fitting error) as an abstract function $\mathsf{f}(\mathsf{x})$. The cost is really a function of the feature prediction errors $\triangle \boldsymbol{x}_{ip} = \underline{\boldsymbol{x}}_{ip} - \boldsymbol{x}(\mathbf{C}_c, \mathbf{P}_i, \mathbf{X}_p)$. But as the observations $\underline{\boldsymbol{x}}_{ip}$ are constants during an adjustment calculation, we leave the cost's dependence on them and on the projection model $\boldsymbol{x}(\cdot)$ implicit, and display only its dependence on the parameters x actually being adjusted.

2.2 Bundle Parametrization

The bundle adjustment parameter space is generally a high-dimensional nonlinear manifold — a large Cartesian product of projective 3D feature, 3D rotation, and camera calibration manifolds, perhaps with nonlinear constraints, *etc*. The state x is not strictly speaking a vector, but rather a point in this space. Depending on how the entities that it contains are represented, x can be subject to various types of complications including singularities, internal constraints, and unwanted internal degrees of freedom. These arise because geometric entities like rotations, 3D lines and even projective points and planes, do not have simple global parametrizations. Their local parametrizations are nonlinear, with singularities that prevent them from covering the whole parameter space uniformly (*e.g.* the many variants on Euler angles for rotations, the singularity of affine point coordinates at infinity). And their global parametrizations either have constraints (*e.g.* quaternions with $\|\boldsymbol{q}\|^2 = 1$), or unwanted internal degrees of freedom (*e.g.* homogeneous projective quantities have a scale factor freedom, two points

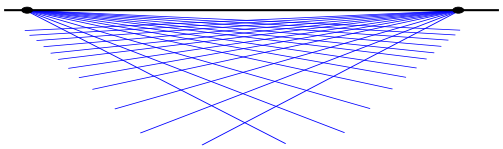


Figure 1: Vision geometry and its error model are essentially projective. Affine parametrization introduces an artificial singularity at projective infinity, which may cause numerical problems for distant features.

defining a line can slide along the line). For more complicated compound entities such as matching tensors and assemblies of 3D features linked by coincidence, parallelism or orthogonality constraints, parametrization becomes even more delicate.

Although they are in principle equivalent, different parametrizations often have profoundly different numerical behaviours which greatly affect the speed and reliability of the adjustment iteration. The most suitable parametrizations for optimization are as uniform, finite and well-behaved as possible *near the current state estimate*. Ideally, they should be locally close to linear in terms of their effect on the chosen error model, so that the cost function is locally nearly quadratic. Nonlinearity hinders convergence by reducing the accuracy of the second order cost model used to predict state updates (§6). Excessive correlations and parametrization singularities cause ill-conditioning and erratic numerical behaviour. Large or infinite parameter values can only be reached after excessively many finite adjustment steps.

Any given parametrization will usually only be well-behaved in this sense over a relatively small section of state space. So to guarantee uniformly good performance, however the state itself may be represented, *state updates should be evaluated using a stable* ***local*** *parametrization based on increments from the current estimate*. As examples we consider 3D points and rotations.

3D points: Even for calibrated cameras, vision geometry and visual reconstructions are intrinsically projective. If a 3D $(X\ Y\ Z)^\top$ parametrization (or equivalently a homogeneous affine $(X\ Y\ Z\ 1)^\top$ one) is used for very distant 3D points, large X, Y, Z displacements are needed to change the image significantly. *I.e.*, in $(X\ Y\ Z)$ space the cost function becomes very flat and steps needed for cost adjustment become very large for distant points. In comparison, with a homogeneous projective parametrization $(X\ Y\ Z\ W)^\top$, the behaviour near infinity is natural, finite and well-conditioned so long as the normalization keeps the homogeneous 4-vector finite at infinity (by sending $W \to 0$ there). In fact, there is no immediate visual distinction between the images of real points near infinity and virtual ones 'beyond' it (all camera geometries admit such virtual points as *bona fide* projective constructs). The optimal reconstruction of a real 3D point may even be virtual in this sense, if image noise happens to push it 'across infinity'. Also, there is nothing to stop a reconstructed point wandering beyond infinity and back during the optimization. This sounds bizarre at first, but it is an inescapable consequence of the fact that the natural geometry and error model for visual reconstruction is projective rather than affine. Projectively, *infinity is just like any other place*. Affine parametrization $(X\ Y\ Z\ 1)^\top$ is acceptable for points near the origin with close-range convergent camera geometries, but it is disastrous for distant ones because it artificially cuts away half of the natural parameter space, and hides the fact by sending the resulting edge to infinite parameter values. Instead, you should use a homogeneous parametrization $(X\ Y\ Z\ W)^\top$ for distant points, *e.g.* with spherical normalization $\sum X_i^2 = 1$.

Rotations: Similarly, experience suggests that quasi-global 3 parameter rotation parametrizations such as Euler angles cause numerical problems unless one can be certain to avoid their singularities and regions of uneven coverage. Rotations should be parametrized using either quaternions subject to $\|\boldsymbol{q}\|^2 = 1$, or local perturbations $\boldsymbol{R}\,\boldsymbol{\delta R}$ or $\boldsymbol{\delta R}\,\boldsymbol{R}$ of an existing rotation R, where $\boldsymbol{\delta R}$ can be any well-behaved 3 parameter small rotation approximation, *e.g.* $\boldsymbol{\delta R} = (\boldsymbol{I} + [\,\boldsymbol{\delta r}\,]_\times)$, the Rodriguez formula,

local Euler angles, *etc*.

State updates: Just as state vectors x represent points in some nonlinear space, state updates $\mathsf{x} \rightarrow \mathsf{x} + \boldsymbol{\delta}\mathsf{x}$ represent displacements in this nonlinear space that often can not be represented exactly by vector addition. Nevertheless, we assume that we can locally linearize the state manifold, locally resolving any internal constraints and freedoms that it may be subject to, to produce an unconstrained vector $\boldsymbol{\delta}\mathsf{x}$ parametrizing the possible local state displacements. We can then, *e.g.*, use Taylor expansion in $\boldsymbol{\delta}\mathsf{x}$ to form a local cost model $\mathsf{f}(\mathsf{x} + \boldsymbol{\delta}\mathsf{x}) \approx \mathsf{f}(\mathsf{x}) + \frac{\mathsf{df}}{\mathsf{dx}}\,\boldsymbol{\delta}\mathsf{x} + \frac{1}{2}\boldsymbol{\delta}\mathsf{x}^\top \frac{\mathsf{d}^2\mathsf{f}}{\mathsf{dx}^2}\,\boldsymbol{\delta}\mathsf{x}$, from which we can estimate the state update $\boldsymbol{\delta}\mathsf{x}$ that optimizes this model (§4). The displacement $\boldsymbol{\delta}\mathsf{x}$ need not have the same structure or representation as x — indeed, if a well-behaved local parametrization is used to represent $\boldsymbol{\delta}\mathsf{x}$, it generally will not have — but we must at least be able to update the state with the displacement to produce a new state estimate. We write this operation as $\mathsf{x} \rightarrow \mathsf{x} + \boldsymbol{\delta}\mathsf{x}$, even though it may involve considerably more than vector addition. For example, apart from the change of representation, an updated quaternion $\boldsymbol{q} \rightarrow \boldsymbol{q} + \boldsymbol{dq}$ will need to have its normalization $\|\boldsymbol{q}\|^2 = 1$ corrected, and a small rotation update of the form $\boldsymbol{R} \rightarrow \boldsymbol{R}(\mathsf{1} + [\,r\,]_\times)$ will not in general give an exact rotation matrix.

3 Error Modelling

We now turn to the choice of the cost function $\mathsf{f}(\mathsf{x})$, which quantifies the total prediction (image reprojection) error of the model parametrized by the combined scene and camera parameters x. Our main conclusion will be that robust statistically-based error metrics based on total (inlier + outlier) log likelihoods should be used, to correctly allow for the presence of outliers. We will argue this at some length as it seems to be poorly understood. The traditional treatments of adjustment methods consider only least squares (albeit with data trimming for robustness), and most discussions of robust statistics give the impression that the choice of robustifier or M-estimator is wholly a matter of personal whim rather than data statistics.

Bundle adjustment is essentially a parameter estimation problem. Any parameter estimation paradigm could be used, but we will consider only **optimal point estimators**, whose output is by definition the single parameter vector that minimizes a predefined **cost function** designed to measure how well the model fits the observations and background knowledge. This framework covers many practical estimators including maximum likelihood (ML) and maximum a posteriori (MAP), but not explicit Bayesian model averaging. Robustification, regularization and model selection terms are easily incorporated in the cost.

A typical ML cost function would be the summed negative log likelihoods of the prediction errors of all the observed image features. For Gaussian error distributions, this reduces to the sum of squared covariance-weighted prediction errors (§3.2). A MAP estimator would typically add cost terms giving certain structure or camera calibration parameters a bias towards their expected values.

The cost function is also a tool for statistical interpretation. To the extent that lower costs are uniformly 'better', it provides a natural model preference ordering, so that cost iso-surfaces above the minimum define natural confidence regions. Locally, these regions are nested ellipsoids centred on the cost minimum, with size and shape characterized by the **dispersion matrix** (the inverse of the cost function Hessian $\mathsf{H} = \frac{\mathsf{d}^2\mathsf{f}}{\mathsf{dx}^2}$ at the minimum). Also, the residual cost at the minimum can be used as a test statistic for model validity (§10). *E.g.*, for a negative log likelihood cost model with Gaussian error distributions, twice the residual is a χ^2 variable.

3.1 Desiderata for the Cost Function

In adjustment computations we go to considerable lengths to optimize a large nonlinear cost model, so it seems reasonable to require that the refinement should actually improve the estimates in some objective (albeit statistical) sense. Heuristically motivated cost functions can not usually guarantee this. They almost always lead to biased parameter estimates, and often severely biased ones. A large body of statistical theory points to maximum likelihood (ML) and its Bayesian cousin maximum a posteriori (MAP) as the estimators of choice. ML simply selects the model for which the total probability of the observed data is highest, or saying the same thing in different words, for which the *total posterior probability* of the model given the observations is highest. MAP adds a prior term representing background information. ML could just as easily have included the prior as an additional 'observation': so far as estimation is concerned, the distinction between ML / MAP and prior / observation is purely terminological.

Information usually comes from many independent sources. In bundle adjustment these include: covariance-weighted reprojection errors of individual image features; other measurements such as 3D positions of control points, GPS or inertial sensor readings; predictions from uncertain dynamical models (for 'Kalman filtering' of dynamic cameras or scenes); prior knowledge expressed as soft constraints (*e.g.* on camera calibration or pose values); and supplementary sources such as overfitting, regularization or description length penalties. Note the variety. One of the great strengths of adjustment computations is their ability to combine information from disparate sources. Assuming that the sources are statistically independent of one another given the model, the total probability for the model given the combined data is the product of the probabilities from the individual sources. To get an additive cost function we take logs, so the total log likelihood for the model given the combined data is the sum of the individual source log likelihoods.

Properties of ML estimators: Apart from their obvious simplicity and intuitive appeal, ML and MAP estimators have strong statistical properties. Many of the most notable ones are **asymptotic**, *i.e.* they apply in the limit of a large number of independent measurements, or more precisely in the **central limit** where the posterior distribution becomes effectively Gaussian[1]. In particular:

- Under mild regularity conditions on the observation distributions, the posterior distribution of the ML estimate converges asymptotically in probability to a Gaussian with covariance equal to the dispersion matrix.
- The ML estimate asymptotically has zero bias and the lowest variance that any unbiased estimator can have. So in this sense, ML estimation is at least as good as any other method[2].

[1] Cost is additive, so as measurements of the same type are added the entire cost surface grows in direct proportion to the amount of data n_{z}. This means that the *relative* sizes of the cost and all of its derivatives — and hence the size r of the region around the minimum over which the second order Taylor terms dominate all higher order ones — remain roughly constant as n_{z} increases. Within this region, the total cost is roughly quadratic, so if the cost function was taken to be the posterior log likelihood, the posterior distribution is roughly Gaussian. However the curvature of the quadratic (*i.e.* the inverse dispersion matrix) increases as data is added, so the posterior standard deviation shrinks as $\mathcal{O}\big(\sigma/\sqrt{n_{\mathsf{z}} - n_{\mathsf{x}}}\big)$, where $\mathcal{O}(\sigma)$ characterizes the average standard deviation from a single observation. For $n_{\mathsf{z}} - n_{\mathsf{x}} \gg (\sigma/r)^2$, essentially the entire posterior probability mass lies inside the quadratic region, so the posterior distribution converges asymptotically in probability to a Gaussian. This happens at *any* proper isolated cost minimum at which second order Taylor expansion is locally valid. The approximation gets better with more data (stronger curvature) and smaller higher order Taylor terms.

[2] This result follows from the **Cramér-Rao bound** (*e.g.* [23]), which says that the covariance of any unbiased estimator is bounded below by the **Fisher information** or mean curvature of the posterior log likelihood surface $\langle (\widehat{\mathsf{x}} - \overline{\mathsf{x}})(\widehat{\mathsf{x}} - \overline{\mathsf{x}})^\top \rangle \succeq -\langle \frac{\mathsf{d}^2 \log p}{\mathsf{dx}^2} \rangle$ where p is the posterior probability, x the parameters being estimated, $\widehat{\mathsf{x}}$ the estimate given by any unbiased estimator, $\overline{\mathsf{x}}$ the true underlying x value, and $\mathsf{A} \succeq \mathsf{B}$ denotes positive semidefiniteness of $\mathsf{A} - \mathsf{B}$. Asymptotically, the posterior distribution becomes Gaussian and the Fisher information converges to the inverse dispersion (the curvature of

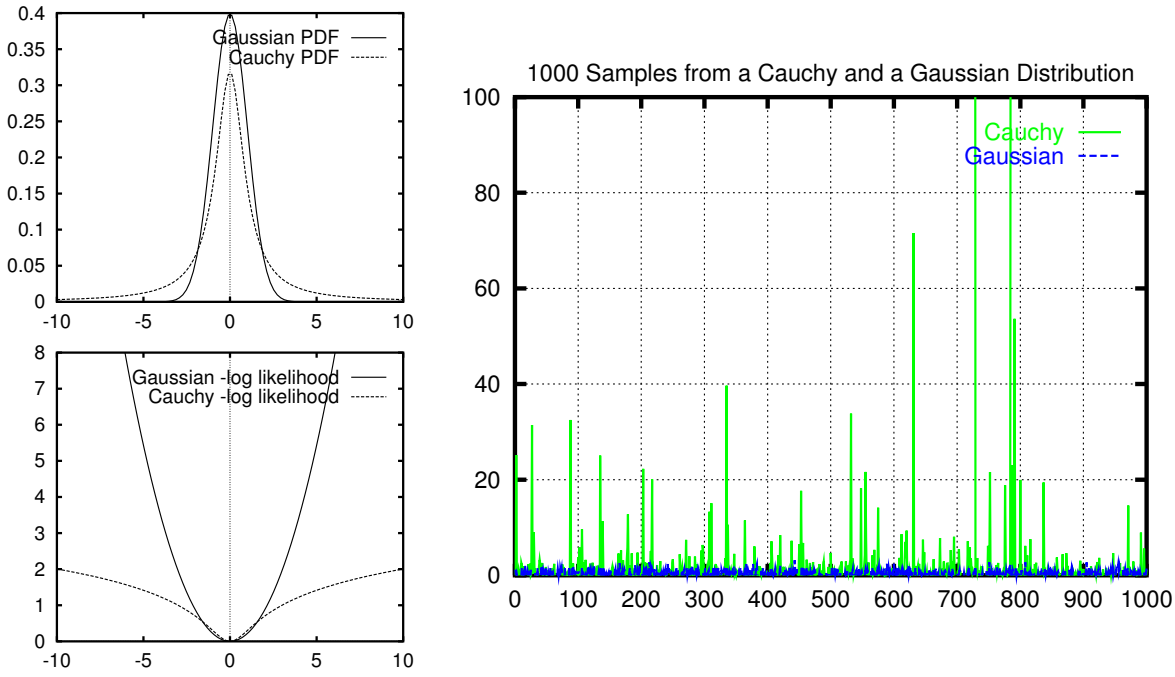


Figure 2: Beware of treating any bell-shaped observation distribution as a Gaussian. Despite being narrower in the peak and broader in the tails, the probability density function of a Cauchy distribution, $p(x) = \left(\pi(1+x^2)\right)^{-1}$, does not look so very different from that of a Gaussian (*top left*). But their negative log likelihoods are very different (*bottom left*), and large deviations ("outliers") are *much* more probable for Cauchy variates than for Gaussian ones (*right*). In fact, the Cauchy distribution has infinite covariance.

Non-asymptotically, the dispersion is not necessarily a good approximation for the covariance of the ML estimator. The asymptotic limit is usually assumed to be a valid for well-designed highly-redundant photogrammetric measurement networks, but recent sampling-based empirical studies of posterior likelihood surfaces [35,80,68] suggest that the case is much less clear for small vision geometry problems and weaker networks. More work is needed on this.

The effect of incorrect error models: It is clear that incorrect modelling of the observation distributions is likely to disturb the ML estimate. Such mismodelling is to some extent inevitable because error distributions stand for influences that we can not fully predict or control. To understand the distortions that unrealistic error models can cause, first realize that geometric fitting is really a special case of parametric probability density estimation. For each set of parameter values, the geometric image projection model and the assumed observation error models combine to predict a probability density for the observations. Maximizing the likelihood corresponds to fitting this *predicted observation density* to the observed data. The geometry and camera model only enter indirectly, via their influence on the predicted distributions.

Accurate noise modelling is just as critical to successful estimation as accurate geometric modelling. The most important mismodelling is failure to take account of the possibility of **outliers** (aberrant data values, caused *e.g.*, by blunders such as incorrect feature correspondences). We stress that so long as the assumed error distributions model the behaviour of *all* of the data used in the

the posterior log likelihood surface at the cost minimum), so the ML estimate attains the Cramér-Rao bound.

fit (including *both* inliers *and* outliers), the above properties of ML estimation including asymptotic minimum variance remain valid in the presence of outliers. In other words, *ML estimation is naturally robust*: there is no need to robustify it so long as realistic error distributions were used in the first place. A distribution that models both inliers and outliers is called a **total distribution**. There is no need to separate the two classes, as ML estimation does not care about the distinction. If the total distribution happens to be an explicit mixture of an inlier and an outlier distribution (*e.g.*, a Gaussian with a locally uniform background of outliers), outliers can be labeled after fitting using likelihood ratio tests, but this is in no way essential to the estimation process.

It is also important to realize the extent to which superficially similar distributions can differ from a Gaussian, or equivalently, how extraordinarily rapidly the tails of a Gaussian distribution fall away compared to more realistic models of real observation errors. See figure 2. In fact, unmodelled outliers typically have very severe effects on the fit. To see this, suppose that the real observations are drawn from a fixed (but perhaps unknown) underlying distribution $p_0(\underline{\mathsf{z}})$. The *law of large numbers* says that their empirical distributions (the observed distribution of each set of samples) converge asymptotically in probability to $p_0(\underline{\mathsf{z}})$. So for each model, the negative log likelihood cost sum $-\sum_i \log p_{\text{model}}(\mathsf{z}_i|\mathsf{x})$ converges to $-n_{\mathsf{z}} \int p_0(\mathsf{z}) \log(p_{\text{model}}(\mathsf{z}|\mathsf{x}))\, \mathsf{dz}$. Up to a model-independent constant, this is n_{z} times the **relative entropy** or **Kullback-Leibler divergence** $\int p_0(\mathsf{z}) \log(p_0(\mathsf{z})/p_{\text{model}}(\mathsf{z}|\mathsf{x}))\, \mathsf{dz}$ of the model distribution w.r.t. the true one $p_0(\underline{\mathsf{z}})$. Hence, even if the model family does not include p_0, *the ML estimate converges asymptotically to the model whose predicted observation distribution has* minimum relative entropy *w.r.t.* p_0. (See, *e.g.* [96, proposition 2.2]). It follows that ML estimates are typically very sensitive to unmodelled outliers, as regions which are relatively probable under p_0 but highly *im*probable under the model make large contributions to the relative entropy. In contrast, allowing for outliers where none actually occur causes relatively little distortion, as no region which is probable under p_0 will have large $-\log p_{\text{model}}$.

In summary, if there is a possibility of outliers, non-robust distribution models such as Gaussians should be replaced with more realistic long-tailed ones such as: mixtures of a narrow 'inlier' and a wide 'outlier' density, Cauchy or α-densities, or densities defined piecewise with a central peaked 'inlier' region surrounded by a constant 'outlier' region[3]. We emphasize again that poor robustness is due entirely to unrealistic distributional assumptions: the maximum likelihood framework itself is naturally robust provided that the total observation distribution including both inliers and outliers is modelled. In fact, real observations can seldom be cleanly divided into inliers and outliers. There is a hard core of outliers such as feature correspondence errors, but there is also a grey area of features that for some reason (a specularity, a shadow, poor focus, motion blur...) were not as accurately located as other features, without clearly being outliers.

3.2 Nonlinear Least Squares

One of the most basic parameter estimation methods is **nonlinear least squares**. Suppose that we have vectors of observations $\underline{\mathsf{z}}_i$ predicted by a model $\mathsf{z}_i = \mathsf{z}_i(\mathsf{x})$, where x is a vector of model parameters. Then nonlinear least squares takes as estimates the parameter values that minimize the **weighted Sum of Squared Error (SSE)** cost function:

$$\mathsf{f}(\mathsf{x}) \equiv \tfrac{1}{2} \sum_i \triangle\mathsf{z}_i(\mathsf{x})^\top \mathsf{W}_i \triangle\mathsf{z}_i(\mathsf{x}), \qquad \triangle\mathsf{z}_i(\mathsf{x}) \equiv \underline{\mathsf{z}}_i - \mathsf{z}_i(\mathsf{x}) \tag{2}$$

[3]The latter case corresponds to a hard inlier / outlier decision rule: for any observation in the 'outlier' region, the density is constant so the observation has no influence at all on the fit. Similarly, the mixture case corresponds to a softer inlier / outlier decision rule.

Here, $\triangle \mathsf{z}_i(\mathsf{x})$ is the feature prediction error and W_i is an arbitrary symmetric positive definite (SPD) **weight matrix**. Modulo normalization terms independent of x, the weighted SSE cost function coincides with the negative log likelihood for observations $\underline{\mathsf{z}}_i$ perturbed by Gaussian noise of mean zero and covariance W_i^{-1}. So for least squares to have a useful statistical interpretation, the W_i should be chosen to approximate the inverse measurement covariance of $\underline{\mathsf{z}}_i$. Even for non-Gaussian noise with this mean and covariance, the **Gauss-Markov theorem** [37, 11] states that if the models $\mathsf{z}_i(\mathsf{x})$ are linear, least squares gives the Best Linear Unbiased Estimator (BLUE), where 'best' means minimum variance[4].

Any weighted least squares model can be converted to an unweighted one ($\mathsf{W}_i = \mathsf{1}$) by pre-multiplying $\underline{\mathsf{z}}_i, \mathsf{z}_i, \triangle \mathsf{z}_i$ by any $\mathsf{L}_i^\top$ satisfying $\mathsf{W}_i = \mathsf{L}_i \mathsf{L}_i^\top$. Such an L_i can be calculated efficiently from W_i or W_i^{-1} using Cholesky decomposition (§B.1). $\overline{\triangle \mathsf{z}}_i = \mathsf{L}_i^\top \triangle \mathsf{z}_i$ is called a **standardized residual**, and the resulting unweighted least squares problem $\min_{\mathsf{x}} \frac{1}{2} \sum_i \|\overline{\triangle \mathsf{z}}_i(\mathsf{x})\|^2$ is said to be in **standard form**. One advantage of this is that optimization methods based on linear least squares solvers can be used in place of ones based on linear (normal) equation solvers, which allows ill-conditioned problems to be handled more stably (§B.2).

Another peculiarity of the SSE cost function is its indifference to the natural boundaries between the observations. If observations $\underline{\mathsf{z}}_i$ from any sources are assembled into a compound observation vector $\underline{\mathsf{z}} \equiv (\underline{\mathsf{z}}_1^\top, \ldots, \underline{\mathsf{z}}_k^\top)^\top$, and their weight matrices W_i are assembled into compound block diagonal weight matrix $\mathsf{W} \equiv \mathrm{diag}(\mathsf{W}_1, \ldots, \mathsf{W}_k)$, the weighted squared error $\mathsf{f}(\mathsf{x}) \equiv \frac{1}{2} \triangle \mathsf{z}(\mathsf{x})^\top \mathsf{W} \triangle \mathsf{z}(\mathsf{x})$ is the same as the original SSE cost function, $\frac{1}{2} \sum_i \triangle \mathsf{z}_i(\mathsf{x})^\top \mathsf{W}_i \triangle \mathsf{z}_i(\mathsf{x})$. The general quadratic form of the SSE cost is preserved under such compounding, and also under arbitrary linear transformations of $\underline{\mathsf{z}}$ that mix components from different observations. The only place that the underlying structure is visible is in the block structure of W. Such invariances do not hold for essentially any other cost function, but they simplify the formulation of least squares considerably.

3.3 Robustified Least Squares

The main problem with least squares is its high sensitivity to outliers. This happens because the Gaussian has extremely small tails compared to most real measurement error distributions. For robust estimates, we must choose a more realistic likelihood model (§3.1). The exact functional form is less important than the general way in which the expected types of outliers enter. A single blunder such as a correspondence error may affect one or a few of the observations, but it will usually leave all of the others unchanged. This locality is the whole basis of robustification. If we can decide which observations were affected, we can down-weight or eliminate them and use the remaining observations for the parameter estimates as usual. If all of the observations had been affected about equally (*e.g.* as by an incorrect projection model), we might still know that something was wrong, but not be able to fix it by simple data cleaning.

We will adopt a 'single layer' robustness model, in which the observations are partitioned into independent groups $\underline{\mathsf{z}}_i$, each group being irreducible in the sense that it is accepted, down-weighted or rejected as a whole, independently of all the other groups. The partitions should reflect the types of blunders that occur. For example, if feature correspondence errors are the most common blunders, the two coordinates of a single image point would naturally form a group as both would usually be invalidated by such a blunder, while no other image point would be affected. Even if one of the coordinates appeared to be correct, if the other were incorrect we would usually want to discard both

[4] It may be possible (and even useful) to do better with either biased (towards the correct solution), or nonlinear estimators.

for safety. On the other hand, in stereo problems, the four coordinates of each pair of corresponding image points might be a more natural grouping, as a point in one image is useless without its correspondent in the other one.

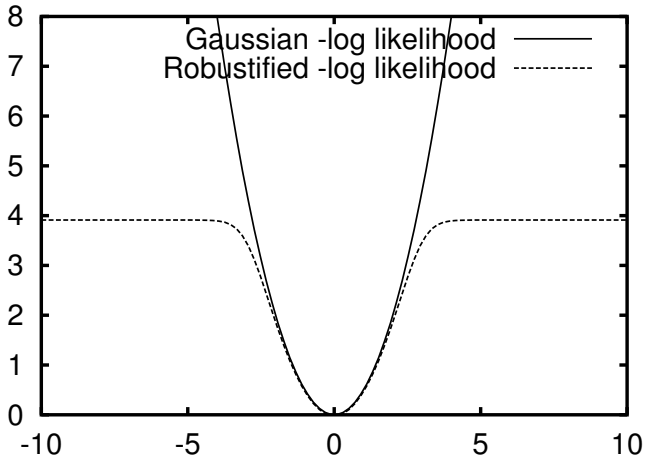
Henceforth, when we say *observation* we mean *irreducible group of observations treated as a unit by the robustifying model*. *I.e.*, our observations need not be scalars, but they must be units, probabilistically independent of one another irrespective of whether they are inliers or outliers.

As usual, each independent observation $\underline{\mathsf{z}}_i$ contributes an independent term $\mathsf{f}_i(\mathsf{x} \,|\, \underline{\mathsf{z}}_i)$ to the total cost function. This could have more or less any form, depending on the expected total distribution of inliers and outliers for the observation. One very natural family are the **radial distributions**, which have negative log likelihoods of the form:

$$\mathsf{f}_i(\mathsf{x}) \;\equiv\; \tfrac{1}{2}\,\rho_i(\;\triangle\mathsf{z}_i(\mathsf{x})^\top\,\mathsf{W}_i\,\triangle\mathsf{z}_i(\mathsf{x})\;) \tag{3}$$

Here, $\rho_i(s)$ can be any increasing function with $\rho_i(0) = 0$ and $\frac{d}{ds}\rho_i(0) = 1$. (These guarantee that at $\triangle\mathsf{z}_i = \mathsf{0}$, f vanishes and $\frac{\mathsf{d}^2\mathsf{f}_i}{\mathsf{dz}_i^2} = \mathsf{W}_i$). Weighted SSE has $\rho_i(s) = s$, while more robust variants have sublinear ρ_i, often tending to a constant at ∞ so that distant outliers are entirely ignored. The dispersion matrix W_i^{-1} determines the spatial spread of $\underline{\mathsf{z}}_i$, and up to scale its covariance (if this is finite). The radial form is preserved under arbitrary affine transformations of $\underline{\mathsf{z}}_i$, so within a group, all of the observations are on an equal footing in the same sense as in least squares. However, non-Gaussian radial distributions are almost never *separable*: the observations in $\underline{\mathsf{z}}_i$ can neither be split into independent subgroups, nor combined into larger groups, without destroying the radial form. Radial cost models do not have the remarkable isotropy of non-robust SSE, but this is exactly what we wanted, as it ensures that all observations in a group will be either left alone, or down-weighted together.

As an example of this, for image features polluted with occasional large outliers caused by correspondence errors, we might model the error distribution as a Gaussian central peak plus a uniform background of outliers. This would give negative log likelihood contributions of the form $\mathsf{f}(\mathsf{x}) = -\log\left(\exp(-\frac{1}{2}\chi_{ip}^2) + \epsilon\right)$ instead of the non-robust weighted SSE model $\mathsf{f}(\mathsf{x}) = \frac{1}{2}\chi_{ip}^2$, where $\chi_{ip}^2 = \triangle\boldsymbol{x}_{ip}^\top\,\mathsf{W}_{ip}\,\triangle\boldsymbol{x}_{ip}$ is the squared weighted residual error (which is a χ^2 variable for a correct model and Gaussian error distribution), and ϵ parametrizes the frequency of outliers.



3.4 Intensity-based methods

The above models apply not only to geometric image features, but also to intensity-based matching of image patches. In this case, the observables are image gray-scales or colors $\mathbf{I}$ rather than feature coordinates u, and the error model is based on intensity residuals. To get from a point projection model $\mathsf{u} = \mathsf{u}(\mathsf{x})$ to an intensity based one, we simply compose with the assumed local intensity model $\mathbf{I} = \mathbf{I}(\mathsf{u})$ (*e.g.* obtained from an image template or another image that we are matching against), premultiply point Jacobians by point-to-intensity Jacobians $\frac{\mathsf{d}\mathbf{I}}{\mathsf{du}}$, *etc*. The full range of intensity models can be implemented within this framework: pure translation, affine, quadratic or homographic patch deformation models, 3D model based intensity predictions, coupled affine or spline patches for surface coverage, *etc*., [1,52,55,9,110,94,53,97,76,104,102]. The structure of intensity based bundle problems is very similar to that of feature based ones, so all of the techniques studied below can be applied.

We will not go into more detail on intensity matching, except to note that it is the real basis of feature based methods. Feature detectors are optimized for detection not localization. To localize a detected feature accurately we need to match (some function of) the image intensities in its region against either an idealized template or another image of the feature, using an appropriate geometric deformation model, *etc*. For example, suppose that the intensity matching model is $\mathsf{f}(\mathsf{u}) = \frac{1}{2}\iint \rho(\|\boldsymbol{\delta}\mathbf{I}(\mathsf{u})\|^2)$ where the integration is over some image patch, $\boldsymbol{\delta}\mathbf{I}$ is the current intensity prediction error, u parametrizes the local geometry (patch translation & warping), and $\rho(\cdot)$ is some intensity error robustifier. Then the cost gradient in terms of u is $\mathsf{g}_\mathsf{u}^\top = \frac{\mathsf{df}}{\mathsf{du}} = \iint \rho' \, \boldsymbol{\delta}\mathbf{I}^\top \frac{\mathsf{d}\mathbf{I}}{\mathsf{du}}$. Similarly, the cost Hessian in u in a Gauss-Newton approximation is $\mathsf{H}_\mathsf{u} = \frac{\mathsf{d}^2\mathsf{f}}{\mathsf{du}^2} \approx \iint \rho'' \, (\frac{\mathsf{d}\mathbf{I}}{\mathsf{du}})^\top \frac{\mathsf{d}\mathbf{I}}{\mathsf{du}}$. In a feature based model, we express $\mathsf{u} = \mathsf{u}(\mathsf{x})$ as a function of the bundle parameters, so if $\mathsf{J}_\mathsf{u} = \frac{\mathsf{du}}{\mathsf{dx}}$ we have a corresponding cost gradient and Hessian contribution $\mathsf{g}_\mathsf{x}^\top = \mathsf{g}_\mathsf{u}^\top \mathsf{J}_\mathsf{u}$ and $\mathsf{H}_\mathsf{x} = \mathsf{J}_\mathsf{u}^\top \mathsf{H}_\mathsf{u} \mathsf{J}_\mathsf{u}$. In other words, the intensity matching model is locally equivalent to a quadratic feature matching one on the 'features' $\mathsf{u}(\mathsf{x})$, with effective weight (inverse covariance) matrix $\mathsf{W}_\mathsf{u} = \mathsf{H}_\mathsf{u}$. All image feature error models in vision are ultimately based on such an underlying intensity matching model. As feature covariances are a function of intensity gradients $\iint \rho'' \, (\frac{\mathsf{d}\mathbf{I}}{\mathsf{du}})^\top \frac{\mathsf{d}\mathbf{I}}{\mathsf{du}}$, they can be both highly variable between features (depending on how much local gradient there is), and highly anisotropic (depending on how directional the gradients are). *E.g.*, for points along a 1D intensity edge, the uncertainty is large in the along edge direction and small in the across edge one.

3.5 Implicit models

Sometimes observations are most naturally expressed in terms of an implicit observation-constraining model $\mathsf{h}(\mathsf{x},\mathsf{z}) = \mathsf{0}$, rather than an explicit observation-predicting one $\mathsf{z} = \mathsf{z}(\mathsf{x})$. (The associated image error still has the form $\mathsf{f}(\underline{\mathsf{z}} - \mathsf{z})$). For example, if the model is a 3D curve and we observe points on it (the noisy images of 3D points that may lie anywhere along the 3D curve), we can predict the whole image curve, but not the exact position of each observation along it. We only have the constraint that the noiseless image of the observed point would lie on the noiseless image of the curve, if we knew these. There are basically two ways to handle implicit models: nuisance parameters and reduction.

Nuisance parameters: In this approach, the model is made explicit by adding additional 'nuisance' parameters representing something equivalent to model-consistent estimates of the unknown noise free observations, *i.e.* to z with $\mathsf{h}(\mathsf{x},\mathsf{z}) = \mathsf{0}$. The most direct way to do this is to include the entire parameter vector z as nuisance parameters, so that we have to solve a constrained optimization problem on the extended parameter space (x,z), minimizing $\mathsf{f}(\underline{\mathsf{z}} - \mathsf{z})$ over (x,z) subject to $\mathsf{h}(\mathsf{x},\mathsf{z}) = \mathsf{0}$.

This is a sparse constrained problem, which can be solved efficiently using sparse matrix techniques (§6.3). In fact, for image observations, the subproblems in z (optimizing $f(\underline{z} - z)$ over z for fixed $\underline{z}$ and x) are small and for typical f rather simple. So in spite of the extra parameters z, optimizing this model is not significantly more expensive than optimizing an explicit one $z = z(x)$ [14, 13, 105, 106]. For example, when estimating matching constraints between image pairs or triplets [60, 62], instead of using an explicit 3D representation, pairs or triplets of corresponding image points can be used as features z_i, subject to the epipolar or trifocal geometry contained in x [105, 106].

However, if a smaller nuisance parameter vector than z can be found, it is wise to use it. In the case of a curve, it suffices to include just one nuisance parameter per observation, saying where along the curve the corresponding noise free observation is predicted to lie. This model exactly satisfies the constraints, so it converts the implicit model to an unconstrained explicit one $z = z(x, \lambda)$, where λ are the along-curve nuisance parameters.

The advantage of the nuisance parameter approach is that it gives the exact optimal parameter estimate for x, and jointly, optimal x-consistent estimates for the noise free observations z.

Reduction: Alternatively, we can regard $h(x, \underline{z})$ rather than $\underline{z}$ as the observation vector, and hence fit the parameters to the explicit log likelihood model for $h(x, \underline{z})$. To do this, we must transfer the underlying error model / distribution $f(\triangle z)$ on $\underline{z}$ to one $f(h)$ on $h(x, \underline{z})$. In principle, this should be done by marginalization: the density for h is given by integrating that for $\triangle z$ over all $\triangle z$ giving the same h. Within the point estimation framework, it can be approximated by replacing the integration with maximization. Neither calculation is easy in general, but in the asymptotic limit where first order Taylor expansion $h(x, \underline{z}) = h(x, z + \triangle z) \approx 0 + \frac{dh}{dz} \triangle z$ is valid, the distribution of h is a marginalization or maximization of that of $\triangle z$ over affine subspaces. This can be evaluated in closed form for some robust distributions. Also, standard covariance propagation gives (more precisely, this applies to the h and $\triangle z$ dispersions):

$$\langle h(x, \underline{z}) \rangle \approx 0\,, \qquad \langle h(x, \underline{z})\, h(x, \underline{z})^\top \rangle \approx \frac{dh}{dz} \langle \triangle z \triangle z^\top \rangle \frac{dh}{dz}^\top = \frac{dh}{dz} W^{-1} \frac{dh}{dz}^\top \tag{4}$$

where W^{-1} is the covariance of $\triangle z$. So at least for an outlier-free Gaussian model, the reduced distribution remains Gaussian (albeit with x-dependent covariance).

4 Basic Numerical Optimization

Having chosen a suitable model quality metric, we must optimize it. This section gives a very rapid sketch of the basic local optimization methods for differentiable functions. See [29, 93, 42] for more details. We need to minimize a cost function $f(x)$ over parameters x, starting from some given initial estimate x of the minimum, presumably supplied by some approximate visual reconstruction method or prior knowledge of the approximate situation. As in §2.2, the parameter space may be nonlinear, but we assume that local displacements can be parametrized by a local coordinate system / vector of free parameters δx. We try to find a displacement $x \to x + \delta x$ that locally minimizes or at least reduces the cost function. Real cost functions are too complicated to minimize in closed form, so instead we minimize an approximate **local model** for the function, *e.g.* based on Taylor expansion or some other approximation at the current point x. Although this does not usually give the exact minimum, with luck it will improve on the initial parameter estimate and allow us to iterate to convergence. The art of reliable optimization is largely in the details that make this happen even without luck: which local model, how to minimize it, how to ensure that the estimate is improved, and how to decide when convergence has occurred. If you not are interested in such subjects, use a professionally designed package (§C.2): details *are* important here.

4.1 Second Order Methods

The reference for all local models is the quadratic Taylor series one:

$$\underbrace{f(x+\delta x) \approx f(x) + g^\top \delta x + \tfrac{1}{2}\delta x^\top H\, \delta x}_{\text{quadratic local model}} \qquad \underbrace{g \equiv \frac{df}{dx}(x)}_{\text{gradient vector}} \qquad \underbrace{H \equiv \frac{d^2 f}{dx^2}(x)}_{\text{Hessian matrix}} \tag{5}$$

For now, assume that the Hessian H is positive definite (but see below and §9). The local model is then a simple quadratic with a unique global minimum, which can be found explicitly using linear algebra. Setting $\frac{df}{dx}(x+\delta x) \approx H\,\delta x + g$ to zero for the stationary point gives the **Newton step**:

$$\delta x = -H^{-1} g \tag{6}$$

The estimated new function value is $f(x+\delta x) \approx f(x) - \frac{1}{2}\delta x^\top H\, \delta x = f(x) - \frac{1}{2} g^\top H^{-1} g$. Iterating the Newton step gives **Newton's method**. This is the canonical optimization method for smooth cost functions, owing to its exceptionally rapid theoretical and practical convergence near the minimum. For quadratic functions it converges in one iteration, and for more general analytic ones its **asymptotic convergence** is **quadratic**: as soon as the estimate gets close enough to the solution for the second order Taylor expansion to be reasonably accurate, the residual state error is approximately *squared* at each iteration. This means that the number of significant digits in the estimate approximately doubles at each iteration, so starting from any reasonable estimate, at most about $\log_2(16) + 1 \approx 5$–6 iterations are needed for full double precision (16 digit) accuracy. Methods that potentially achieve such rapid asymptotic convergence are called **second order methods**. This is a high accolade for a local optimization method, but it can only be achieved if the Newton step is asymptotically well approximated. Despite their conceptual simplicity and asymptotic performance, Newton-like methods have some disadvantages:

- To guarantee convergence, a suitable step control policy must be added (§4.2).
- Solving the $n \times n$ Newton step equations takes time $\mathcal{O}\left(n^3\right)$ for a dense system (§B.1), which can be prohibitive for large n. Although the cost can often be reduced (very substantially for bundle adjustment) by exploiting sparseness in H, it remains true that Newton-like methods tend to have a high cost per iteration, which increases relative to that of other methods as the problem size increases. For this reason, it is sometimes worthwhile to consider more approximate **first order methods** (§7), which are occasionally more efficient, and generally simpler to implement, than sparse Newton-like methods.
- Calculating second derivatives H is by no means trivial for a complicated cost function, both computationally, and in terms of implementation effort. The **Gauss-Newton** method (§4.3) offers a simple analytic approximation to H for nonlinear least squares problems. Some other methods build up approximations to H from the way the gradient g changes during the iteration are in use (see §7.1, Krylov methods).
- The asymptotic convergence of Newton-like methods is sometimes felt to be an expensive luxury when far from the minimum, especially when damping (see below) is active. However, it must be said that Newton-like methods generally do require significantly fewer iterations than first order ones, even far from the minimum.

4.2 Step Control

Unfortunately, Newton's method can fail in several ways. It may converge to a saddle point rather than a minimum, and for large steps the second order cost prediction may be inaccurate, so there is no

guarantee that the true cost will actually decrease. To guarantee convergence to a minimum, the step must follow a local **descent direction** (a direction with a non-negligible component down the local cost gradient, or if the gradient is zero near a saddle point, down a negative curvature direction of the Hessian), and it must make reasonable progress in this direction (neither so little that the optimization runs slowly or stalls, nor so much that it greatly overshoots the cost minimum along this direction). It is also necessary to decide when the iteration has converged, and perhaps to limit any over-large steps that are requested. Together, these topics form the delicate subject of **step control**.

To choose a descent direction, one can take the Newton step direction if this descends (it may not near a saddle point), or more generally some combination of the Newton and gradient directions. **Damped Newton methods** solve a regularized system to find the step:

$$(\mathsf{H} + \lambda \mathsf{W})\, \delta\mathsf{x} \;=\; -\mathsf{g} \tag{7}$$

Here, λ is some weighting factor and W is some positive definite weight matrix (often the identity, so $\lambda \to \infty$ becomes steepest descent $\delta\mathsf{x} \propto -\mathsf{g}$). λ can be chosen to limit the step to a dynamically chosen maximum size (**trust region methods**), or manipulated more heuristically, to shorten the step if the prediction is poor (**Levenberg-Marquardt methods**).

Given a descent direction, progress along it is usually assured by a **line search** method, of which there are many based on quadratic and cubic 1D cost models. If the suggested (*e.g.* Newton) step is $\delta\mathsf{x}$, line search finds the α that actually minimizes f along the line $\mathsf{x} + \alpha\,\delta\mathsf{x}$, rather than simply taking the estimate $\alpha = 1$.

There is no space for further details on step control here (again, see [29,93,42]). However note that poor step control can make a huge difference in reliability and convergence rates, especially for ill-conditioned problems. Unless you are familiar with these issues, it is advisable to use professionally designed methods.

4.3 Gauss-Newton and Least Squares

Consider the nonlinear weighted SSE cost model $\mathsf{f}(\mathsf{x}) \equiv \frac{1}{2}\,\triangle\mathsf{z}(\mathsf{x})^\top \mathsf{W}\,\triangle\mathsf{z}(\mathsf{x})$ (§3.2) with prediction error $\triangle\mathsf{z}(\mathsf{x}) = \underline{\mathsf{z}} - \mathsf{z}(\mathsf{x})$ and weight matrix W. Differentiation gives the gradient and Hessian in terms of the **Jacobian** or **design matrix** of the predictive model, $\mathsf{J} \equiv \frac{\mathsf{dz}}{\mathsf{dx}}$:

$$\mathsf{g} \;\equiv\; \tfrac{\mathsf{df}}{\mathsf{dx}} \;=\; \triangle\mathsf{z}^\top \mathsf{W}\,\mathsf{J} \qquad\qquad \mathsf{H} \;\equiv\; \tfrac{\mathsf{d}^2\mathsf{f}}{\mathsf{dx}^2} \;=\; \mathsf{J}^\top \mathsf{W}\,\mathsf{J} \;+\; \textstyle\sum_i (\triangle\mathsf{z}^\top \mathsf{W})_i\, \tfrac{\mathsf{d}^2\mathsf{z}_i}{\mathsf{dx}^2} \tag{8}$$

These formulae could be used directly in a damped Newton method, but the $\frac{\mathsf{d}^2\mathsf{z}_i}{\mathsf{dx}^2}$ term in H is likely to be small in comparison to the corresponding components of $\mathsf{J}^\top \mathsf{W}\,\mathsf{J}$ if either: (*i*) the prediction error $\triangle\mathsf{z}(\mathsf{x})$ is small; or (*ii*) the model is nearly linear, $\frac{\mathsf{d}^2\mathsf{z}_i}{\mathsf{dx}^2} \approx \mathsf{0}$. Dropping the second term gives the **Gauss-Newton approximation** to the least squares Hessian, $\mathsf{H} \approx \mathsf{J}^\top \mathsf{W}\,\mathsf{J}$. With this approximation, the Newton step prediction equations become the **Gauss-Newton** or **normal** equations:

$$(\mathsf{J}^\top \mathsf{W}\,\mathsf{J})\, \delta\mathsf{x} \;=\; -\mathsf{J}^\top \mathsf{W}\,\triangle\mathsf{z} \tag{9}$$

The Gauss-Newton approximation is extremely common in nonlinear least squares, and practically all current bundle implementations use it. Its main advantage is simplicity: the second derivatives of the projection model $\mathsf{z}(\mathsf{x})$ are complex and troublesome to implement.

In fact, the normal equations are just one of many methods of solving the weighted linear least squares problem[5] $\min_{\delta\mathsf{x}} \frac{1}{2}(\mathsf{J}\,\delta\mathsf{x} - \triangle\mathsf{z})^\top \mathsf{W}\,(\mathsf{J}\,\delta\mathsf{x} - \triangle\mathsf{z})$. Another notable method is that based on

[5] Here, the dependence of J on x is ignored, which amounts to the same thing as ignoring the $\frac{\mathsf{d}^2\mathsf{z}_i}{\mathsf{dx}^2}$ term in H.

QR decomposition (§B.2, [11,44]), which is up to a factor of two slower than the normal equations, but much less sensitive to ill-conditioning in J [6].

Whichever solution method is used, the main disadvantage of the Gauss-Newton approximation is that when the discarded terms are not negligible, the convergence rate is greatly reduced (§7.2). In our experience, such reductions are indeed common in highly nonlinear problems with (at the current step) large residuals. For example, near a saddle point the Gauss-Newton approximation is *never* accurate, as its predicted Hessian is always at least positive semidefinite. However, for well-parametrized (*i.e.* locally near linear, §2.2) bundle problems under an outlier-free least squares cost model evaluated near the cost minimum, the Gauss-Newton approximation is usually very accurate. Feature extraction errors and hence $\triangle\mathsf{z}$ and W^{-1} have characteristic scales of at most a few pixels. In contrast, the nonlinearities of $\mathsf{z}(\mathsf{x})$ are caused by nonlinear 3D feature-camera geometry (perspective effects) and nonlinear image projection (lens distortion). For typical geometries and lenses, neither effect varies significantly on a scale of a few pixels. So the nonlinear corrections are usually small compared to the leading order linear terms, and bundle adjustment behaves as a near-linear small residual problem.

However note that this does *not* extend to robust cost models. Robustification works by introducing strong nonlinearity into the cost function at the scale of typical feature reprojection errors. For accurate step prediction, the optimization routine must take account of this. For radial cost functions (§3.3), a reasonable compromise is to take account of the exact second order derivatives of the robustifiers $\rho_i(\cdot)$, while retaining only the first order Gauss-Newton approximation for the predicted observations $\mathsf{z}_i(\mathsf{x})$. If ρ_i' and ρ'' are respectively the first and second derivatives of ρ_i at the current evaluation point, we have a **robustified Gauss-Newton approximation**:

$$\mathsf{g}_i \;=\; \rho_i' \, \mathsf{J}_i^\top \mathsf{W}_i \triangle\mathsf{z}_i \qquad \mathsf{H}_i \;\approx\; \mathsf{J}_i^\top \left(\rho_i' \, \mathsf{W}_i + 2\, \rho_i'' \, (\mathsf{W}_i \triangle\mathsf{z}_i) \, (\mathsf{W}_i \triangle\mathsf{z}_i)^\top \right) \mathsf{J}_i \tag{10}$$

So robustification has two effects: (*i*) it down-weights the entire observation (both g_i and H_i) by ρ_i'; and (*ii*) it makes a rank-one reduction[7] of the curvature H_i in the radial ($\triangle\mathsf{z}_i$) direction, to account for the way in which the weight changes with the residual. There are reweighting-based optimization methods that include only the first effect. They still find the true cost minimum $\mathsf{g} = \mathsf{0}$ as the g_i are evaluated exactly[8], but convergence may be slowed owing to inaccuracy of H, especially for the mainly radial deviations produced by non-robust initializers containing outliers. H_i has a direction of negative curvature if $\rho_i'' \triangle\mathsf{z}_i^\top \mathsf{W}_i \triangle\mathsf{z}_i < -\frac{1}{2}\rho_i'$, but if not we can even reduce the robustified Gauss-Newton model to a local unweighted SSE one for which linear least squares methods can be used. For simplicity suppose that W_i has already reduced to $\mathsf{1}$ by premultiplying z_i and J_i by $\mathsf{L}_i^\top$ where $\mathsf{L}_i \, \mathsf{L}_i^\top = \mathsf{W}_i$. Then minimizing the **effective squared error** $\frac{1}{2}\|\overline{\delta\mathsf{z}}_i - \overline{\mathsf{J}}_i \, \delta\mathsf{x}\|^2$ gives the correct second

[6] The QR method gives the solution to a relative error of about $\mathcal{O}(C\epsilon)$, as compared to $\mathcal{O}\left(C^2\epsilon\right)$ for the normal equations, where C is the condition number (the ratio of the largest to the smallest singular value) of J, and ϵ is the machine precision (10^{-16} for double precision floating point).

[7] The useful robustifiers ρ_i are sublinear, with $\rho_i' < 1$ and $\rho_i'' < 0$ in the outlier region.

[8] Reweighting is also sometimes used in vision to handle projective homogeneous scale factors rather than error weighting. *E.g.*, suppose that image points $(u/w, v/w)^\top$ are generated by a homogeneous projection equation $(u, v, w)^\top = \mathbf{P}\,(X, Y, Z, 1)^\top$, where $\mathbf{P}$ is the 3×4 homogeneous image projection matrix. A scale factor reweighting scheme might take derivatives w.r.t. u, v while treating the inverse weight w as a constant within each iteration. Minimizing the resulting globally bilinear linear least squares error model over $\mathbf{P}$ and $(X, Y, Z)^\top$ does *not* give the true cost minimum: it zeros the gradient-ignoring-w-variations, not the true cost gradient. Such schemes should not be used for precise work as the bias can be substantial, especially for wide-angle lenses and close geometries.

order robust state update, where $\alpha \equiv \text{RootOf}(\frac{1}{2}\alpha^2 - \alpha - \rho_i''/\rho_i' \, \|\triangle \mathsf{z}_i\|^2)$ and:

$$\overline{\boldsymbol{\delta}\mathsf{z}}_i \;\equiv\; \frac{\sqrt{\rho_i'}}{1-\alpha}\,\triangle \mathsf{z}_i(\mathsf{x}) \qquad \overline{\mathsf{J}}_i \;\equiv\; \sqrt{\rho_i'}\left(\mathsf{1} - \alpha\,\frac{\triangle \mathsf{z}_i\,\triangle \mathsf{z}_i^\top}{\|\triangle \mathsf{z}_i\|^2}\right)\mathsf{J}_i \tag{11}$$

In practice, if $\rho_i'' \, \|\triangle \mathsf{z}_i\|^2 \lesssim -\frac{1}{2}\rho_i'$, we can use the same formulae but limit $\alpha \leq 1 - \epsilon$ for some small ϵ. However, the full curvature correction is not applied in this case.

4.4 Constrained Problems

More generally, we may want to minimize a function $\mathsf{f}(\mathsf{x})$ subject to a set of constraints $\mathsf{c}(\mathsf{x}) = \mathsf{0}$ on x. These might be scene constraints, internal consistency constraints on the parametrization (§2.2), or constraints arising from an implicit observation model (§3.5). Given an initial estimate x of the solution, we try to improve this by optimizing the quadratic local model for f subject to a linear local model of the constraints c. This linearly constrained quadratic problem has an exact solution in linear algebra. Let g, H be the gradient and Hessian of f as before, and let the first order expansion of the constraints be $\mathsf{c}(\mathsf{x}+\boldsymbol{\delta}\mathsf{x}) \;\approx\; \mathsf{c}(\mathsf{x}) + \mathsf{C}\,\boldsymbol{\delta}\mathsf{x}$ where $\mathsf{C} \;\equiv\; \frac{\mathsf{dc}}{\mathsf{dx}}$. Introduce a vector of Lagrange multipliers $\boldsymbol{\lambda}$ for c. We seek the $\mathsf{x}+\boldsymbol{\delta}\mathsf{x}$ that optimizes $\mathsf{f}+\mathsf{c}^\top\boldsymbol{\lambda}$ subject to $\mathsf{c} = \mathsf{0}$, *i.e.* $\mathsf{0} = \frac{\mathsf{d}}{\mathsf{dx}}(\mathsf{f}+\mathsf{c}^\top\boldsymbol{\lambda})(\mathsf{x}+\boldsymbol{\delta}\mathsf{x}) \;\approx\; \mathsf{g}+\mathsf{H}\,\boldsymbol{\delta}\mathsf{x}+\mathsf{C}^\top\boldsymbol{\lambda}$ and $\mathsf{0} = \mathsf{c}(\mathsf{x}+\boldsymbol{\delta}\mathsf{x}) \approx \mathsf{c}(\mathsf{x})+\mathsf{C}\,\boldsymbol{\delta}\mathsf{x}$. Combining these gives the **Sequential Quadratic Programming (SQP)** step:

$$\begin{pmatrix}\mathsf{H} & \mathsf{C}^\top \\ \mathsf{C} & \mathsf{0}\end{pmatrix}\begin{pmatrix}\boldsymbol{\delta}\mathsf{x} \\ \boldsymbol{\lambda}\end{pmatrix} \;=\; -\begin{pmatrix}\mathsf{g} \\ \mathsf{c}\end{pmatrix}, \quad \mathsf{f}(\mathsf{x}+\boldsymbol{\delta}\mathsf{x}) \;\approx\; \mathsf{f}(\mathsf{x}) - \tfrac{1}{2}\begin{pmatrix}\mathsf{g}^\top & \mathsf{c}^\top\end{pmatrix}\begin{pmatrix}\mathsf{H} & \mathsf{C}^\top \\ \mathsf{C} & \mathsf{0}\end{pmatrix}^{-1}\begin{pmatrix}\mathsf{g} \\ \mathsf{c}\end{pmatrix} \tag{12}$$

$$\begin{pmatrix}\mathsf{H} & \mathsf{C}^\top \\ \mathsf{C} & \mathsf{0}\end{pmatrix}^{-1} \;=\; \begin{pmatrix}\mathsf{H}^{-1} - \mathsf{H}^{-1}\mathsf{C}^\top\mathsf{D}^{-1}\mathsf{C}\,\mathsf{H}^{-1} & \mathsf{H}^{-1}\mathsf{C}^\top\mathsf{D}^{-1} \\ \mathsf{D}^{-1}\mathsf{C}\,\mathsf{H}^{-1} & -\mathsf{D}^{-1}\end{pmatrix}, \qquad \mathsf{D} \;\equiv\; \mathsf{C}\,\mathsf{H}^{-1}\mathsf{C}^\top \tag{13}$$

At the optimum $\boldsymbol{\delta}\mathsf{x}$ and c vanish, but $\mathsf{C}^\top\boldsymbol{\lambda} = -\mathsf{g}$, which is generally non-zero.

An alternative constrained approach uses the linearized constraints to eliminate some of the variables, then optimizes over the rest. Suppose that we can order the variables to give partitions $\mathsf{x} = (\mathsf{x}_1 \; \mathsf{x}_2)^\top$ and $\mathsf{C} = (\mathsf{C}_1 \; \mathsf{C}_2)$, where C_1 is square and invertible. Then using $\mathsf{C}_1\,\mathsf{x}_1 + \mathsf{C}_2\,\mathsf{x}_2 = \mathsf{C}\,\mathsf{x} = -\mathsf{c}$, we can solve for x_1 in terms of x_2 and c: $\mathsf{x}_1 = -\mathsf{C}_1^{-1}(\mathsf{C}_2\mathsf{x}_2 + \mathsf{c})$. Substituting this into the quadratic cost model has the effect of eliminating x_1, leaving a smaller unconstrained **reduced problem** $\overline{\mathsf{H}}_{22}\,\mathsf{x}_2 = -\overline{\mathsf{g}}_2$, where:

$$\overline{\mathsf{H}}_{22} \;\equiv\; \mathsf{H}_{22} - \mathsf{H}_{21}\,\mathsf{C}_1^{-1}\mathsf{C}_2 - \mathsf{C}_2^\top\mathsf{C}_1^{-\top}\mathsf{H}_{12} + \mathsf{C}_2^\top\mathsf{C}_1^{-\top}\mathsf{H}_{11}\,\mathsf{C}_1^{-1}\mathsf{C}_2 \tag{14}$$

$$\overline{\mathsf{g}}_2 \;\equiv\; \mathsf{g}_2 - \mathsf{C}_2^\top\mathsf{C}_1^{-\top}\mathsf{g}_1 - (\mathsf{H}_{21} - \mathsf{C}_2^\top\mathsf{C}_1^{-\top}\mathsf{H}_{11})\,\mathsf{C}_1^{-1}\mathsf{c} \tag{15}$$

(These matrices can be evaluated efficiently using simple matrix factorization schemes [11]). This method is stable provided that the chosen C_1 is well-conditioned. It works well for dense problems, but is not always suitable for sparse ones because if C is dense, the reduced Hessian $\overline{\mathsf{H}}_{22}$ becomes dense too.

For least squares cost models, constraints can also be handled within the linear least squares framework, *e.g.* see [11].

4.5 General Implementation Issues

Before going into details, we mention a few points of good numerical practice for large-scale optimization problems such as bundle adjustment:

Exploit the problem structure: Large-scale problems are almost always highly structured and bundle adjustment is no exception. In professional cartography and photogrammetric site-modelling, bundle problems with thousands of images and many tens of thousands of features are regularly solved. Such problems would simply be infeasible without a thorough exploitation of the natural structure and sparsity of the bundle problem. We will have much to say about sparsity below.

Use factorization effectively: Many of above formulae contain matrix inverses. This is a convenient short-hand for theoretical calculations, but *numerically, matrix inversion is almost never used*. Instead, the matrix is decomposed into its Cholesky, LU, QR, *etc*., factors and these are used directly, *e.g*. linear systems are solved using forwards and backwards substitution. This is much faster and numerically more accurate than explicit use of the inverse, particularly for sparse matrices such as the bundle Hessian, whose factors are still quite sparse, but whose inverse is always dense. Explicit inversion is required only occasionally, *e.g*. for covariance estimates, and even then only a few of the entries may be needed (*e.g*. diagonal blocks of the covariance). Factorization is the heart of the optimization iteration, where most of the time is spent and where most can be done to improve efficiency (by exploiting sparsity, symmetry and other problem structure) and numerical stability (by pivoting and scaling). Similarly, certain matrices (subspace projectors, Householder matrices) have (diagonal)+(low rank) forms which should not be explicitly evaluated as they can be applied more efficiently in pieces.

Use stable local parametrizations: As discussed in §2.2, the parametrization used for step prediction need not coincide with the global one used to store the state estimate. It is more important that it should be finite, uniform and locally as nearly linear as possible. If the global parametrization is in some way complex, highly nonlinear, or potentially ill-conditioned, it is usually preferable to use a stable local parametrization based on perturbations of the current state for step prediction.

Scaling and preconditioning: Another parametrization issue that has a profound and too-rarely recognized influence on numerical performance is **variable scaling** (the choice of 'units' or reference scale to use for each parameter), and more generally **preconditioning** (the choice of which linear combinations of parameters to use). These represent the linear part of the general parametrization problem. The performance of steepest descent and most other linearly convergent optimization methods is critically dependent on preconditioning, to the extent that for large problems, they are seldom practically useful without it.

One of the great advantages of the Newton-like methods is their theoretical independence of such scaling issues[9]. But even for these, scaling makes itself felt indirectly in several ways: (*i*) Step control strategies including convergence tests, maximum step size limitations, and damping strategies (trust region, Levenberg-Marquardt) are usually all based on some implicit norm $\|\delta \mathsf{x}\|^2$, and hence change under linear transformations of x (*e.g*., damping makes the step more like the non-invariant steepest descent one). (*ii*) Pivoting strategies for factoring H are highly dependent on variable scaling, as they choose 'large' elements on which to pivot. Here, 'large' *should* mean 'in which little numerical cancellation has occurred' but with uneven scaling it becomes 'with the largest scale'. (*iii*) The choice of gauge (datum, §9) may depend on variable scaling, and this can significantly influence convergence [82, 81].

For all of these reasons, it is important to choose variable scalings that relate meaningfully to the problem structure. This involves a judicious comparison of the relative influence of, *e.g*., a unit of error on a nearby point, a unit of error on a very distant one, a camera rotation error, a radial distortion

[9] Under a linear change of coordinates $\mathsf{x} \to \mathsf{T}\mathsf{x}$ we have $\mathsf{g} \to \mathsf{T}^{-\top}\mathsf{g}$ and $\mathsf{H} \to \mathsf{T}^{-\top}\mathsf{H}\mathsf{T}^{-1}$, so the Newton step $\delta\mathsf{x} = -\mathsf{H}^{-1}\mathsf{g}$ varies correctly as $\delta\mathsf{x} \to \mathsf{T}\,\delta\mathsf{x}$, whereas the gradient one $\delta\mathsf{x} \sim \mathsf{g}$ varies incorrectly as $\delta\mathsf{x} \to \mathsf{T}^{-\top}\,\delta\mathsf{x}$. The Newton and steepest descent steps agree only when $\mathsf{T}^\top\mathsf{T} = \mathsf{H}$.

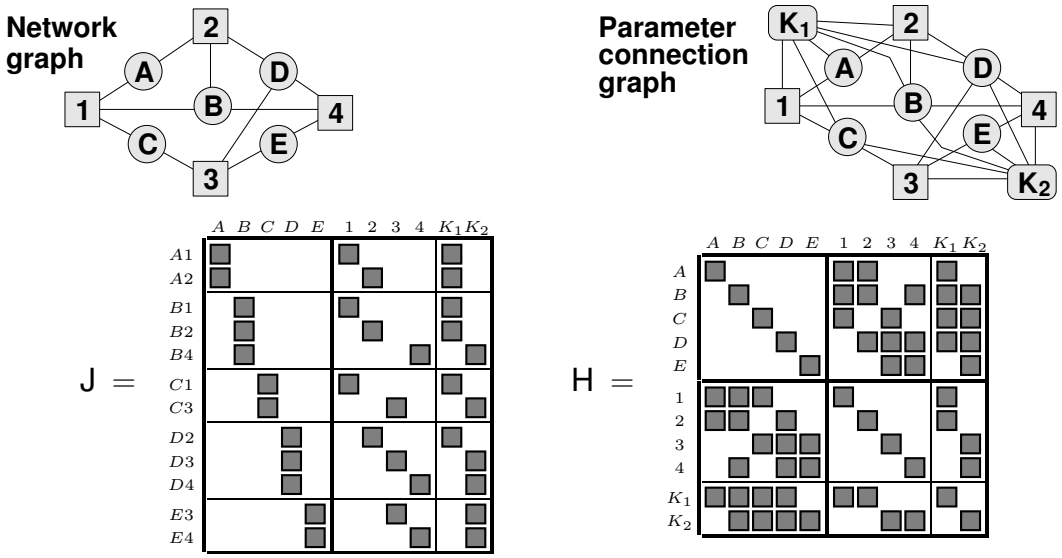


Figure 3: The network graph, parameter connection graph, Jacobian structure and Hessian structure for a toy bundle problem with five 3D features A–E, four images 1–4 and two camera calibrations K_1 (shared by images 1,2) and K_2 (shared by images 3,4). Feature A is seen in images 1,2; B in 1,2,4; C in 1,3; D in 2–4; and E in 3,4.

error, *etc*. For this, it is advisable to use an 'ideal' Hessian or weight matrix rather than the observed one, otherwise the scaling might break down if the Hessian happens to become ill-conditioned or non-positive during a few iterations before settling down.

5 Network Structure

Adjustment networks have a rich structure, illustrated in figure 3 for a toy bundle problem. The free parameters subdivide naturally into blocks corresponding to: 3D feature coordinates A,..., E; camera poses and unshared (single image) calibration parameters 1,..., 4; and calibration parameters shared across several images K_1, K_2. Parameter blocks interact only via their joint influence on image features and other observations, *i.e*. via their joint appearance in cost function contributions. The abstract structure of the measurement network can be characterized graphically by the **network graph** (top left), which shows which features are seen in which images, and the **parameter connection graph** (top right) which details the sparse structure by showing which parameter blocks have direct interactions. Blocks are linked if and only if they jointly influence at least one observation. The cost function Jacobian (bottom left) and Hessian (bottom right) reflect this sparse structure. The shaded boxes correspond to non-zero blocks of matrix entries. Each block of rows in the Jacobian corresponds to an observed image feature and contains contributions from each of the parameter blocks that influenced this observation. The Hessian contains an off-diagonal block for each edge of the parameter connection graph, *i.e*. for each pair of parameters that couple to at least one common feature / appear in at least one common cost contribution[10].

[10]The Jacobian structure can be described more directly by a bipartite graph whose nodes correspond on one side to the observations, and on the other to the parameter blocks that influence them. The parameter connection graph is then obtained by deleting each observation node and linking each pair of parameter nodes that it connects to. This is an example of elimination graph processing (see below).

Two layers of structure are visible in the Hessian. The **primary structure** consists of the subdivision into structure (A–E) and camera (1–4, K_1–K_2) submatrices. Note that the structure submatrix is block diagonal: 3D features couple only to cameras, not to other features. (This would no longer hold if inter-feature measurements such as distances or angles between points were present). The camera submatrix is often also block diagonal, but in this example the sharing of unknown calibration parameters produces off-diagonal blocks. The **secondary structure** is the internal sparsity pattern of the structure-camera Hessian submatrix. This is dense for small problems where all features are seen in all images, but in larger problems it often becomes quite sparse because each image only sees a fraction of the features.

All worthwhile bundle methods exploit at least the primary structure of the Hessian, and advanced methods exploit the secondary structure as well. The secondary structure is particularly sparse and regular in surface coverage problems such grids of photographs in aerial cartography. Such problems can be handled using a fixed 'nested dissection' variable reordering (§6.3). But for the more irregular connectivities of close range problems, general sparse factorization methods may be required to handle secondary structure.

Bundle problems are by no means limited to the above structures. For example, for more complex scene models with moving or articulated objects, there will be additional connections to object pose or joint angle nodes, with linkages reflecting the kinematic chain structure of the scene. It is often also necessary to add constraints to the adjustment, *e.g.* coplanarity of certain points. One of the greatest advantages of the bundle technique is its ability to adapt to almost arbitrarily complex scene, observation and constraint models.

6 Implementation Strategy 1: Second Order Adjustment Methods

The next three sections cover implementation strategies for optimizing the bundle adjustment cost function $\mathsf{f}(\mathsf{x})$ over the complete set of unknown structure and camera parameters x. This section is devoted to second-order Newton-style approaches, which are the basis of the great majority of current implementations. Their most notable characteristics are rapid (second order) asymptotic convergence but relatively high cost per iteration, with an emphasis on exploiting the network structure (the sparsity of the Hessian $\mathsf{H} = \frac{\mathsf{d}^2\mathsf{f}}{\mathsf{dx}^2}$) for efficiency. In fact, the optimization aspects are more or less standard (§4, [29,93,42]), so we will concentrate entirely on efficient methods for solving the linearized Newton step prediction equations $\delta\mathsf{x} = -\mathsf{H}^{-1}\mathsf{g}$, (6). For now, we will assume that the Hessian H is non-singular. This will be amended in §9 on gauge freedom, without changing the conclusions reached here.

6.1 The Schur Complement and the Reduced Bundle System

Schur complement: Consider the following block triangular matrix factorization:

$$\mathsf{M} = \begin{pmatrix} \mathsf{A} & \mathsf{B} \\ \mathsf{C} & \mathsf{D} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \mathsf{C}\mathsf{A}^{-1} & 1 \end{pmatrix} \begin{pmatrix} \mathsf{A} & 0 \\ 0 & \overline{\mathsf{D}} \end{pmatrix} \begin{pmatrix} 1 & \mathsf{A}^{-1}\mathsf{B} \\ 0 & 1 \end{pmatrix}, \qquad \overline{\mathsf{D}} \equiv \mathsf{D} - \mathsf{C}\mathsf{A}^{-1}\mathsf{B} \tag{16}$$

$$\begin{pmatrix} \mathsf{A} & \mathsf{B} \\ \mathsf{C} & \mathsf{D} \end{pmatrix}^{-1} = \begin{pmatrix} 1 & -\mathsf{A}^{-1}\mathsf{B} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mathsf{A}^{-1} & 0 \\ 0 & \overline{\mathsf{D}}^{-1} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\mathsf{C}\mathsf{A}^{-1} & 1 \end{pmatrix} = \begin{pmatrix} \mathsf{A}^{-1}+\mathsf{A}^{-1}\mathsf{B}\overline{\mathsf{D}}^{-1}\mathsf{C}\mathsf{A}^{-1} & -\mathsf{A}^{-1}\mathsf{B}\overline{\mathsf{D}}^{-1} \\ -\overline{\mathsf{D}}^{-1}\mathsf{C}\mathsf{A}^{-1} & \overline{\mathsf{D}}^{-1} \end{pmatrix} \tag{17}$$

Here A must be square and invertible, and for (17), the whole matrix must also be square and invertible. $\overline{\mathsf{D}}$ is called the **Schur complement** of A in M. If both A and D are invertible, complementing on

D rather than A gives

$$\begin{pmatrix} \mathsf{A} & \mathsf{B} \\ \mathsf{C} & \mathsf{D} \end{pmatrix}^{-1} = \begin{pmatrix} \overline{\mathsf{A}}^{-1} & -\overline{\mathsf{A}}^{-1}\mathsf{B}\mathsf{D}^{-1} \\ -\mathsf{D}\mathsf{C}\overline{\mathsf{A}}^{-1} & \mathsf{D}^{-1}+\mathsf{D}^{-1}\mathsf{C}\overline{\mathsf{A}}^{-1}\mathsf{B}\mathsf{D}^{-1} \end{pmatrix}, \qquad \overline{\mathsf{A}} = \mathsf{A} - \mathsf{B}\mathsf{D}^{-1}\mathsf{C}$$

Equating upper left blocks gives the **Woodbury formula**:

$$(\mathsf{A} \pm \mathsf{B}\mathsf{D}^{-1}\mathsf{C})^{-1} = \mathsf{A}^{-1} \mp \mathsf{A}^{-1}\mathsf{B}\,(\mathsf{D} \pm \mathsf{C}\mathsf{A}^{-1}\mathsf{B})^{-1}\,\mathsf{C}\mathsf{A}^{-1} \tag{18}$$

This is the usual method of updating the inverse of a nonsingular matrix A after an update (especially a low rank one) $\mathsf{A} \rightarrow \mathsf{A} \pm \mathsf{B}\mathsf{D}^{-1}\mathsf{C}$. (See §8.1).

Reduction: Now consider the linear system $\left(\begin{smallmatrix} \mathsf{A} & \mathsf{B} \\ \mathsf{C} & \mathsf{D} \end{smallmatrix}\right)\left(\begin{smallmatrix} \mathsf{x}_1 \\ \mathsf{x}_2 \end{smallmatrix}\right) = \left(\begin{smallmatrix} \mathsf{b}_1 \\ \mathsf{b}_2 \end{smallmatrix}\right)$. Pre-multiplying by $\left(\begin{smallmatrix} 1 & 0 \\ -\mathsf{C}\mathsf{A}^{-1} & 1 \end{smallmatrix}\right)$ gives $\left(\begin{smallmatrix} \mathsf{A} & \mathsf{B} \\ 0 & \overline{\mathsf{D}} \end{smallmatrix}\right)\left(\begin{smallmatrix} \mathsf{x}_1 \\ \mathsf{x}_2 \end{smallmatrix}\right) = \begin{pmatrix} \mathsf{b}_1 \\ \overline{\mathsf{b}}_2 \end{pmatrix}$ where $\overline{\mathsf{b}}_2 \equiv \mathsf{b}_2 - \mathsf{C}\mathsf{A}^{-1}\mathsf{b}_1$. Hence we can use Schur complement and forward substitution to find a **reduced system** $\overline{\mathsf{D}}\,\mathsf{x}_2 = \overline{\mathsf{b}}_2$, solve this for x_2, then back-substitute and solve to find x_1:

$$\begin{array}{lll} \overline{\mathsf{D}} \equiv \mathsf{D} - \mathsf{C}\mathsf{A}^{-1}\mathsf{B} & & \\ \overline{\mathsf{b}}_2 \equiv \mathsf{b}_2 - \mathsf{C}\mathsf{A}^{-1}\mathsf{b}_1 & \overline{\mathsf{D}}\,\mathsf{x}_2 = \overline{\mathsf{b}}_2 & \mathsf{A}\,\mathsf{x}_1 = \mathsf{b}_1 - \mathsf{B}\,\mathsf{x}_2 \\ \text{Schur complement + forward substitution} & \text{reduced system} & \text{back-substitution} \end{array} \tag{19}$$

Note that the reduced system entirely subsumes the contribution of the x_1 rows and columns to the network. Once we have reduced, we can pretend that the problem does not involve x_1 at all — it can be found later by back-substitution if needed, or ignored if not. This is the basis of all recursive filtering methods. In bundle adjustment, if we use the primary subdivision into feature and camera variables and subsume the structure ones, we get the **reduced camera system** $\overline{\mathsf{H}}_{CC}\,\mathsf{x}_C = \overline{\mathsf{g}}_C$, where:

$$\begin{aligned} \overline{\mathsf{H}}_{CC} &\equiv \mathsf{H}_{CC} - \mathsf{H}_{CS}\,\mathsf{H}_{SS}^{-1}\,\mathsf{H}_{SC} = \mathsf{H}_{CC} - \textstyle\sum_p \mathsf{H}_{Cp}\,\mathsf{H}_{pp}^{-1}\,\mathsf{H}_{pC} \\ \overline{\mathsf{g}}_C &\equiv \mathsf{g}_C - \mathsf{H}_{CS}\,\mathsf{H}_{SS}^{-1}\,\mathsf{g}_S = \mathsf{g}_C - \textstyle\sum_p \mathsf{H}_{Cp}\,\mathsf{H}_{pp}^{-1}\,\mathsf{g}_p \end{aligned} \tag{20}$$

Here, 'S' selects the structure block and 'C' the camera one. H_{SS} is block diagonal, so the reduction can be calculated rapidly by a sum of contributions from the individual 3D features 'p' in S. Brown's original 1958 method for bundle adjustment [16, 19, 100] was based on finding the reduced camera system as above, and solving it using Gaussian elimination. Profile Cholesky decomposition (§B.3) offers a more streamlined method of achieving this.

Occasionally, long image sequences have more camera parameters than structure ones. In this case it is more efficient to reduce the camera parameters, leaving a **reduced structure system**.

6.2 Triangular decompositions

If D in (16) is further subdivided into blocks, the factorization process can be continued recursively. In fact, there is a family of block (lower triangular)*(diagonal)*(upper triangular) factorizations $\mathsf{A} = \mathsf{L}\,\mathsf{D}\,\mathsf{U}$:

$$\begin{pmatrix} \mathsf{A}_{11} & \mathsf{A}_{12} & \cdots & \mathsf{A}_{1n} \\ \mathsf{A}_{21} & \mathsf{A}_{22} & \cdots & \mathsf{A}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathsf{A}_{m1} & \mathsf{A}_{m2} & \cdots & \mathsf{A}_{mn} \end{pmatrix} = \begin{pmatrix} \mathsf{L}_{11} & & & \\ \mathsf{L}_{21} & \mathsf{L}_{22} & & \\ \vdots & \vdots & \ddots & \\ \vdots & \vdots & & \vdots \\ \mathsf{L}_{m1} & \mathsf{L}_{m2} & \cdots & \mathsf{L}_{mr} \end{pmatrix} \begin{pmatrix} \mathsf{D}_1 & & & \\ & \mathsf{D}_2 & & \\ & & \ddots & \\ & & & \mathsf{D}_r \end{pmatrix} \begin{pmatrix} \mathsf{U}_{11} & \mathsf{U}_{12} & \cdots & \cdots & \mathsf{U}_{1n} \\ & \mathsf{U}_{22} & \cdots & \cdots & \mathsf{U}_{2n} \\ & & \ddots & & \vdots \\ & & & \cdots & \mathsf{U}_{rn} \end{pmatrix} \tag{21}$$

See §B.1 for computational details. The main advantage of triangular factorizations is that they make linear algebra computations with the matrix much easier. In particular, if the input matrix A is square and nonsingular, linear equations $\mathsf{A}\,\mathsf{x} = \mathsf{b}$ can be solved by a sequence of three recursions that implicitly implement multiplication by $\mathsf{A}^{-1} = \mathsf{U}^{-1}\,\mathsf{D}^{-1}\,\mathsf{L}^{-1}$:

$$\mathsf{L}\,\mathsf{c} = \mathsf{b} \qquad \mathsf{c}_i \leftarrow \mathsf{L}_{ii}^{-1}\left(\mathsf{b}_i - \textstyle\sum_{j<i} \mathsf{L}_{ij}\,\mathsf{c}_j\right) \qquad \text{forward substitution} \qquad (22)$$

$$\mathsf{D}\,\mathsf{d} = \mathsf{c} \qquad \mathsf{d}_i \leftarrow \mathsf{D}_i^{-1}\,\mathsf{c}_i \qquad \text{diagonal solution} \qquad (23)$$

$$\mathsf{U}\,\mathsf{x} = \mathsf{d} \qquad \mathsf{x}_i \leftarrow \mathsf{U}_{ii}^{-1}\left(\mathsf{d}_i - \textstyle\sum_{j>i} \mathsf{U}_{ij}\,\mathsf{x}_j\right) \qquad \text{back-substitution} \qquad (24)$$

Forward substitution corrects for the influence of earlier variables on later ones, diagonal solution solves the transformed system, and back-substitution propagates corrections due to later variables back to earlier ones. In practice, this is usual method of solving linear equations such as the Newton step prediction equations. It is stabler and much faster than explicitly inverting A and multiplying by A^{-1}.

The diagonal blocks $\mathsf{L}_{ii}, \mathsf{D}_i, \mathsf{U}_{ii}$ can be set arbitrarily provided that the product $\mathsf{L}_{ii}\,\mathsf{D}_i\,\mathsf{U}_{ii}$ remains constant. This gives a number of well-known factorizations, each optimized for a different class of matrices. **Pivoting** (row and/or column exchanges designed to improve the conditioning of L and/or U, §B.1) is also necessary in most cases, to ensure stability. Choosing $\mathsf{L}_{ii} = \mathsf{D}_{ii} = \mathsf{1}$ gives the (block) **LU decomposition** $\mathsf{A} = \mathsf{L}\,\mathsf{U}$, the matrix representation of (block) Gaussian elimination. Pivoted by rows, this is the standard method for non-symmetric matrices. For symmetric A, roughly half of the work of factorization can be saved by using a symmetry-preserving $\mathrm{LDL}^\top$ factorization, for which D is symmetric and $\mathsf{U} = \mathsf{L}^\top$. The pivoting strategy must also preserve symmetry in this case, so it has to permute columns in the same way as the corresponding rows. If A is symmetric positive definite we can further set $\mathsf{D} = \mathsf{1}$ to get the **Cholesky decomposition** $\mathsf{A} = \mathsf{L}\,\mathsf{L}^\top$. This is stable even without pivoting, and hence extremely simple to implement. It is the standard decomposition method for almost all unconstrained optimization problems including bundle adjustment, as the Hessian is positive definite near a non-degenerate cost minimum (and in the Gauss-Newton approximation, almost everywhere else, too). If A is symmetric but only positive *semi*definite, **diagonally pivoted Cholesky decomposition** can be used. This is the case, *e.g.* in subset selection methods of gauge fixing (§9.5). Finally, if A is symmetric but indefinite, it is not possible to reduce D stably to $\mathsf{1}$. Instead, the **Bunch-Kaufman method** is used. This is a diagonally pivoted $\mathrm{LDL}^\top$ method, where D has a mixture of 1×1 and 2×2 diagonal blocks. The augmented Hessian $\left(\begin{smallmatrix} \mathsf{H} & \mathsf{C} \\ \mathsf{C}^\top & \mathsf{0} \end{smallmatrix}\right)$ of the Lagrange multiplier method for constrained optimization problems (12) is always symmetric indefinite, so Bunch-Kaufman is the recommended method for solving constrained bundle problems. (It is something like 40% faster than Gaussian elimination, and about equally stable).

Another use of factorization is matrix inversion. Inverses can be calculated by factoring, inverting each triangular factor by forwards or backwards substitution (53), and multiplying out: $\mathsf{A}^{-1} = \mathsf{U}^{-1}\,\mathsf{D}^{-1}\,\mathsf{L}^{-1}$. However, explicit inverses are rarely used in numerical analysis, it being both stabler and much faster in almost all cases to leave them implicit and work by forward/backward substitution w.r.t. a factorization, rather than multiplication by the inverse. One place where inversion *is* needed in its own right, is to calculate the dispersion matrix (inverse Hessian, which asymptotically gives the posterior covariance) as a measure of the likely variability of parameter estimates. The dispersion can be calculated by explicit inversion of the factored Hessian, but often only a few of its entries are needed, *e.g.* the diagonal blocks and a few key off-diagonal parameter covariances. In this case (54) can be used, which efficiently calculates the covariance entries corresponding to just the nonzero elements of $\mathsf{L}, \mathsf{D}, \mathsf{U}$.

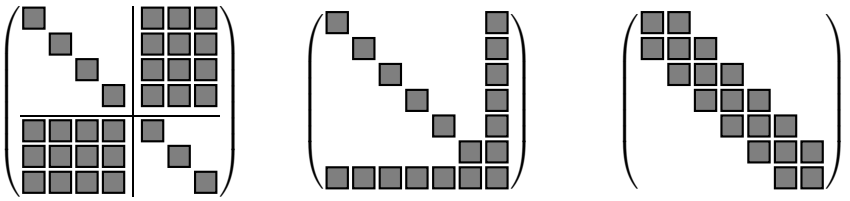
6.3 Sparse factorization

To apply the above decompositions to sparse matrices, we must obviously avoid storing and manipulating the zero blocks. But there is more to the subject than this. As a sparse matrix is decomposed, zero positions tend to rapidly **fill in** (become non-zero), essentially because decomposition is based on repeated linear combination of matrix rows, which is generically non-zero wherever any one of its inputs is. Fill-in depends strongly on the order in which variables are eliminated, so efficient sparse factorization routines attempt to minimize either operation counts or fill-in by re-ordering the variables. (The Schur process is fixed in advance, so this is the only available freedom). Globally minimizing either operations or fill-in is NP complete, but reasonably good heuristics exist (see below). Variable order affects stability (pivoting) as well as speed, and these two goals conflict to some extent. Finding heuristics that work well on both counts is still a research problem.

Algorithmically, fill-in is characterized by an **elimination graph** derived from the parameter coupling / Hessian graph [40, 26, 11]. To create this, nodes (blocks of parameters) are visited in the given elimination ordering, at each step linking together all unvisited nodes that are currently linked to the current node. The coupling of block i to block j via visited block k corresponds to a non-zero Schur contribution $\mathsf{L}_{ik}\,\mathsf{D}_k^{-1}\,\mathsf{U}_{kj}$, and at each stage the subgraph on the currently unvisited nodes is the coupling graph of the current reduced Hessian. The amount of fill-in is the number of new graph edges created in this process.

6.3.1 Pattern matrices

We seek variable orderings that approximately minimize the total operation count or fill-in over the whole elimination chain. For many problems a suitable ordering can be fixed in advance, typically giving one of a few standard pattern matrices such as band or arrowhead matrices, perhaps with such structure at several levels.



bundle Hessian arrowhead matrix block tridiagonal matrix (25)

The most prominent pattern structure in bundle adjustment is the primary subdivision of the Hessian into structure and camera blocks. To get the reduced camera system (19), we treat the Hessian as an arrowhead matrix with a broad final column containing all of the camera parameters. Arrowhead matrices are trivial to factor or reduce by block 2×2 Schur complementation, *c.f.* (16, 19). For bundle problems with many independent images and only a few features, one can also complement on the image parameter block to get a reduced *structure* system.

Another very common pattern structure is the block tridiagonal one which characterizes all singly coupled chains (sequences of images with only pairwise overlap, Kalman filtering and other time recursions, simple kinematic chains). Tridiagonal matrices are factored or reduced by recursive block 2×2 Schur complementation starting from one end. The L and U factors are also block tridiagonal, but the inverse is generally dense.

Pattern orderings are often very natural but it is unwise to think of them as immutable: structure often occurs at several levels and deeper structure or simply changes in the relative sizes of the various parameter classes may make alternative orderings preferable. For more difficult problems there are

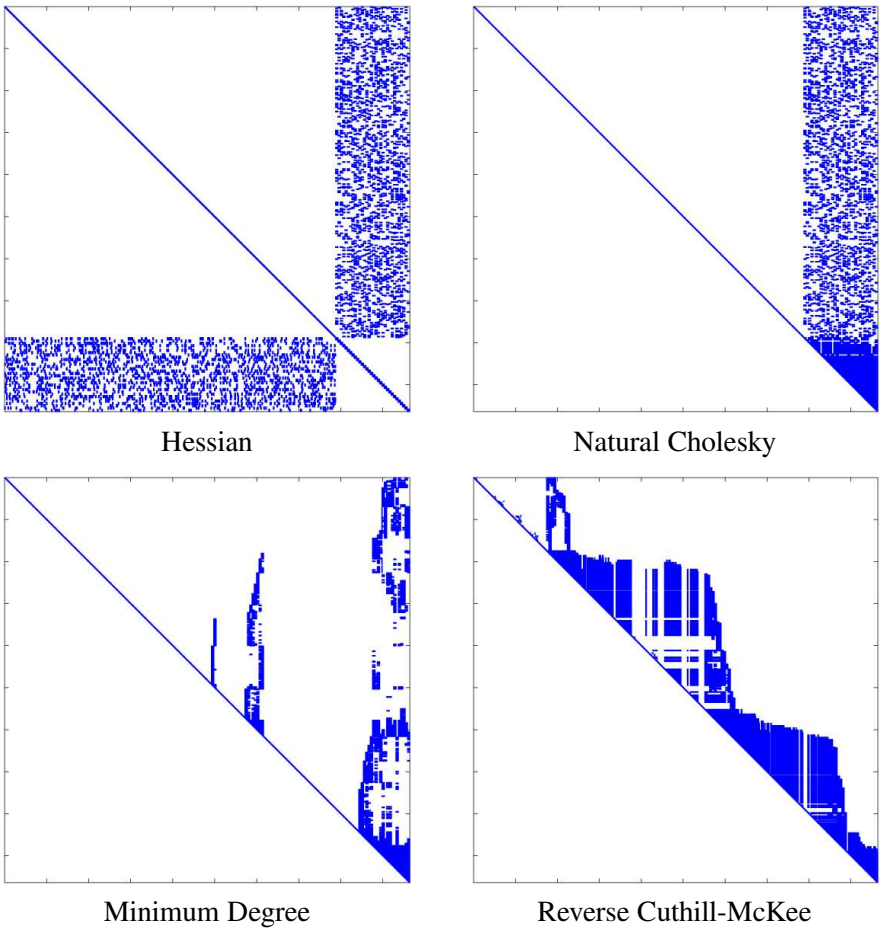


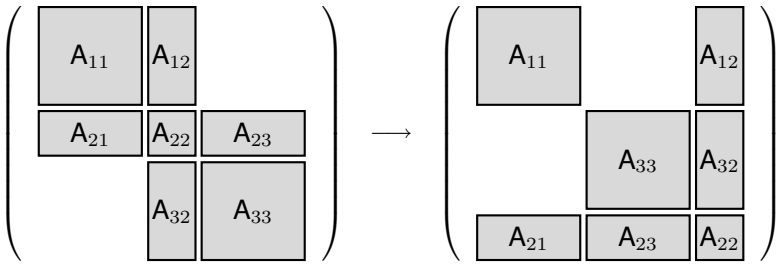
Figure 4: A bundle Hessian for an irregular coverage problem with only local connections, and its Cholesky factor in natural (structure-then-camera), minimum degree, and reverse Cuthill-McKee ordering.

two basic classes of on-line ordering strategies. *Bottom-up* methods try to minimize fill-in locally and greedily at each step, at the risk of global short-sightedness. *Top-down* methods take a divide-and-conquer approach, recursively splitting the problem into smaller sub-problems which are solved quasi-independently and later merged.

6.3.2 Top-down Ordering Methods

The most common top-down method is called **nested dissection** or **recursive partitioning** [64, 57, 19, 38, 40, 11]. The basic idea is to recursively split the factorization problem into smaller sub-problems, solve these independently, and then glue the solutions together along their common boundaries. Splitting involves choosing a **separating set** of variables, whose deletion will separate the remaining variables into two or more independent subsets. This corresponds to finding a **(vertex) graph cut** of the elimination graph, *i.e.* a set of vertices whose deletion will split it into two or more disconnected components. Given such a partitioning, the variables are reordered into connected components, with

the separating set ones last. This produces an 'arrowhead' matrix, *e.g.* :



(26)

The arrowhead matrix is factored by blocks, as in reduction or profile Cholesky, taking account of any internal sparsity in the diagonal blocks and the borders. Any suitable factorization method can be used for the diagonal blocks, including further recursive partitionings.

Nested dissection is most useful when comparatively small separating sets can be found. A trivial example is the primary structure of the bundle problem: the camera variables separate the 3D structure into independent features, giving the standard arrowhead form of the bundle Hessian. More interestingly, networks with good geometric or temporal locality (surface- and site-covering networks, video sequences) tend to have small separating sets based on spatial or temporal subdivision. The classic examples are geodesic and aerial cartography networks with their local 2D connections — spatial bisection gives simple and very efficient recursive decompositions for these [64,57,19].

For sparse problems with less regular structure, one can use graph partitioning algorithms to find small separating sets. Finding a globally minimal partition sequence is NP complete but several effective heuristics exist. This is currently an active research field. One promising family are multilevel schemes [70,71,65,4] which decimate (subsample) the graph, partition using *e.g.* a spectral method, then refine the result to the original graph. (These algorithms should also be very well-suited to graph based visual segmentation and matching).

6.3.3 Bottom-up Ordering Methods

Many bottom-up variable ordering heuristics exist. Probably the most widespread and effective is **minimum degree ordering**. At each step, this eliminates the variable coupled to the fewest remaining ones (*i.e.* the elimination graph node with the fewest unvisited neighbours), so it minimizes the number $\mathcal{O}(n^2_{\text{neighbours}})$ of changed matrix elements and hence FLOPs for the step. The minimum degree ordering can also be computed quite rapidly without explicit graph chasing. A related ordering, **minimum deficiency**, minimizes the fill-in (newly created edges) at each step, but this is considerably slower to calculate and not usually so effective.

Fill-in or operation minimizing strategies tend to produce somewhat fragmentary matrices that require pointer- or index-based sparse matrix implementations (see fig. 4). This increases complexity and tends to reduce cache locality and pipeline-ability. An alternative is to use **profile matrices** which (for lower triangles) store all elements in each row between the first non-zero one and the diagonal in a contiguous block. This is easy to implement (see §B.3), and practically efficient so long as about 30% or more of the stored elements are actually non-zero. Orderings for this case aim to minimize the sum of the profile lengths rather than the number of non-zero elements. Profiling enforces a multiply-linked chain structure on the variables, so it is especially successful for linear / chain-like / one dimensional problems, *e.g.* space or time sequences. The simplest profiling strategy is **reverse Cuthill-McKee** which chooses some initial variable (very preferably one from one 'end' of the chain), adds all variables coupled to that, then all variables coupled to those, *etc.*, then reverses the

ordering (otherwise, any highly-coupled variables get eliminated early on, which causes disastrous fill-in). More sophisticated are the so-called **banker's strategies**, which maintain an active set of all the variables coupled to the already-eliminated ones, and choose the next variable — from the active set (King [72]), it and its neighbours (Snay [101]) or all uneliminated variables (Levy [75]) — to minimize the new size of the active set at each step. In particular, **Snay's banker's algorithm** is reported to perform well on geodesy and aerial cartography problems [101, 24].

For all of these automatic ordering methods, it often pays to do some of the initial work by hand, *e.g.* it might be appropriate to enforce the structure / camera division beforehand and only order the reduced camera system. If there are nodes of particularly high degree such as inner gauge constraints, the ordering calculation will usually run faster and the quality may also be improved by removing these from the graph and placing them last by hand.

The above ordering methods apply to both Cholesky / $\mathsf{LDL}^\top$ decomposition of the Hessian and QR decomposition of the least squares Jacobian. Sparse QR methods can be implemented either with Givens rotations or (more efficiently) with sparse Householder transformations. Row ordering is important for the Givens methods [39]. For Householder ones (and some Givens ones too) the **multifrontal** organization is now usual [41, 11], as it captures the natural parallelism of the problem.

7 Implementation Strategy 2: First Order Adjustment Methods

We have seen that for large problems, factoring the Hessian H to compute the Newton step can be both expensive and (if done efficiently) rather complex. In this section we consider alternative methods that avoid the cost of exact factorization. As the Newton step can not be calculated, such methods generally only achieve first order (linear) asymptotic convergence: when close to the final state estimate, the error is asymptotically reduced by a constant (and in practice often depressingly small) factor at each step, whereas quadratically convergent Newton methods roughly double the number of significant digits at each step. So first order methods require more iterations than second order ones, but each iteration is usually much cheaper. The relative efficiency depends on the relative sizes of these two effects, both of which can be substantial. For large problems, the reduction in work per iteration is usually at least $\mathcal{O}(n)$, where n is the problem size. But whereas Newton methods converge from $\mathcal{O}(1)$ to $\mathcal{O}\big(10^{-16}\big)$ in about $1 + \log_2 16 = 5$ iterations, linearly convergent ones take respectively $\log 10^{-16} / \log(1-\gamma) = 16, 350, 3700$ iterations for reduction $\gamma = 0.9, 0.1, 0.01$ per iteration. Unfortunately, reductions of only 1% or less are by no means unusual in practice (§7.2), and the reduction tends to decrease as n increases.

7.1 First Order Iterations

We first consider a number of common first order methods, before returning to the question of why they are often slow.

Steepest descent: The simplest first order method is **steepest descent** or **gradient descent**. It "slides down the gradient" by taking $\delta\mathsf{x} \sim \mathsf{g}$ or $\mathsf{H}_a = \mathsf{1}$. Line search is needed, to find an appropriate scale for the step. Even with exact line search (*i.e.* the minimum along the line is found exactly), steepest descent is spectacularly inefficient for most problems (see §7.2), unless the Hessian actually happens to be very close to a multiple of $\mathsf{1}$. This can be arranged by preconditioning with a linear transform L, $\mathsf{x} \to \mathsf{L}\,\mathsf{x}$, $\mathsf{g} \to \mathsf{L}^{-\top}\mathsf{g}$ and $\mathsf{H} \to \mathsf{L}^{-\top}\mathsf{H}\,\mathsf{L}^{-1}$, where $\mathsf{L}\,\mathsf{L}^\top \sim \mathsf{H}$ is an approximate Cholesky factor (or other left square root) of H, so that $\mathsf{H} \to \mathsf{L}^{-\top}\mathsf{H}\,\mathsf{L}^{-1} \sim \mathsf{1}$. In this very special case, preconditioned steepest descent approximates the Newton method. Strictly speaking, "steepest" descent is a cheat.

The gradient is a covector (linear form on vectors) rather than a vector. It does not define a preferred direction in search space, but merely gives the rate of descent along any given gradient vector. The sensitivity of steepest descent to the coordinate system is one symptom of this.

Alternation: Another simple approach is **alternation**: partition the variables into groups and cycle through the groups optimizing over each in turn, with the other groups held fixed. This is most appropriate when the subproblems are significantly easier to optimize than the full one. A natural and often-rediscovered alternation for the bundle problem is **resection-intersection**, which interleaves steps of *resection* (finding the camera poses and if necessary calibrations from fixed 3D features) and *intersection* (finding the 3D features from fixed camera poses and calibrations). The subproblems for individual features and cameras are independent, so only the diagonal blocks of H are required.

Alternation can be used in several ways. One extreme is to optimize (or perhaps only perform one step of optimization) over each group in turn, with a state update and re-evaluation of (the relevant components of) g, H after each group. Alternatively, some of the re-evaluations can be simulated by evaluating the linearized effects of the parameter group update on the other groups. *E.g.*, for resection-intersection with structure update $\boldsymbol{\delta}\mathsf{x}_S = -\mathsf{H}_{SS}\,\mathsf{g}_S(\mathsf{x}_S, \mathsf{x}_C)$ (where 'S' selects the structure variables and 'C' the camera ones), the updated camera gradient is exactly the gradient of the reduced camera system, $\mathsf{g}_C(\mathsf{x}_S + \boldsymbol{\delta}\mathsf{x}_S, \mathsf{x}_C) \approx \mathsf{g}_C(\mathsf{x}_S, \mathsf{x}_C) + \mathsf{H}_{CS}\boldsymbol{\delta}\mathsf{x}_S = \mathsf{g}_C - \mathsf{H}_{CS}\,\mathsf{H}_{SS}^{-1}\,\mathsf{g}_C$. So the total update for the cycle is $\begin{pmatrix} \boldsymbol{\delta}\mathsf{x}_S \\ \boldsymbol{\delta}\mathsf{x}_C \end{pmatrix} = -\begin{pmatrix} \mathsf{H}_{SS}^{-1} & 0 \\ -\mathsf{H}_{CC}^{-1}\,\mathsf{H}_{CS}\,\mathsf{H}_{SS}^{-1} & \mathsf{H}_{CC}^{-1} \end{pmatrix}\begin{pmatrix} \mathsf{g}_S \\ \mathsf{g}_C \end{pmatrix} = \begin{pmatrix} \mathsf{H}_{SS} & 0 \\ \mathsf{H}_{CS} & \mathsf{H}_{CC} \end{pmatrix}^{-1}\begin{pmatrix} \mathsf{g}_S \\ \mathsf{g}_C \end{pmatrix}$. In general, this correction propagation amounts to solving the system as if the above-diagonal triangle of H were zero. Once we have cycled through the variables, we can update the full state and relinearize. This is the **nonlinear Gauss-Seidel method**. Alternatively, we can split the above-diagonal triangle of H off as a correction (back-propagation) term and continue iterating $\begin{pmatrix} \mathsf{H}_{SS} & 0 \\ \mathsf{H}_{CS} & \mathsf{H}_{CC} \end{pmatrix}\begin{pmatrix} \boldsymbol{\delta}\mathsf{x}_S \\ \boldsymbol{\delta}\mathsf{x}_C \end{pmatrix}_{(k)} = -\begin{pmatrix} \mathsf{g}_S \\ \mathsf{g}_C \end{pmatrix} - \begin{pmatrix} 0 & \mathsf{H}_{SC} \\ 0 & 0 \end{pmatrix}\begin{pmatrix} \boldsymbol{\delta}\mathsf{x}_S \\ \boldsymbol{\delta}\mathsf{x}_C \end{pmatrix}_{(k-1)}$ until (hopefully) $\begin{pmatrix} \boldsymbol{\delta}\mathsf{x}_S \\ \boldsymbol{\delta}\mathsf{x}_C \end{pmatrix}$ converges to the full Newton step $\boldsymbol{\delta}\mathsf{x} = -\mathsf{H}^{-1}\mathsf{g}$. This is the **linear Gauss-Seidel method** applied to solving the Newton step prediction equations. Finally, alternation methods always tend to underestimate the size of the Newton step because they fail to account for the cost-reducing effects of including the back-substitution terms. **Successive Over-Relaxation (SOR)** methods improve the convergence rate by artificially lengthening the update steps by a heuristic factor $1 < \gamma < 2$.

Most if not all of the above alternations have been applied to both the bundle problem and the independent model one many times, *e.g.* [19,95,2,108,91,20]. Brown considered the relatively sophisticated SOR method for aerial cartography problems as early as 1964, before developing his recursive decomposition method [19]. None of these alternations are very effective for traditional large-scale problems, although §7.4 below shows that they can sometimes compete for smaller highly connected ones.

Krylov subspace methods: Another large family of iterative techniques are the **Krylov subspace methods**, based on the remarkable properties of the power subspaces $\text{Span}(\{\mathsf{A}^k\, b | k = 0 \ldots n\})$ for fixed A, b as n increases. Krylov iterations predominate in many large-scale linear algebra applications, including linear equation solving.

The earliest and greatest Krylov method is the **conjugate gradient** iteration for solving a positive definite linear system or optimizing a quadratic cost function. By augmenting the steepest descent step with a carefully chosen multiple of the previous step, this manages to minimize the quadratic model function over the entire k^{th} Krylov subspace at the k^{th} iteration, and hence (in exact arithmetic) over the whole space at the n_{x}^{th} one. This no longer holds when there is round-off error, but $\mathcal{O}(n_{\mathsf{x}})$ iterations usually still suffice to find the Newton step. Each iteration is $\mathcal{O}\left(n_{\mathsf{x}}^2\right)$ so this is not in itself a large gain over explicit factorization. However convergence is significantly faster if the

eigenvalues of H are tightly clustered away from zero: if the eigenvalues are covered by intervals $[a_i, b_i]_{i=1\ldots k}$, convergence occurs in $\mathcal{O}\left(\sum_{i=1}^{k} \sqrt{b_i/a_i}\right)$ iterations [99,47,48][11]. Preconditioning (see below) aims at achieving such clustering. As with alternation methods, there is a range of possible update / re-linearization choices, ranging from a fully nonlinear method that relinearizes after each step, to solving the Newton equations exactly using many linear iterations. One major advantage of conjugate gradient is its simplicity: there is no factorization, all that is needed is multiplication by H. For the full nonlinear method, H is not even needed — one simply makes a line search to find the cost minimum along the direction defined by g and the previous step.

One disadvantage of nonlinear conjugate gradient is its high sensitivity to the accuracy of the line search. Achieving the required accuracy may waste several function evaluations at each step. One way to avoid this is to make the information obtained by the conjugation process more explicit by building up an explicit approximation to H or H^{-1}. **Quasi-Newton** methods such as the BFGS method do this, and hence need less accurate line searches. The quasi-Newton approximation to H or H^{-1} is dense and hence expensive to store and manipulate, but **Limited Memory Quasi-Newton (LMQN)** methods often get much of the desired effect by maintaining only a low-rank approximation.

There are variants of all of these methods for least squares (Jacobian rather than Hessian based) and for constrained problems (non-positive definite matrices).

7.2 Why Are First Order Methods Slow?

To understand why first order methods often have slow convergence, consider the effect of approximating the Hessian in Newton's method. Suppose that in some local parametrization x centred at a cost minimum $\mathsf{x} = \mathsf{0}$, the cost function is well approximated by a quadratic near $\mathsf{0}$: $\mathsf{f}(\mathsf{x}) \approx \frac{1}{2}\mathsf{x}^\top \mathsf{H}\,\mathsf{x}$ and hence $\mathsf{g}(\mathsf{x}) \equiv \mathsf{H}\,\mathsf{x}$, where H is the true Hessian. For most first order methods, the predicted step is linear in the gradient g. If we adopt a Newton-like state update $\delta\mathsf{x} = -\mathsf{H}_a^{-1}\mathsf{g}(\mathsf{x})$ based on some approximation H_a to H, we get an iteration:

$$\mathsf{x}_{k+1} \;=\; \mathsf{x}_k - \mathsf{H}_a^{-1}\mathsf{g}(\mathsf{x}_k) \;\approx\; (\mathsf{1} - \mathsf{H}_a^{-1}\mathsf{H})\;\mathsf{x}_k \;\approx\; (\mathsf{1} - \mathsf{H}_a^{-1}\mathsf{H})^{k+1}\;\mathsf{x}_0 \tag{27}$$

The numerical behaviour is determined by projecting x_0 along the eigenvectors of $\mathsf{1} - \mathsf{H}_a^{-1}\mathsf{H}$. The components corresponding to large-modulus eigenvalues decay slowly and hence asymptotically dominate the residual error. For generic x_0, the method converges 'linearly' (*i.e.* exponentially) at rate $\|\mathsf{1} - \mathsf{H}_a^{-1}\mathsf{H}\|_2$, or diverges if this is greater than one. (Of course, the exact Newton step $\delta\mathsf{x} = -\mathsf{H}^{-1}\mathsf{g}$ converges in a single iteration, as $\mathsf{H}_a = \mathsf{H}$). Along eigen-directions corresponding to positive eigenvalues (for which H_a overestimates H), the iteration is over-damped and convergence is slow but monotonic. Conversely, along directions corresponding to negative eigenvalues (for which H_a underestimates H), the iteration is under-damped and zigzags towards the solution. If H is underestimated by a factor greater than two along any direction, there is divergence. Figure 5 shows an example of the typical asymptotic behaviour of first and second order methods in a small bundle problem.

Ignoring the camera-feature coupling: As an example, many approximate bundle methods ignore or approximate the off-diagonal feature-camera blocks of the Hessian. This amounts to ignoring the fact that the cost of a feature displacement can be partially offset by a compensatory camera displacement and vice versa. It therefore significantly over-estimates the total 'stiffness' of the network, particularly for large, loosely connected networks. The fact that off-diagonal blocks are *not* negligible compared to the diagonal ones can be seen in several ways:

[11] For other eigenvalue based based analyses of the bundle adjustment covariance, see [103,92].

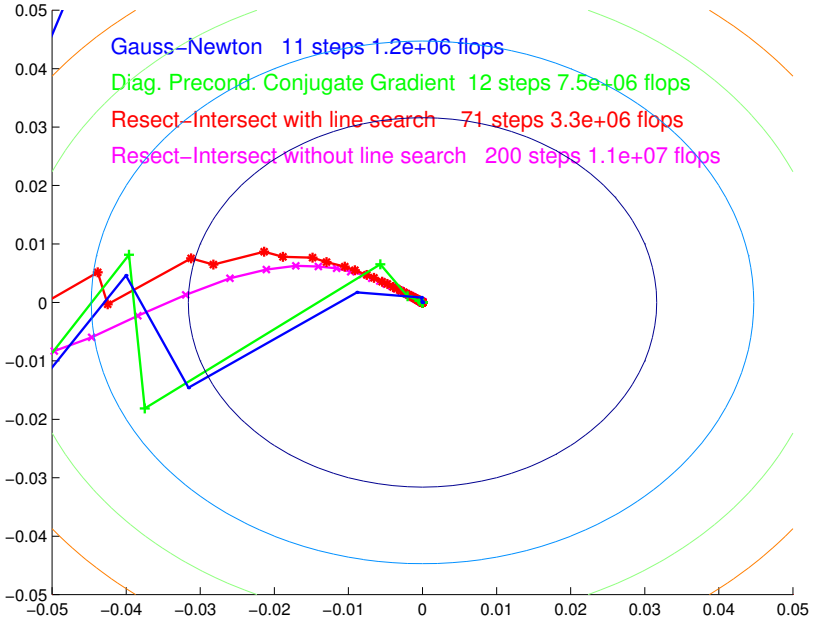


Figure 5: An example of the typical behaviour of first and second order convergent methods near the minimum. This is a 2D projection of a small but ill-conditioned bundle problem along the two most variable directions. The second order methods converge quite rapidly, whether they use exact (Gauss-Newton) or iterative (diagonally preconditioned conjugate gradient) linear solver for the Newton equations. In contrast, first order methods such as resection-intersection converge slowly near the minimum owing to their inaccurate model of the Hessian. The effects of mismodelling can be reduced to some extent by adding a line search.

- Looking forward to §9, before the gauge is fixed, the full Hessian is singular owing to gauge freedom. The diagonal blocks by themselves are well-conditioned, but including the off-diagonal ones entirely cancels this along the gauge orbit directions. Although gauge fixing removes the resulting singularity, it can not change the fact that the off-diagonal blocks have enough weight to counteract the diagonal ones.
- In bundle adjustment, certain well-known ambiguities (poorly-controlled parameter combinations) often dominate the uncertainty. Camera distance and focal length estimates, and structure depth and camera baseline ones (bas-relief), are both strongly correlated whenever the perspective is weak and become strict ambiguities in the affine limit. The well-conditioned diagonal blocks of the Hessian give no hint of these ambiguities: when both features and cameras are free, the overall network is *much* less rigid than it appears to be when each treats the other as fixed.
- During bundle adjustment, local structure refinements cause 'ripples' that must be propagated throughout the network. The camera-feature coupling information carried in the off-diagonal blocks is essential to this. In the diagonal-only model, ripples can propagate at most one feature-camera-feature step per iteration, so it takes many iterations for them to cross and re-cross a sparsely coupled network.

These arguments suggest that any approximation H_a to the bundle Hessian H that suppresses or significantly alters the off-diagonal terms is likely to have large $\|\mathsf{1} - \mathsf{H}_a^{-1}\mathsf{H}\|$ and hence slow convergence. This is exactly what we have observed in practice for all such methods that we have tested: near the

minimum, convergence is linear and for large problems often extremely slow, with $\|\mathsf{1} - \mathsf{H}_a^{-1}\mathsf{H}\|_2$ very close to 1. The iteration may either zigzag or converge slowly and monotonically, depending on the exact method and parameter values.

Line search: The above behaviour can be improved to some extent by adding a line search to the method. In principle, this is enough to ensure convergence for *any* positive definite H_a. However, accurate modelling of H is still highly desirable. Even with an exactly quadratic cost function and exact line searches (*i.e.* the minimum along the line is found exactly), the approximate Newton method (or, with $\mathsf{H}_a = \mathsf{1}$, steepest descent) converges at a rate[12]:

$$\|\mathsf{x}_{k+1}\| \;\leq\; \left(\frac{1 - C^{-1}}{1 + C^{-1}}\right)^{k+1} \|\mathsf{x}_0\| \;\approx_{\;C\gg 1}\; (1 - 2\,C^{-1})^{k+1}\; \|\mathsf{x}_0\| \tag{28}$$

Here, C is the condition number (ratio of largest to smallest eigenvalues) of $\mathsf{H}_a^{-1}\mathsf{H}$. If $\|\mathsf{H}_a\| \approx \|\mathsf{H}\|$, which is often the case in practice, adding a line search roughly halves the number of steps for a given final accuracy, as the convergence rate becomes $(1 - 2\,C^{-1})$ instead of $\|\mathsf{1} - \mathsf{H}_a^{-1}\mathsf{H}\| \approx (1 - C^{-1})$. (Of course, the line search itself is likely to require several function evaluations, so each step is now more expensive). More adaptive methods such as conjugate gradient and quasi-Newton also require a line search, but — at least in principle and in the absence of rounding errors — they can converge in $\mathcal{O}(n_{\mathsf{x}})$ iterations. For large bundle problems with thousands of parameters this is still prohibitive, so to make the method useful the number of iterations must be reduced further by incorporating knowledge about H via a suitable *preconditioner*.

7.3 Preconditioning

Steepest descent and Krylov methods are sensitive to the coordinate system and their practical success depends critically on good preconditioning. The aim is to find a linear transformation $\mathsf{x} \rightarrow \mathsf{T}\,\mathsf{x}$ and hence $\mathsf{g} \rightarrow \mathsf{T}^{-\top}\mathsf{g}$ and $\mathsf{H} \rightarrow \mathsf{T}^{-\top}\mathsf{H}\,\mathsf{T}$ for which the transformed H is near $\mathsf{1}$, or at least has only a few clusters of eigenvalues well separated from the origin. Ideally, T should be an accurate, low-cost approximation to the left Cholesky factor of H. (Exactly evaluating this would give the expensive Newton method again). In the experiments below, we tried conjugate gradient with preconditioners based on the diagonal blocks of H, and on **partial Cholesky decomposition**, dropping either all filled-in elements, or all that are smaller than a preset size when performing Cholesky decomposition. These methods were not competitive with the exact Gauss-Newton ones in the 'strip' experiments below, but for large enough problems it is likely that a preconditioned Krylov method would predominate, especially if more effective preconditioners could be found.

An exact Cholesky factor of H from a previous iteration is often a quite effective preconditioner. This gives hybrid methods in which H is only evaluated and factored every few iterations, with the Newton step at these iterations and well-preconditioned steepest descent or conjugate gradient at the others.

7.4 Experiments

Figure 6 shows the relative performance of several methods on two synthetic projective bundle adjustment problems. In both cases, the number of 3D points increases in proportion to the number of images, so the dense factorization time is $\mathcal{O}(n^3)$ where n is the number of points or images.

[12] Strictly speaking, this is only an upper bound, but it is usually an accurate estimate in practice. See, *e.g.*, [93, theorem 3.3].

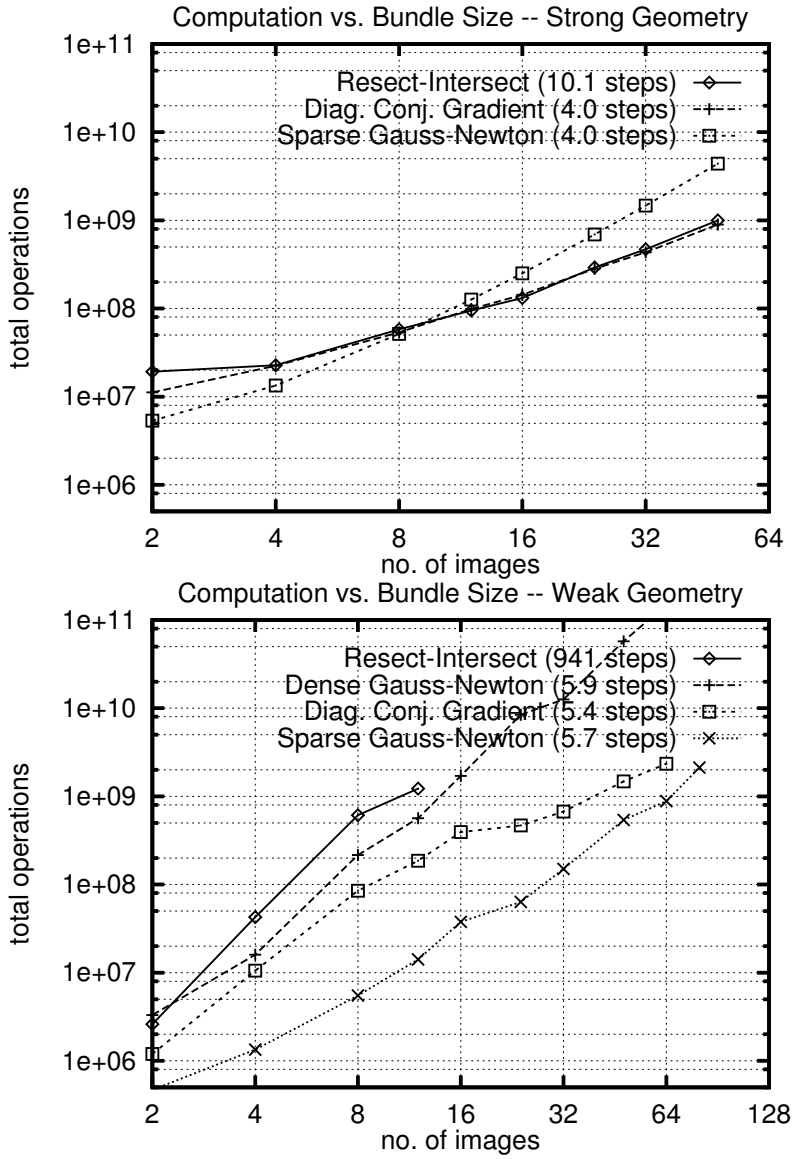


Figure 6: Relative speeds of various bundle optimization methods for strong 'spherical cloud' and weak 'strip' geometries.

The following methods are shown: 'Sparse Gauss-Newton' — sparse Cholesky decomposition with variables ordered naturally (features then cameras); 'Dense Gauss-Newton' — the same, but (inefficiently) ignoring all sparsity of the Hessian; 'Diag. Conj. Gradient' — the Newton step is found by an iterative conjugate gradient linear system solver, preconditioned using the Cholesky factors of the diagonal blocks of the Hessian; 'Resect-Intersect' — the state is optimized by alternate steps of resection and intersection, with relinearization after each. In the 'spherical cloud' problem, the points are uniformly distributed within a spherical cloud, all points are visible in all images, and the camera geometry is strongly convergent. These are ideal conditions, giving a low diameter network graph and a well-conditioned, nearly diagonal-dominant Hessian. All of the methods converge quite rapidly. Resection-intersection is a competitive method for larger problems owing to its low cost per

iteration. Unfortunately, although this geometry is often used for testing computer vision algorithms, it is atypical for large-scale bundle problems. The 'strip' experiment has a more representative geometry. The images are arranged in a long strip, with each feature seen in about 3 overlapping images. The strip's long thin weakly-connected network structure gives it large scale low stiffness 'flexing' modes, with correspondingly poor Hessian conditioning. The off-diagonal terms are critical here, so the approximate methods perform very poorly. Resection-intersection is slower even than dense Cholesky decomposition ignoring all sparsity. For 16 or more images it fails to converge even after 3000 iterations. The sparse Cholesky methods continue to perform reasonably well, with the natural, minimum degree and reverse Cuthill-McKee orderings all giving very similar run times in this case. For all of the methods that we tested, including resection-intersection with its linear per-iteration cost, the total run time for long chain-like geometries scaled roughly as $\mathcal{O}\left(n^3\right)$.

8 Implementation Strategy 3: Updating and Recursion

8.1 Updating rules

It is often convenient to be able to **update** a state estimate to reflect various types of changes, *e.g.* to incorporate new observations or to delete erroneous ones ('**downdating**'). Parameters may have to be added or deleted too. Updating rules are often used recursively, to incorporate a series of observations one-by-one rather than solving a single batch system. This is useful in on-line applications where a rapid response is needed, and also to provide preliminary predictions, *e.g.* for correspondence searches. Much of the early development of updating methods was aimed at on-line data editing in aerial cartography workstations.

The main challenge in adding or deleting observations is efficiently updating either a factorization of the Hessian H, or the covariance H^{-1}. Given either of these, the state update $\delta\mathsf{x}$ is easily found by solving the Newton step equations $\mathsf{H}\,\delta\mathsf{x} = -\mathsf{g}$, where (assuming that we started at an un-updated optimum $\mathsf{g} = \mathsf{0}$) the gradient g depends only on the newly added terms. The Hessian update $\mathsf{H} \rightarrow \mathsf{H} \pm \mathsf{B}\,\mathsf{W}\,\mathsf{B}^\top$ needs to have relatively low rank, otherwise nothing is saved over recomputing the batch solution. In least squares the rank is the number of independent observations added or deleted, but even without this the rank is often low in bundle problems because relatively few parameters are affected by any given observation.

One limitation of updating is that it is seldom as accurate as a batch solution owing to build-up of round-off error. Updating (adding observations) itself is numerically stable, but downdating (deleting observations) is potentially ill-conditioned as it reduces the positivity of the Hessian, and may cause previously good pivot choices to become arbitrarily bad. This is particularly a problem if all observations relating to a parameter are deleted, or if there are repeated insertion-deletion cycles as in time window filtering. Factorization updating methods are stabler than Woodbury formula / covariance updating ones.

Consider first the case where no parameters need be added nor deleted, *e.g.* adding or deleting an observation of an existing point in an existing image. Several methods have been suggested [54,66]. Mikhail & Helmering [88] use the Woodbury formula (18) to update the covariance H^{-1}. This simple approach becomes inefficient for problems with many features because the sparse structure is not exploited: the full covariance matrix is dense and we would normally avoid calculating it in its entirety. Grün [51,54] avoids this problem by maintaining a running copy of the reduced camera system (20), using an incremental Schur complement / forward substitution (16) to fold each new observation into this, and then re-factorizing and solving as usual after each update. This is effective

when there are many features in a few images, but for larger numbers of images it becomes inefficient owing to the re-factorization step. Factorization updating methods such as (56, 57) are currently the recommended update methods for most applications: they allow the existing factorization to be exploited, they handle any number of images and features and arbitrary problem structure efficiently, and they are numerically more accurate than Woodbury formula methods. The Givens rotation method [12,54], which is equivalent to the rank 1 Cholesky update (57), is probably the most common such method. The other updating methods are confusingly named in the literature. Mikhail & Helmering's method [88] is sometimes called 'Kalman filtering', even though no dynamics and hence no actual filtering is involved. Grün's reduced camera system method [51] is called 'triangular factor update (TFU)', even though it actually updates the (square) reduced Hessian rather than its triangular factors.

For updates involving a previously unseen 3D feature or image, new variables must also be added to the system. This is easy. We simply choose where to put the variables in the elimination sequence, and extend H and its L,D,U factors with the corresponding rows and columns, setting all of the newly created positions to zero (except for the unit diagonals of LDL$^\top$'s and LU's L factor). The factorization can then be updated as usual, presumably adding enough cost terms to make the extended Hessian nonsingular and couple the new parameters into the old network. If a direct covariance update is needed, the Woodbury formula (18) can be used on the old part of the matrix, then (17) to fill in the new blocks (equivalently, invert (55), with $\mathsf{D}_1 \leftarrow \mathsf{A}$ representing the old blocks and $\mathsf{D}_2 \leftarrow \mathsf{0}$ the new ones).

Conversely, it may be necessary to delete parameters, *e.g.* if an image or 3D feature has lost most or all of its support. The corresponding rows and columns of the Hessian H (and rows of g, columns of J) must be deleted, and all cost contributions involving the deleted parameters must also be removed using the usual factorization downdates (56, 57). To delete the rows and columns of block b in a matrix A, we first delete the b rows and columns of $\mathsf{L}, \mathsf{D}, \mathsf{U}$. This maintains triangularity and gives the correct trimmed A, except that the blocks in the lower right corner $\mathsf{A}_{ij} = \sum_{k \leq \min(i,j)} \mathsf{L}_{ik}\, \mathsf{D}_k\, \mathsf{U}_{kj}$, $i, j > b$ are missing a term $\mathsf{L}_{ib}\, \mathsf{D}_b\, \mathsf{U}_{bj}$ from the deleted column b of L / row b of U. This is added using an update $+\mathsf{L}_{*b}\, \mathsf{D}_b\, \mathsf{U}_{b*}$, $* > b$. To update A^{-1} when rows and columns of A are deleted, permute the deleted rows and columns to the end and use (17) backwards: $(\mathsf{A}_{11})^{-1} = (\mathsf{A}^{-1})_{11} - (\mathsf{A}^{-1})_{12}\, (\mathsf{A}^{-1})_{22}^{-1}\, (\mathsf{A}^{-1})_{21}$.

It is also possible to freeze some live parameters at fixed (current or default) values, or to add extra parameters / unfreeze some previously frozen ones, *c.f.* (49, 50) below. In this case, rows and columns corresponding to the frozen parameters must be deleted or added, but no other change to the cost function is required. Deletion is as above. To insert rows and columns $\mathsf{A}_{b*}, \mathsf{A}_{*b}$ at block b of matrix A, we open space in row and column b of $\mathsf{L}, \mathsf{D}, \mathsf{U}$ and fill these positions with the usual recursively defined values (52). For $i, j > b$, the sum (52) will now have a contribution $\mathsf{L}_{ib}\, \mathsf{D}_b\, \mathsf{U}_{bj}$ that it should not have, so to correct this we downdate the lower right submatrix $* > b$ with a cost cancelling contribution $-\mathsf{L}_{*b}\, \mathsf{D}_b\, \mathsf{U}_{b*}$.

8.2 Recursive Methods and Reduction

Each update computation is roughly quadratic in the size of the state vector, so if new features and images are continually added the situation will eventually become unmanageable. We must limit what we compute. In principle parameter refinement never stops: each observation update affects all components of the state estimate and its covariance. However, the refinements are in a sense trivial for parameters that are not directly coupled to the observation. If these parameters are eliminated using

reduction (19), the observation update can be applied directly to the reduced Hessian and gradient[13]. The eliminated parameters can then be updated by simple back-substitution (19) and their covariances by (17). In particular, if we cease to receive new information relating to a block of parameters (an image that has been fully treated, a 3D feature that has become invisible), they and all the observations relating to them can be subsumed once-and-for-all in a reduced Hessian and gradient on the remaining parameters. If required, we can later re-estimate the eliminated parameters by back-substitution. Otherwise, we do not need to consider them further.

This elimination process has some limitations. Only 'dead' parameters can be eliminated: to merge a new observation into the problem, we need the current Hessian or factorization entries for all parameter blocks relating to it. Reduction also commits us to a linearized / quadratic cost approximation for the eliminated variables at their current estimates, although to the extent that this model is correct, the remaining variables can still be treated nonlinearly. It is perhaps best to view reduction as the first half-iteration of a full nonlinear optimization: by (19), the Newton method for the full model can be implemented by repeated cycles of reduction, solving the reduced system, and back-substitution, with relinearization after each cycle, whereas for eliminated variables we stop after solving the first reduced system. Equivalently, reduction evaluates just the reduced components of the full Newton step and the full covariance, leaving us the option of computing the remaining eliminated ones later if we wish.

Reduction can be used to refine estimates of relative camera poses (or fundamental matrices, *etc.*) for a fixed set of images, by reducing a sequence of feature correspondences to their camera coordinates. Or conversely, to refine 3D structure estimates for a fixed set of features in many images, by reducing onto the feature coordinates.

Reduction is also the basis of recursive (Kalman) filtering. In this case, one has a (*e.g.* time) series of system state vectors linked by some probabilistic transition rule ('dynamical model'), for which we also have some observations ('observation model'). The parameter space consists of the combined state vectors for all times, *i.e.* it represents a path through the states. Both the dynamical and the observation models provide "observations" in the sense of probabilistic constraints on the full state parameters, and we seek a maximum likelihood (or similar) parameter estimate / path through the states. The full Hessian is block tridiagonal: the observations couple only to the current state and give the diagonal blocks, and dynamics couples only to the previous and next ones and gives the off-diagonal blocks (differential observations can also be included in the dynamics likelihood). So the model is large (if there are many time steps) but very sparse. As always with a tridiagonal matrix, the Hessian can be decomposed by recursive steps of reduction, at each step Schur complementing to get the current reduced block $\overline{\mathsf{H}}_t$ from the previous one $\overline{\mathsf{H}}_{t-1}$, the off-diagonal (dynamical) coupling $\mathsf{H}_{t\,t-1}$ and the current unreduced block (observation Hessian) H_t: $\overline{\mathsf{H}}_t = \mathsf{H}_t - \mathsf{H}_{t\,t-1}\,\overline{\mathsf{H}}_{t-1}^{-1}\,\mathsf{H}_{t\,t-1}^{\top}$. Similarly, for the gradient $\overline{\mathsf{g}}_t = \mathsf{g}_t - \mathsf{H}_{t\,t-1}\,\overline{\mathsf{H}}_{t-1}^{-1}\,\overline{\mathsf{g}}_{t-1}$, and as usual the reduced state update is $\delta\mathsf{x}_t = -\overline{\mathsf{H}}_t^{-1}\,\overline{\mathsf{g}}_t$.

This forwards reduction process is called **filtering**. At each time step it finds the optimal (linearized) current state estimate given all of the previous observations and dynamics. The corresponding unwinding of the recursion by back-substitution, **smoothing**, finds the optimal state estimate at each time given both past and future observations and dynamics. The usual equations of Kalman filtering and smoothing are easily derived from this recursion, but we will not do this here. We emphasize that filtering is merely the first half-iteration of a nonlinear optimization procedure: even for nonlinear dynamics and observation models, we can find the exact maximum likelihood state path by

[13] In (19), only D and b_2 are affected by the observation as it is independent of the subsumed components $\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{b}_1$. So applying the update to $\overline{\mathsf{D}}, \mathsf{b}_2$ has the same effect as applying it to D, b_2.

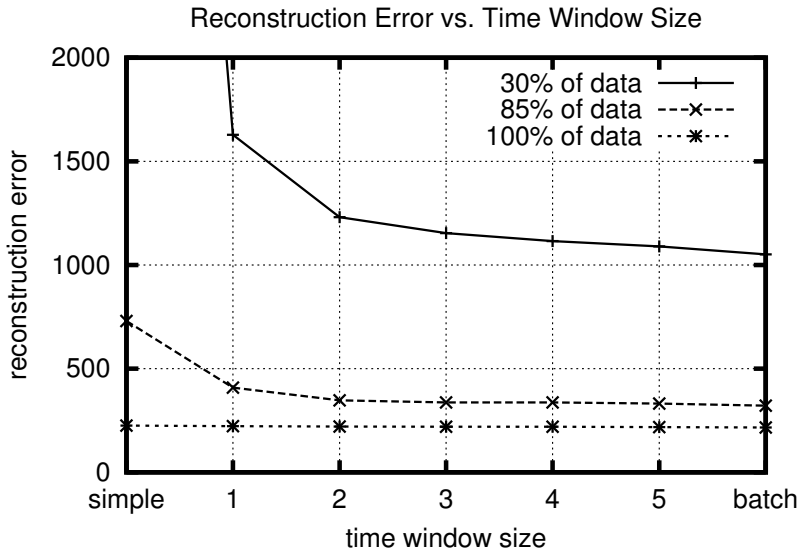


Figure 7: The residual state estimation error of the VSDF sequential bundle algorithm for progressively increasing sizes of rolling time window. The residual error at image $t = 16$ is shown for rolling windows of 1–5 previous images, and also for a 'batch' method (all previous images) and a 'simple' one (reconstruction / intersection is performed independently of camera location / resection). To simulate the effects of decreasing amounts of image data, 0%, 15% and 70% of the image measurements are randomly deleted to make runs with 100%, 85% and only 30% of the supplied image data. The main conclusion is that window size has little effect for strong data, but becomes increasingly important as the data becomes weaker.

cyclic passes of filtering and smoothing, with relinearization after each.

For long or unbounded sequences it may not be feasible to run the full iteration, but it can still be very helpful to run short sections of it, *e.g.* smoothing back over the last 3–4 state estimates then filtering forwards again, to verify previous correspondences and anneal the effects of nonlinearities. (The traditional **extended Kalman filter** optimizes nonlinearly over just the current state, assuming all previous ones to be linearized). The effects of variable window size on the Variable State Dimension Filter (VSDF) sequential bundle algorithm [85, 86, 83, 84] are shown in figure 7.

9 Gauge Freedom

Coordinates are a very convenient device for reducing geometry to algebra, but they come at the price of some arbitrariness. The coordinate system can be changed at any time, without affecting the underlying geometry. This is very familiar, but it leaves us with two problems: (*i*) algorithmically, we need some concrete way of deciding which particular coordinate system to use at each moment, and hence *breaking* the arbitrariness; (*ii*) we need to allow for the fact that the results may look quite different under different choices, even though they represent the same underlying geometry.

Consider the choice of 3D coordinates in visual reconstruction. The only objects in the 3D space are the reconstructed cameras and features, so we have to decide where to place the coordinate system relative to these ... Or in coordinate-centred language, where to place the reconstruction relative to the coordinate system. Moreover, bundle adjustment updates and uncertainties can perturb the recon-

structed structure almost arbitrarily, so we must specify coordinate systems not just for the current structure, but also for *every possible nearby one*. Ultimately, this comes down to constraining the coordinate values of certain aspects of the reconstructed structure — features, cameras or combinations of these — whatever the rest of the structure might be. Saying this more intrinsically, the coordinate frame is specified and held fixed with respect to the chosen reference elements, and the rest of the geometry is then expressed in this frame as usual. In measurement science such a set of coordinate system specifying rules is called a **datum**, but we will follow the wider mathematics and physics usage and call it a **gauge**[14]. The freedom in the choice of coordinate fixing rules is called **gauge freedom**.

As a gauge anchors the coordinate system rigidly to its chosen reference elements, perturbing the reference elements has no effect on their own coordinates. Instead, it changes the coordinate system itself and hence systematically changes the coordinates of all the *other* features, while leaving the reference coordinates fixed. Similarly, uncertainties in the reference elements do not affect their own coordinates, but appear as highly correlated uncertainties in all of the *other* reconstructed features. The moral is that *structural perturbations and uncertainties are highly relative*. Their form depends profoundly on the gauge, and especially on how this changes as the state varies (*i.e.* which elements it holds fixed). The effects of disturbances are *not* restricted to the coordinates of the features actually disturbed, but may appear almost anywhere depending on the gauge.

In visual reconstruction, the differences between object-centred and camera-centred gauges are often particularly marked. In object-centred gauges, object points appear to be relatively certain while cameras appear to have large and highly correlated uncertainties. In camera-centred gauges, it is the camera that appears to be precise and the object points that appear to have large correlated uncertainties. One often sees statements like "the reconstructed depths are very uncertain". This may be true in the camera frame, yet the object may be very well reconstructed in its own frame — it all depends on what fraction of the total depth fluctuations are simply due to global uncertainty in the camera location, and hence identical for all object points.

Besides 3D coordinates, many other types of geometric parametrization in vision involve arbitrary choices, and hence are subject to gauge freedoms [106]. These include the choice of: homogeneous scale factors in homogeneous-projective representations; supporting points in supporting-point based representations of lines and planes; reference plane in plane + parallax representations; and homographies in homography-epipole representations of matching tensors. In each case the symptoms and the remedies are the same.

9.1 General Formulation

The general set up is as follows: We take as our state vector x the set of all of the 3D feature coordinates, camera poses and calibrations, *etc*., that enter the problem. This state space has internal symmetries related to the arbitrary choices of 3D coordinates, homogeneous scale factors, *etc*., that are embedded in x. Any two state vectors that differ only by such choices represent the same underlying 3D geometry, and hence have exactly the same image projections and other intrinsic properties. So under change-of-coordinates equivalence, the state space is partitioned into classes of intrinsically equivalent state vectors, each class representing exactly one underlying 3D geometry. These classes are called **gauge orbits**. Formally, they are the group orbits of the state space action of the relevant **gauge group** (coordinate transformation group), but we will not need the group structure below. A

[14] Here, *gauge* just means reference frame. The sense is that of a reference against which something is *judged* (O.Fr. jauger, gauger). Pronounce $g\bar{e}^{i}dj$.

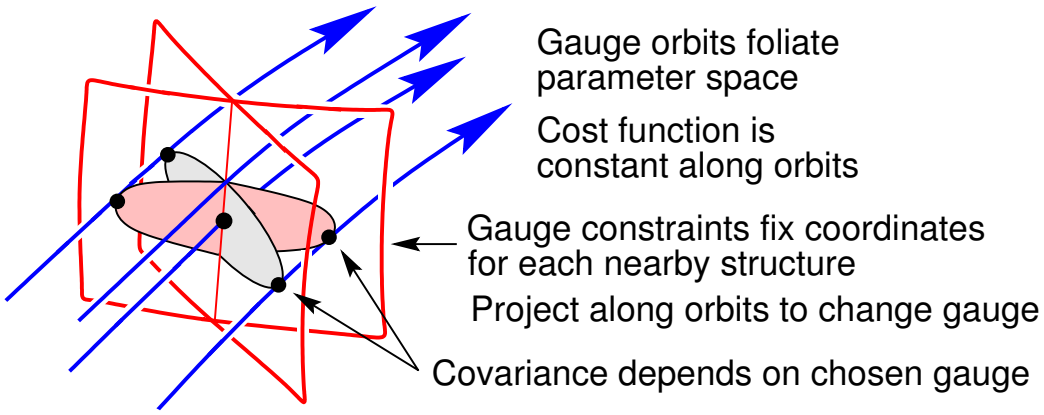


Figure 8: Gauge orbits in state space, two gauge cross-sections and their covariances.

state space function represents an intrinsic function of the underlying geometry if and only if it is constant along each gauge orbit (*i.e.* coordinate system independent). Such quantities are called **gauge invariants**. We want the bundle adjustment cost function to quantify 'intrinsic merit', so it must be chosen to be gauge invariant.

In visual reconstruction, the principal gauge groups are the $3+3+1=7$ dimensional group of 3D similarity (scaled Euclidean) transformations for Euclidean reconstruction, and the 15 dimensional group of projective 3D coordinate transformations for projective reconstruction. But other gauge freedoms are also present. Examples include: (*i*) The arbitrary scale factors of homogeneous projective feature representations, with their 1D rescaling gauge groups. (*ii*) The arbitrary positions of the points in 'two point' line parametrizations, with their two 1D motion-along-line groups. (*iii*) The underspecified 3×3 homographies used for 'homography + epipole' parametrizations of matching tensors [77,62,106]. For example, the fundamental matrix can be parametrized as $\boldsymbol{F} = [\boldsymbol{e}]_\times \boldsymbol{H}$ where $\boldsymbol{e}$ is its left epipole and $\boldsymbol{H}$ is the inter-image homography induced by any 3D plane. The choice of plane gives a freedom $\boldsymbol{H} \rightarrow \boldsymbol{H} + \boldsymbol{e}\,\boldsymbol{a}^\top$ where $\boldsymbol{a}$ is an arbitrary 3-vector, and hence a 3D linear gauge group.

Now consider how to specify a gauge, *i.e.* a rule saying how each possible underlying geometry near the current one should be expressed in coordinates. Coordinatizations are represented by state space points, so this is a matter of choosing exactly one point (structure coordinatization) from each gauge orbit (underlying geometry). Mathematically, the gauge orbits foliate (fill without crossing) the state space, and a gauge is a local transversal 'cross-section' $\mathcal{G}$ through this foliation. See fig. 8. Different gauges represent different but geometrically equivalent coordinatization rules. Results can be mapped between gauges by pushing them along gauge orbits, *i.e.* by applying local coordinate transformations that vary depending on the particular structure involved. Such transformations are called **S-transforms** ('similarity' transforms) [6,107,22,25]. Different gauges through the same central state represent coordinatization rules that agree for the central geometry but differ for perturbed ones — the S-transform is the identity at the centre but not elsewhere.

Given a gauge, only state perturbations that lie within the gauge cross-section are authorized. This is what we want, as such state perturbations are in one-to-one correspondence with perturbations of the underlying geometry. Indeed, any state perturbation is equivalent to some on-gauge one under the gauge group (*i.e.* under a small coordinate transformation that pushes the perturbed state along its gauge orbit until it meets the gauge cross-section). State perturbations along the gauge orbits are

uninteresting, because they do not change the underlying geometry at all.

Covariances are averages of squared perturbations and must also be based on on-gauge perturbations (they would be infinite if we permitted perturbations along the gauge orbits, as there is no limit to these — they do not change the cost at all). So covariance matrices are gauge-dependent and in fact represent ellipsoids tangent to the gauge cross-section at the cost minimum. They can look very different for different gauges. But, as with states, S-transforms map them between gauges by projecting along gauge orbits / state equivalence classes.

Note that there is no intrinsic notion of orthogonality on state space, so it is meaningless to ask which state-space directions are 'orthogonal' to the gauge orbits. This would involve deciding when two different structures have been "expressed in the same coordinate system", so every gauge believes its own cross section to be orthogonal and all others to be skewed.

9.2 Gauge constraints

We will work near some point x of state space, perhaps a cost minimum or a running state estimate. Let n_{x} be the dimension of x and n_{g} the dimension of the gauge orbits. Let $\mathsf{f}, \mathsf{g}, \mathsf{H}$ be the cost function and its gradient and Hessian, and G be any $n_{\mathsf{x}} \times n_{\mathsf{g}}$ matrix whose columns span the local gauge orbit directions at x [15]. By the exact gauge invariance of f, its gradient and Hessian vanish along orbit directions: $\mathsf{g}^\top \mathsf{G} = \mathsf{0}$ and $\mathsf{H}\,\mathsf{G} = \mathsf{0}$. Note that the gauged Hessian H is singular with (at least) rank deficiency n_{g} and null space G. This is called **gauge deficiency**. Many numerical optimization routines assume nonsingular H, and must be modified to work in gauge invariant problems. The singularity is an expression of indifference: when we come to calculate state updates, any two updates ending on the same gauge orbit are equivalent, both in terms of cost and in terms of the change in the underlying geometry. All that we need is a method of telling the routine which particular update to choose.

Gauge constraints are the most direct means of doing this. A gauge cross-section $\mathcal{G}$ can be specified in two ways: (*i*) *constrained form:* specify n_{g} local constraints $\mathsf{d}(\mathsf{x})$ with $\mathsf{d}(\mathsf{x}) = \mathsf{0}$ for points on $\mathcal{G}$; (*ii*) *parametric form:* specify a function $\mathsf{x}(\mathsf{y})$ of $n_{\mathsf{x}} - n_{\mathsf{g}}$ independent local parameters y, with $\mathsf{x} = \mathsf{x}(\mathsf{y})$ being the points of $\mathcal{G}$. For example, a **trivial gauge** is one that simply freezes the values of n_{g} of the parameters in x (usually feature or camera coordinates). In this case we can take $\mathsf{d}(\mathsf{x})$ to be the parameter freezing constraints and y to be the remaining unfrozen parameters. Note that once the gauge is fixed the problem is no longer gauge invariant — the whole purpose of $\mathsf{d}(\mathsf{x}), \mathsf{x}(\mathsf{y})$ is to *break* the underlying gauge invariance.

Examples of trivial gauges include: (*i*) using several visible 3D points as a 'projective basis' for reconstruction (*i.e.* fixing their projective 3D coordinates to simple values, as in [27]); and (*ii*) fixing the components of one projective 3×4 camera matrix as $(\boldsymbol{I}\ \boldsymbol{0})$, as in [61] (this only partially fixes the 3D projective gauge — 3 projective 3D degrees of freedom remain unfixed).

Linearized gauge: Let the local linearizations of the gauge functions be:

$$\mathsf{d}(\mathsf{x} + \boldsymbol{\delta}\mathsf{x}) \;\approx\; \mathsf{d}(\mathsf{x}) + \mathsf{D}\,\boldsymbol{\delta}\mathsf{x} \qquad\qquad \mathsf{D} \;\equiv\; \frac{\mathsf{dd}}{\mathsf{dx}} \tag{29}$$

$$\mathsf{x}(\mathsf{y} + \boldsymbol{\delta}\mathsf{y}) \;\approx\; \mathsf{x}(\mathsf{y}) + \mathsf{Y}\,\boldsymbol{\delta}\mathsf{y} \qquad\qquad \mathsf{Y} \;\equiv\; \frac{\mathsf{dx}}{\mathsf{dy}} \tag{30}$$

Compatibility between the two gauge specification methods requires $\mathsf{d}(\mathsf{x}(\mathsf{y})) = \mathsf{0}$ for all y, and hence $\mathsf{D}\,\mathsf{Y} = \mathsf{0}$. Also, since $\mathcal{G}$ must be transversal to the gauge orbits, $\mathsf{D}\,\mathsf{G}$ must have full rank n_{g} and

[15] A suitable G is easily calculated from the infinitesimal action of the gauge group on x. For example, for spatial similarities the columns of G would be the $n_{\mathsf{g}} = 3 + 3 + 1 = 7$ state velocity vectors describing the effects of infinitesimal translations, rotations and changes of spatial scale on x.

$(\mathsf{Y}\ \ \mathsf{G})$ must have full rank n_{x}. Assuming that x itself is on $\mathcal{G}$, a perturbation $\mathsf{x} + \boldsymbol{\delta}\mathsf{x}_{\mathcal{G}}$ is on $\mathcal{G}$ to first order iff $\mathsf{D}\,\boldsymbol{\delta}\mathsf{x}_{\mathcal{G}} = \mathsf{0}$ or $\boldsymbol{\delta}\mathsf{x}_{\mathcal{G}} = \mathsf{Y}\,\boldsymbol{\delta}\mathsf{y}$ for some $\boldsymbol{\delta}\mathsf{y}$.

Two $n_{\mathsf{x}} \times n_{\mathsf{x}}$ rank $n_{\mathsf{x}} - n_{\mathsf{g}}$ matrices characterize $\mathcal{G}$. The **gauge projection matrix** $\mathsf{P}_{\mathcal{G}}$ implements linearized projection of state displacement vectors $\boldsymbol{\delta}\mathsf{x}$ along their gauge orbits onto the local gauge cross-section: $\boldsymbol{\delta}\mathsf{x} \to \boldsymbol{\delta}\mathsf{x}_{\mathcal{G}} = \mathsf{P}_{\mathcal{G}}\,\boldsymbol{\delta}\mathsf{x}$. (The projection is usually non-orthogonal: $\mathsf{P}_{\mathcal{G}}^{\top} \neq \mathsf{P}_{\mathcal{G}}$). The **gauged covariance matrix** $\mathsf{V}_{\mathcal{G}}$ plays the role of the inverse Hessian. It gives the cost-minimizing Newton step within $\mathcal{G}$, $\boldsymbol{\delta}\mathsf{x}_{\mathcal{G}} = -\mathsf{V}_{\mathcal{G}}\,\mathsf{g}$, and also the asymptotic covariance of $\boldsymbol{\delta}\mathsf{x}_{\mathcal{G}}$. $\mathsf{P}_{\mathcal{G}}$ and $\mathsf{V}_{\mathcal{G}}$ have many special properties and equivalent forms. For convenience, we display some of these now[16] — let $\mathsf{V} \equiv (\mathsf{H} + \mathsf{D}^{\top}\mathsf{B}\,\mathsf{D})^{-1}$ where B is any nonsingular symmetric $n_{\mathsf{g}} \times n_{\mathsf{g}}$ matrix, and let $\mathcal{G}'$ be any other gauge:

$$\begin{aligned}
\mathsf{V}_{\mathcal{G}} &\equiv \mathsf{Y}\,(\mathsf{Y}^{\top}\mathsf{H}\,\mathsf{Y})^{-1}\,\mathsf{Y}^{\top} = \mathsf{V}\,\mathsf{H}\,\mathsf{V} = \mathsf{V} - \mathsf{G}\,(\mathsf{D}\,\mathsf{G})^{-1}\mathsf{B}^{-1}\,(\mathsf{D}\,\mathsf{G})^{-\top}\mathsf{G}^{\top} && (31)\\
&= \mathsf{P}_{\mathcal{G}}\,\mathsf{V} = \mathsf{P}_{\mathcal{G}}\,\mathsf{V}_{\mathcal{G}} = \mathsf{P}_{\mathcal{G}}\,\mathsf{V}_{\mathcal{G}'}\,\mathsf{P}_{\mathcal{G}}^{\top} && (32)\\
\mathsf{P}_{\mathcal{G}} &\equiv \mathsf{1} - \mathsf{G}\,(\mathsf{D}\,\mathsf{G})^{-1}\,\mathsf{D} = \mathsf{Y}\,(\mathsf{Y}^{\top}\mathsf{H}\,\mathsf{Y})^{-1}\mathsf{Y}^{\top}\mathsf{H} = \mathsf{V}\,\mathsf{H} = \mathsf{V}_{\mathcal{G}}\,\mathsf{H} = \mathsf{P}_{\mathcal{G}}\,\mathsf{P}_{\mathcal{G}'} && (33)\\
\mathsf{P}_{\mathcal{G}}\,\mathsf{G} &= \mathsf{0}, \qquad \mathsf{P}_{\mathcal{G}}\,\mathsf{Y} = \mathsf{Y}, \qquad \mathsf{D}\,\mathsf{P}_{\mathcal{G}} = \mathsf{D}\,\mathsf{V}_{\mathcal{G}} = \mathsf{0} && (34)\\
\mathsf{g}^{\top}\mathsf{P}_{\mathcal{G}} &= \mathsf{g}^{\top}, \qquad \mathsf{H}\,\mathsf{P}_{\mathcal{G}} = \mathsf{H}, \qquad \mathsf{V}_{\mathcal{G}}\,\mathsf{g} = \mathsf{V}\,\mathsf{g} && (35)
\end{aligned}$$

These relations can be summarized by saying that $\mathsf{V}_{\mathcal{G}}$ is the $\mathcal{G}$-supported generalized inverse of H and that $\mathsf{P}_{\mathcal{G}}$: (*i*) projects along gauge orbits ($\mathsf{P}_{\mathcal{G}}\,\mathsf{G} = \mathsf{0}$); (*ii*) projects onto the gauge cross-section $\mathcal{G}$ ($\mathsf{D}\,\mathsf{P}_{\mathcal{G}} = \mathsf{0},\ \mathsf{P}_{\mathcal{G}}\,\mathsf{Y} = \mathsf{Y},\ \mathsf{P}_{\mathcal{G}}\boldsymbol{\delta}\mathsf{x} = \boldsymbol{\delta}\mathsf{x}_{\mathcal{G}}$ and $\mathsf{V}_{\mathcal{G}} = \mathsf{P}_{\mathcal{G}}\,\mathsf{V}_{\mathcal{G}'}\,\mathsf{P}_{\mathcal{G}}^{\top}$); and (*iii*) preserves gauge invariants (*e.g.* $\mathsf{f}(\mathsf{x} + \mathsf{P}_{\mathcal{G}}\,\boldsymbol{\delta}\mathsf{x}) = \mathsf{f}(\mathsf{x} + \boldsymbol{\delta}\mathsf{x}),\ \mathsf{g}^{\top}\mathsf{P}_{\mathcal{G}} = \mathsf{g}^{\top}$ and $\mathsf{H}\,\mathsf{P}_{\mathcal{G}} = \mathsf{H}$). Both $\mathsf{V}_{\mathcal{G}}$ and H have rank $n_{\mathsf{x}} - n_{\mathsf{g}}$. Their null spaces $\mathsf{D}^{\top}$ and G are transversal but otherwise unrelated. $\mathsf{P}_{\mathcal{G}}$ has left null space D and right null space G.

State updates: It is straightforward to add gauge fixing to the bundle update equations. First consider the constrained form. Enforcing the gauge constraints $\mathsf{d}(\mathsf{x} + \boldsymbol{\delta}\mathsf{x}_{\mathcal{G}}) = \mathsf{0}$ with Lagrange multipliers $\boldsymbol{\lambda}$ gives an SQP step:

$$\begin{pmatrix} \mathsf{H} & \mathsf{D}^{\top} \\ \mathsf{D} & \mathsf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\delta}\mathsf{x}_{\mathcal{G}} \\ \boldsymbol{\lambda} \end{pmatrix} = -\begin{pmatrix} \mathsf{g} \\ \mathsf{d} \end{pmatrix}, \qquad \begin{pmatrix} \mathsf{H} & \mathsf{D}^{\top} \\ \mathsf{D} & \mathsf{0} \end{pmatrix}^{-1} = \begin{pmatrix} \mathsf{V}_{\mathcal{G}} & \mathsf{G}\,(\mathsf{D}\,\mathsf{G})^{-1} \\ (\mathsf{D}\,\mathsf{G})^{-\top}\mathsf{G}^{\top} & \mathsf{0} \end{pmatrix} \tag{36}$$

$$\text{so} \quad \boldsymbol{\delta}\mathsf{x}_{\mathcal{G}} = -\left(\mathsf{V}_{\mathcal{G}}\,\mathsf{g} + \mathsf{G}\,(\mathsf{D}\,\mathsf{G})^{-1}\mathsf{d}\right), \qquad \boldsymbol{\lambda} = \mathsf{0} \tag{37}$$

This is a rather atypical constrained problem. For typical cost functions the gradient has a component pointing away from the constraint surface, so $\mathsf{g} \neq \mathsf{0}$ at the constrained minimum and a non-vanishing force $\boldsymbol{\lambda} \neq \mathsf{0}$ is required to hold the solution on the constraints. Here, the cost function and its derivatives are entirely indifferent to motions along the orbits. Nothing actively forces the state to move off the gauge, so the constraint force $\boldsymbol{\lambda}$ vanishes everywhere, g vanishes at the optimum, and the constrained minimum value of f is identical to the unconstrained minimum. The only effect of the constraints is to correct any gradual drift away from $\mathcal{G}$ that happens to occur, via the d term in $\boldsymbol{\delta}\mathsf{x}_{\mathcal{G}}$.

A simpler way to get the same effect is to add a gauge-invariance breaking term such as $\frac{1}{2}\mathsf{d}(\mathsf{x})^{\top}\mathsf{B}\,\mathsf{d}(\mathsf{x})$ to the cost function, where B is some positive $n_{\mathsf{g}} \times n_{\mathsf{g}}$ weight matrix. Note that $\frac{1}{2}\mathsf{d}(\mathsf{x})^{\top}\mathsf{B}\,\mathsf{d}(\mathsf{x})$ has a unique minimum of 0 on each orbit at the point $\mathsf{d}(\mathsf{x}) = \mathsf{0}$, *i.e.* for x on $\mathcal{G}$. As f is constant

[16] These results are most easily proved by inserting strategic factors of $(\mathsf{Y}\ \ \mathsf{G})\,(\mathsf{Y}\ \ \mathsf{G})^{-1}$ and using $\mathsf{H}\,\mathsf{G} = \mathsf{0}$, $\mathsf{D}\,\mathsf{Y} = \mathsf{0}$ and $(\mathsf{Y}\ \ \mathsf{G})^{-1} = \begin{pmatrix} (\mathsf{Y}^{\top}\mathsf{H}\,\mathsf{Y})^{-1}\mathsf{Y}^{\top}\mathsf{H} \\ (\mathsf{D}\,\mathsf{G})^{-1}\,\mathsf{D} \end{pmatrix}$. For any $n_{\mathsf{g}} \times n_{\mathsf{g}}$ B including $\mathsf{0}$, $\begin{pmatrix} \mathsf{Y}^{\top} \\ \mathsf{G}^{\top} \end{pmatrix}\,(\mathsf{H} + \mathsf{D}^{\top}\mathsf{B}\,\mathsf{D})\,(\mathsf{Y}\ \ \mathsf{G}) = \begin{pmatrix} \mathsf{Y}^{\top}\mathsf{H}\,\mathsf{Y} & \mathsf{0} \\ \mathsf{0} & (\mathsf{D}\,\mathsf{G})^{\top}\mathsf{B}\,(\mathsf{D}\,\mathsf{G}) \end{pmatrix}$. If B is nonsingular, $\mathsf{V} = (\mathsf{H} + \mathsf{D}^{\top}\mathsf{B}\,\mathsf{D})^{-1} = \mathsf{Y}\,(\mathsf{Y}^{\top}\mathsf{H}\,\mathsf{Y})^{-1}\mathsf{Y}^{\top} + \mathsf{G}\,(\mathsf{D}\,\mathsf{G})^{-1}\mathsf{B}^{-1}\,(\mathsf{D}\,\mathsf{G})^{-\top}\mathsf{G}^{\top}$.

along gauge orbits, optimization of $\mathsf{f}(\mathsf{x}) + \frac{1}{2}\mathsf{d}(\mathsf{x})^\top \mathsf{B}\,\mathsf{d}(\mathsf{x})$ along each orbit enforces $\mathcal{G}$ and hence returns the orbit's f value, so global optimization will find the global constrained minimum of f. The cost function $\mathsf{f}(\mathsf{x}) + \frac{1}{2}\mathsf{d}(\mathsf{x})^\top \mathsf{B}\,\mathsf{d}(\mathsf{x})$ is nonsingular with Newton step $\boldsymbol{\delta}\mathsf{x}_\mathcal{G} = \mathsf{V}\,(\mathsf{g} + \mathsf{D}^\top \mathsf{B}\,\mathsf{d})$ where $\mathsf{V} = (\mathsf{H} + \mathsf{D}^\top \mathsf{B}\,\mathsf{D})^{-1}$ is the new inverse Hessian. By (35, 31), this is identical to the SQP step (37), so the SQP and cost-modifying methods are equivalent. This strategy works only because no force is required to keep the state on-gauge — if this were not the case, the weight B would have to be infinite. Also, for dense D this form is not practically useful because $\mathsf{H} + \mathsf{D}^\top \mathsf{B}\,\mathsf{D}$ is dense and hence slow to factorize, although updating formulae can be used.

Finally, consider the parametric form $\mathsf{x} = \mathsf{x}(\mathsf{y})$ of $\mathcal{G}$. Suppose that we already have a current reduced state estimate y. We can approximate $\mathsf{f}(\mathsf{x}(\mathsf{y} + \boldsymbol{\delta}\mathsf{y}))$ to get a reduced system for $\boldsymbol{\delta}\mathsf{y}$, solve this, and find $\boldsymbol{\delta}\mathsf{x}_\mathcal{G}$ afterwards if necessary:

$$(\mathsf{Y}^\top \mathsf{H}\,\mathsf{Y})\ \boldsymbol{\delta}\mathsf{y} \ = \ -\mathsf{Y}^\top \mathsf{g}\,, \qquad \boldsymbol{\delta}\mathsf{x}_\mathcal{G} \ = \ \mathsf{Y}\,\boldsymbol{\delta}\mathsf{y} \ = \ -\mathsf{V}_\mathcal{G}\,\mathsf{g} \tag{38}$$

The $(n_\mathsf{x} - n_\mathsf{g}) \times (n_\mathsf{x} - n_\mathsf{g})$ matrix $\mathsf{Y}^\top \mathsf{H}\,\mathsf{Y}$ is generically nonsingular despite the singularity of H. In the case of a trivial gauge, Y simply selects the submatrices of g, H corresponding to the unfrozen parameters, and solves for these. For less trivial gauges, both Y and D are often dense and there is a risk that substantial fill-in will occur in all of the above methods.

Gauged covariance: By (31) and standard covariance propagation in (38), the covariance of the on-gauge fluctuations $\boldsymbol{\delta}\mathsf{x}_\mathcal{G}$ is $\mathsf{E}\left[\boldsymbol{\delta}\mathsf{x}_\mathcal{G}\,\boldsymbol{\delta}\mathsf{x}_\mathcal{G}^\top\right] = \mathsf{Y}\,(\mathsf{Y}^\top \mathsf{H}\,\mathsf{Y})^{-1}\,\mathsf{Y}^\top = \mathsf{V}_\mathcal{G}$. $\boldsymbol{\delta}\mathsf{x}_\mathcal{G}$ never moves off $\mathcal{G}$, so $\mathsf{V}_\mathcal{G}$ represents a rank $n_\mathsf{x} - n_\mathsf{g}$ covariance ellipsoid 'flattened onto $\mathcal{G}$'. In a trivial gauge, $\mathsf{V}_\mathcal{G}$ is the covariance $(\mathsf{Y}^\top \mathsf{H}\,\mathsf{Y})^{-1}$ of the free variables, padded with zeros for the fixed ones.

Given $\mathsf{V}_\mathcal{G}$, the linearized gauged covariance of a function $\mathsf{h}(\mathsf{x})$ is $\frac{\mathsf{dh}}{\mathsf{dx}}\,\mathsf{V}_\mathcal{G}\,\frac{\mathsf{dh}}{\mathsf{dx}}^\top$ as usual. If $\mathsf{h}(\mathsf{x})$ is gauge invariant (constant along gauge orbits) this is just its ordinary covariance. Intuitively, $\mathsf{V}_\mathcal{G}$ and $\frac{\mathsf{dh}}{\mathsf{dx}}\,\mathsf{V}_\mathcal{G}\,\frac{\mathsf{dh}}{\mathsf{dx}}^\top$ depend on the gauge because they measure not absolute uncertainty, but uncertainty relative to the reference features on which the gauge is based. Just as there are no absolute reference frames, there are no absolute uncertainties. The best we can do is relative ones.

Gauge transforms: We can change the gauge at will during a computation, *e.g.* to improve sparseness or numerical conditioning or re-express results in some standard gauge. This is simply a matter of an **S-transform** [6], *i.e.* pushing all gauged quantities along their gauge orbits onto the new gauge cross-section $\mathcal{G}$. We will assume that the base point x is unchanged. If not, a fixed (structure independent) change of coordinates achieves this. Locally, an S-transform then linearizes into a linear projection along the orbits spanned by G onto the new gauge constraints given by D or Y. This is implemented by the $n_\mathsf{x} \times n_\mathsf{x}$ rank $n_\mathsf{x} - n_\mathsf{g}$ non-orthogonal projection matrix $\mathsf{P}_\mathcal{G}$ defined in (33). The projection preserves all gauge invariants — *e.g.* $\mathsf{f}(\mathsf{x} + \mathsf{P}_\mathcal{G}\,\boldsymbol{\delta}\mathsf{x}) = \mathsf{f}(\mathsf{x} + \boldsymbol{\delta}\mathsf{x})$ — and it cancels the effects of projection onto any other gauge: $\mathsf{P}_\mathcal{G}\,\mathsf{P}_{\mathcal{G}'} = \mathsf{P}_\mathcal{G}$.

9.3 Inner Constraints

Given the wide range of gauges and the significant impact that they have on the appearance of the state updates and covariance matrix, it is useful to ask which gauges give the "smallest" or "best behaved" updates and covariances. This is useful for interpreting and comparing results, and it also gives beneficial numerical properties. Basically it is a matter of deciding which features or cameras we care most about and tying the gauge to some stable average of them, so that gauge-induced correlations in them are as small as possible. For object reconstruction the resulting gauge will usually be object-centred, for vehicle navigation camera-centred. We stress that such choices are only a mat-

ter of superficial appearance: in principle, all gauges are equivalent and give identical values and covariances for all gauge invariants.

Another way to say this is that it is only for gauge invariants that we can find meaningful (coordinate system independent) values and covariances. But one of the most fruitful ways to create invariants is to locate features w.r.t. a basis of reference features, *i.e.* w.r.t. the gauge based on them. The choice of inner constraints is thus a choice of a stable basis of compound features w.r.t. which invariants can be measured. By including an average of many features in the compound, we reduce the invariants' dependence on the basis features.

As a performance criterion we can minimize some sort of weighted average size, either of the state update or of the covariance. Let W be an $n_{\mathsf{x}} \times n_{\mathsf{x}}$ information-like weight matrix encoding the relative importance of the various error components, and L be any left square root for it, $\mathsf{L}\,\mathsf{L}^\top = \mathsf{W}$. *The local gauge at* x *that minimizes the weighted size of the state update* $\delta\mathsf{x}_{\mathcal{G}}^\top\,\mathsf{W}\,\delta\mathsf{x}_{\mathcal{G}}$*, the weighted covariance sum* $\mathrm{Trace}(\mathsf{W}\,\mathsf{V}_{\mathcal{G}}) = \mathrm{Trace}(\mathsf{L}^\top\,\mathsf{V}_{\mathcal{G}}\,\mathsf{L})$*, and the* L_2 *or Frobenius norm of* $\mathsf{L}^\top\,\mathsf{V}_{\mathcal{G}}\,\mathsf{L}$*, is given by the* **inner constraints** [87, 89, 6, 22, 25][17]:

$$\mathsf{D}\,\delta\mathsf{x} \;=\; 0 \qquad \text{where} \qquad \mathsf{D} \;\equiv\; \mathsf{G}^\top\,\mathsf{W} \tag{39}$$

The corresponding covariance $\mathsf{V}_{\mathcal{G}}$ is given by (31) with $\mathsf{D} = \mathsf{G}^\top\,\mathsf{W}$, and the state update is $\delta\mathsf{x}_{\mathcal{G}} = -\mathsf{V}_{\mathcal{G}}\,\mathsf{g}$ as usual. Also, if W is nonsingular, $\mathsf{V}_{\mathcal{G}}$ is given by the weighted rank $n_{\mathsf{x}} - n_{\mathsf{g}}$ pseudo-inverse $\mathsf{L}^{-\top}\,(\mathsf{L}^{-1}\,\mathsf{H}\,\mathsf{L}^{-\top})^\dagger\,\mathsf{L}^{-1}$, where $\mathsf{W} = \mathsf{L}\,\mathsf{L}^\top$ is the Cholesky decomposition of W and $(\cdot)^\dagger$ is the Moore-Penrose pseudo-inverse.

The inner constraints are covariant under global transformations $\mathsf{x} \to \mathsf{t}(\mathsf{x})$ provided that W is transformed in the usual information matrix / Hessian way $\mathsf{W} \to \mathsf{T}^{-\top}\,\mathsf{W}\,\mathsf{T}^{-1}$ where $\mathsf{T} = \frac{\mathsf{dt}}{\mathsf{dx}}$ [18]. However, such transformations seldom preserve the form of W (diagonality, $\mathsf{W} = 1$, *etc.*). If W represents an isotropic weighted sum over 3D points[19], its form is preserved under global 3D Euclidean transformations, and rescaled under scalings. But this extends neither to points under projective transformations, nor to camera poses, 3D planes and other non-point-like features even under Euclidean ones. (The choice of origin has a significant influence For poses, planes, *etc.* : changes of origin propagate rotational uncertainties into translational ones).

Inner constraints were originally introduced in geodesy in the case $\mathsf{W} = 1$ [87]. The meaning of this is entirely dependent on the chosen 3D coordinates and variable scaling. In bundle adjustment there is little to recommend $\mathsf{W} = 1$ unless the coordinate origin has been carefully chosen and the variables carefully pre-scaled as above, *i.e.* $\mathsf{x} \to \mathsf{L}^\top\,\mathsf{x}$ and hence $\mathsf{H} \to \mathsf{L}^{-1}\,\mathsf{H}\,\mathsf{L}^{-\top}$, where $\mathsf{W} \sim \mathsf{L}\,\mathsf{L}^\top$ is a

[17] *Sketch proof*: For $\mathsf{W} = 1$ (whence $\mathsf{L} = 1$) and diagonal $\mathsf{H} = \left(\begin{smallmatrix} \mathbf{\Lambda} & 0 \\ 0 & 0 \end{smallmatrix}\right)$, we have $\mathsf{G} = \left(\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right)$ and $\mathsf{g} = \left(\begin{smallmatrix} \mathsf{g}' \\ 0 \end{smallmatrix}\right)$ as $\mathsf{g}^\top\,\mathsf{G} = 0$. Any gauge $\mathcal{G}$ transversal to G has the form $\overline{\mathsf{D}} = (-\overline{\mathsf{B}}\;\;\overline{\mathsf{C}})$ with nonsingular $\overline{\mathsf{C}}$. Premultiplying by $\overline{\mathsf{C}}^{-1}$ reduces $\overline{\mathsf{D}}$ to the form $\mathsf{D} = (-\mathsf{B}\;\;1)$ for some $n_{\mathsf{g}} \times (n_{\mathsf{x}} - n_{\mathsf{g}})$ matrix B. It follows that $\mathsf{P}_{\mathcal{G}} = \left(\begin{smallmatrix} 1 & 0 \\ \mathsf{B} & 0 \end{smallmatrix}\right)$ and $\mathsf{V}_{\mathcal{G}} = \left(\begin{smallmatrix} 1 \\ \mathsf{B} \end{smallmatrix}\right) \mathbf{\Lambda}^{-1}\,(1\;\;\mathsf{B}^\top)$, whence $\delta\mathsf{x}_{\mathcal{G}}^\top\,\mathsf{W}\,\delta\mathsf{x}_{\mathcal{G}} = \mathsf{g}^\top\,\mathsf{V}_{\mathcal{G}}\,\mathsf{W}\,\mathsf{V}_{\mathcal{G}}\,\mathsf{g} = \mathsf{g}'^\top\,\mathbf{\Lambda}^{-1}\,(1 + \mathsf{B}^\top\,\mathsf{B})\,\mathbf{\Lambda}^{-1}\,\mathsf{g}'$ and $\mathrm{Trace}(\mathsf{V}_{\mathcal{G}}) = \mathrm{Trace}(\mathbf{\Lambda}^{-1}) + \mathrm{Trace}\big(\mathsf{B}\,\mathbf{\Lambda}^{-1}\,\mathsf{B}^\top\big)$. Both criteria are clearly minimized by taking $\mathsf{B} = 0$, so $\mathsf{D} = (0\;\;1) = \mathsf{G}^\top\,\mathsf{W}$ as claimed. For nonsingular $\mathsf{W} = \mathsf{L}\,\mathsf{L}^\top$, scaling the coordinates by $\mathsf{x} \to \mathsf{L}\,\mathsf{x}$ reduces us to $\mathsf{W} \to 1$, $\mathsf{g}^\top \to \mathsf{g}^\top\mathsf{L}^{-1}$ and $\mathsf{H} \to \mathsf{L}^{-1}\,\mathsf{H}\,\mathsf{L}^{-\top}$. Eigen-decomposition then reduces us to diagonal H. Neither transformation affects $\delta\mathsf{x}_{\mathcal{G}}^\top\,\mathsf{W}\,\delta\mathsf{x}_{\mathcal{G}}$ or $\mathrm{Trace}(\mathsf{W}\,\mathsf{V}_{\mathcal{G}})$, and back substituting gives the general result. For singular W, use a limiting argument on $\mathsf{D} = \mathsf{G}^\top\,\mathsf{W}$. Similarly, using $\mathsf{V}_{\mathcal{G}}$ as above, $\mathsf{B} \to 0$, and hence the inner constraint, minimizes the L_2 and Frobenius norms of $\mathsf{L}^\top\,\mathsf{V}_{\mathcal{G}}\,\mathsf{L}$. Indeed, by the interlacing property of eigenvalues [44, §8.1], $\mathsf{B} \to 0$ minimizes *any* strictly non-decreasing rotationally invariant function of $\mathsf{L}^\top\,\mathsf{V}_{\mathcal{G}}\,\mathsf{L}$ (*i.e.* any strictly non-decreasing function of its eigenvalues). □

[18] $\mathsf{G} \to \mathsf{T}\,\mathsf{G}$ implies that $\mathsf{D} \to \mathsf{D}\,\mathsf{T}^{-1}$, whence $\mathsf{V}_{\mathcal{G}} \to \mathsf{T}\,\mathsf{V}_{\mathcal{G}}\,\mathsf{T}^\top$, $\mathsf{P}_{\mathcal{G}} \to \mathsf{T}\,\mathsf{P}_{\mathcal{G}}\,\mathsf{T}^{-1}$, and $\delta\mathsf{x}_{\mathcal{G}} \to \mathsf{T}\,\delta\mathsf{x}_{\mathcal{G}}$. So $\delta\mathsf{x}_{\mathcal{G}}^\top\,\mathsf{W}\,\delta\mathsf{x}_{\mathcal{G}}$ and $\mathrm{Trace}(\mathsf{W}\,\mathsf{V}_{\mathcal{G}})$ are preserved.

[19] This means that it vanishes identically for all non-point features, camera parameters, *etc.*, and is a weighted identity matrix $\mathsf{W}_i = w_i\,\boldsymbol{I}_{3\times 3}$ for each 3D point, or more generally it has the form $\overline{\mathsf{W}} \otimes \boldsymbol{I}_{3\times 3}$ on the block of 3D point coordinates, where $\overline{\mathsf{W}}$ is some $n_{\text{points}} \times n_{\text{points}}$ inter-point weighting matrix.

fixed weight matrix that takes account of the fact that the covariances of features, camera translations and rotations, focal lengths, aspect ratios and lens distortions, all have entirely different units, scales and relative importances. For $\mathsf{W} = \mathsf{1}$, the gauge projection $\mathsf{P}_\mathcal{G}$ becomes orthogonal and symmetric.

9.4 Free Networks

Gauges can be divided roughly into **outer gauges**, which are locked to predefined external reference features giving a **fixed network** adjustment, and **inner gauges**, which are locked only to the recovered structure giving a **free network** adjustment. (If their weight W is concentrated on the external reference, the inner constraints give an outer gauge). As above, well-chosen inner gauges do not distort the intrinsic covariance structure so much as most outer ones, so they tend to have better numerical conditioning and give a more representative idea of the true accuracy of the network. It is also useful to make another, slightly different fixed / free distinction. In order to control the gauge deficiency, any gauge fixing method must at least specify which motions are *locally* possible at each iteration. However, it is not indispensable for these local decisions to cohere to enforce a global gauge. A method is **globally fixed** if it does enforce a global gauge (whether inner or outer), and **globally free** if not. For example, the standard photogrammetric inner constraints [87,89,22,25] give a globally free inner gauge. They require that the cloud of reconstructed points should not be translated, rotated or rescaled under perturbations (*i.e*. the centroid and average directions and distances from the centroid remain unchanged). However, they do not specify where the cloud actually is and how it is oriented and scaled, and they do not attempt to correct for any gradual drift in the position that may occur during the optimization iterations, *e.g*. owing to accumulation of truncation errors. In contrast, McLauchlan globally fixes the inner gauge by locking it to the reconstructed centroid and scatter matrix [82,81]. This seems to give good numerical properties (although more testing is required to determine whether there is much improvement over a globally free inner gauge), and it has the advantage of actually fixing the coordinate system so that direct comparisons of solutions, covariances, *etc*., are possible. Numerically, a globally fixed gauge can be implemented either by including the 'd' term in (37), or simply by applying a rectifying gauge transformation to the estimate, at each step or when it drifts too far from the chosen gauge.

9.5 Implementation of Gauge Constraints

Given that all gauges are in principle equivalent, it does not seem worthwhile to pay a high computational cost for gauge fixing during step prediction, so methods requiring large dense factorizations or (pseudo-)inverses should not be used directly. Instead, the main computation can be done in any convenient, low cost gauge, and the results later transformed into the desired gauge using the gauge projector[20] $\mathsf{P}_\mathcal{G} = \mathsf{1} - \mathsf{G}\,(\mathsf{D}\,\mathsf{G})^{-1}\,\mathsf{D}$. It is probably easiest to use a trivial gauge for the computation. This is simply a matter of deleting the rows and columns of g, H corresponding to n_g preselected parameters, which should be chosen to give a reasonably well-conditioned gauge. The choice can be made automatically by a **subset selection** method (*c.f*., *e.g*. [11]). H is left intact and factored as usual, except that the final dense (owing to fill-in) submatrix is factored using a stable pivoted method, and the factorization is stopped n_g columns before completion. The remaining $n_\mathsf{g} \times n_\mathsf{g}$ block (and the corresponding block of the forward-substituted gradient g) should be zero owing to gauge deficiency. The corresponding rows of the state update are set to zero (or anything else that is wanted)

[20] The projector $\mathsf{P}_\mathcal{G}$ itself is never calculated. Instead, it is applied in pieces, multiplying by D, *etc*. The gauged Newton step $\delta\mathsf{x}_\mathcal{G}$ is easily found like this, and selected blocks of the covariance $\mathsf{V}_\mathcal{G} = \mathsf{P}_\mathcal{G}\,\mathsf{V}_{\mathcal{G}'}\,\mathsf{P}_\mathcal{G}^\top$ can also be found in this way, expanding $\mathsf{P}_\mathcal{G}$ and using (54) for the leading term, and for the remaining ones finding $\mathsf{L}^{-1}\,\mathsf{D}^\top$, *etc*., by forwards substitution.

and back-substitution gives the remaining update components as usual. This method effectively finds the n_{g} parameters that are least well constrained by the data, and chooses the gauge constraints that freeze these by setting the corresponding $\delta \mathsf{x}_{\mathcal{G}}$ components to zero.

10 Quality Control

This section discusses quality control methods for bundle adjustment, giving diagnostic tests that can be used to detect outliers and characterize the overall accuracy and reliability of the parameter estimates. These techniques are not well known in vision so we will go into some detail. Skip the technical details if you are not interested in them.

Quality control is a serious issue in measurement science, and it is perhaps here that the philosophical differences between photogrammetrists and vision workers are greatest: the photogrammetrist insists on good equipment, careful project planning, exploitation of prior knowledge and thorough error analyses, while the vision researcher advocates a more casual, flexible 'point-and-shoot' approach with minimal prior assumptions. Many applications demand a judicious compromise between these virtues.

A basic maxim is "quality = accuracy + reliability"[21]. The absolute *accuracy* of the system depends on the imaging geometry, number of measurements, *etc*. But theoretical precision by itself is not enough: the system must also be *reliable* in the face of outliers, small modelling errors, and so forth. The key to reliability is the intelligent use of *redundancy*: the results should represent an internally self-consistent consensus among many independent observations, no aspect of them should rely excessively on just a few observations.

The photogrammetric literature on quality control deserves to be better known in vision, especially among researchers working on statistical issues. Förstner [33,34] and Grün [49,50] give introductions with some sobering examples of the effects of poor design. See also [7,8,21,22]. All of these papers use least squares cost functions and scalar measurements. Our treatment generalizes this to allow robust cost functions and vector measurements, and is also slightly more self-consistent than the traditional approach. The techniques considered are useful for data analysis and reporting, and also to check whether design requirements are realistically attainable during project planning. Several properties should be verified. **Internal reliability** is the ability to detect and remove large aberrant observations using internal self-consistency checks. This is provided by traditional outlier detection and/or robust estimation procedures. **External reliability** is the extent to which any remaining *un*detected outliers can affect the estimates. **Sensitivity analysis** gives useful criteria for the quality of a design. Finally, **model selection tests** attempt to decide which of several possible models is most appropriate and whether certain parameters can be eliminated.

10.1 Cost Perturbations

We start by analyzing the approximate effects of adding or deleting an observation, which changes the cost function and hence the solution. We will use second order Taylor expansion to characterize the effects of this. Let $\mathsf{f}_-(\mathsf{x})$ and $\mathsf{f}_+(\mathsf{x}) \equiv \mathsf{f}_-(\mathsf{x}) + \delta\mathsf{f}(\mathsf{x})$ be respectively the total cost functions without and with the observation included, where $\delta\mathsf{f}(\mathsf{x})$ is the cost contribution of the observation itself. Let $\mathsf{g}_\pm, \delta\mathsf{g}$ be the gradients and $\mathsf{H}_\pm, \delta\mathsf{H}$ the Hessians of $\mathsf{f}_\pm, \delta\mathsf{f}$. Let x_0 be the unknown true

[21] 'Accuracy' is sometimes called 'precision' in photogrammetry, but we have preferred to retain the familiar meanings from numerical analysis: 'precision' means numerical error / number of working digits and 'accuracy' means statistical error / number of significant digits.

underlying state and $\mathsf{x}_\pm$ be the minima of $\mathsf{f}_\pm(\mathsf{x})$ (*i.e.* the optimal state estimates with and without the observation included). Residuals at x_0 are the most meaningful quantities for outlier decisions, but x_0 is unknown so we will be forced to use residuals at $\mathsf{x}_\pm$ instead. Unfortunately, as we will see below, these are biased. The bias is small for strong geometries but it can become large for weaker ones, so to produce uniformly reliable statistical tests we will have to correct for it. The fundamental result is: *For any sufficiently well behaved cost function, the* **difference in fitted residuals** $\mathsf{f}_+(\mathsf{x}_+) - \mathsf{f}_-(\mathsf{x}_-)$ *is asymptotically an unbiased and accurate estimate of* $\delta\mathsf{f}(\mathsf{x}_0)$ [22]:

$$\delta\mathsf{f}(\mathsf{x}_0) \;\approx\; \mathsf{f}_+(\mathsf{x}_+) - \mathsf{f}_-(\mathsf{x}_-) \;+\; \nu, \quad \nu \;\sim\; \mathcal{O}\big(\|\boldsymbol{\delta}\mathsf{g}\|/\sqrt{n_\mathsf{z} - n_\mathsf{x}}\big)\,, \quad \langle\nu\rangle \;\sim\; 0 \tag{40}$$

Note that by combining values at two known evaluation points $\mathsf{x}_\pm$, we simulate a value at a third unknown one x_0. The estimate is not perfect, but it is the best that we can do in the circumstances.

There are usually many observations to test, so to avoid having to refit the model many times we approximate the effects of adding or removing observations. Working at $\mathsf{x}_\pm$ and using the fact that $\mathsf{g}_\pm(\mathsf{x}_\pm) = \mathsf{0}$, the Newton step $\boldsymbol{\delta}\mathsf{x} \equiv \mathsf{x}_+ - \mathsf{x}_- \approx -\mathsf{H}_\mp^{-1}\boldsymbol{\delta}\mathsf{g}(\mathsf{x}_\pm)$ implies a change in fitted residual of:

$$\begin{aligned} \mathsf{f}_+(\mathsf{x}_+) - \mathsf{f}_-(\mathsf{x}_-) \;&\approx\; \delta\mathsf{f}(\mathsf{x}_\pm) \pm \tfrac{1}{2}\boldsymbol{\delta}\mathsf{x}^\top \mathsf{H}_\mp\, \boldsymbol{\delta}\mathsf{x} \\ =\; \delta\mathsf{f}(\mathsf{x}_\pm) \pm \tfrac{1}{2}\boldsymbol{\delta}\mathsf{g}(\mathsf{x}_\pm)^\top \mathsf{H}_\mp^{-1}\boldsymbol{\delta}\mathsf{g}(\mathsf{x}_\pm) \end{aligned} \tag{41}$$

So $\delta\mathsf{f}(\mathsf{x}_+)$ systematically underestimates $\mathsf{f}_+(\mathsf{x}_+) - \mathsf{f}_-(\mathsf{x}_-)$ and hence $\delta\mathsf{f}(\mathsf{x}_0)$ by about $\frac{1}{2}\boldsymbol{\delta}\mathsf{x}^\top \mathsf{H}_-\, \boldsymbol{\delta}\mathsf{x}$, and $\delta\mathsf{f}(\mathsf{x}_-)$ overestimates it by about $\frac{1}{2}\boldsymbol{\delta}\mathsf{x}^\top \mathsf{H}_+\, \boldsymbol{\delta}\mathsf{x}$. These biases are of order $\mathcal{O}(1/(n_\mathsf{z} - n_\mathsf{x}))$ and hence negligible when there is plenty of data, but they become large at low redundancies. Intuitively, including $\delta\mathsf{f}$ improves the estimate on average, bringing about a 'good' reduction of $\delta\mathsf{f}$, but it also overfits $\delta\mathsf{f}$ slightly, bringing about a further 'bad' reduction. Alternatively, the reduction in $\delta\mathsf{f}$ on moving from x_- to x_+ is bought at the cost of a slight increase in f_- (since x_- was already the minimum of f_-), which should morally also be 'charged' to $\delta\mathsf{f}$.

When deleting observations, we will usually have already evaluated H_+^{-1} (or a corresponding factorization of H_+) to find the Newton step near x_+, whereas (41) requires H_-^{-1}. And vice versa for addition. Provided that $\boldsymbol{\delta}\mathsf{H} \ll \mathsf{H}$, it is usually sufficient to use $\mathsf{H}_\pm^{-1}$ in place of $\mathsf{H}_\mp^{-1}$ in the simple tests below. However if the observation couples to relatively few state variables, it is possible to calculate the relevant components of $\mathsf{H}_\mp^{-1}$ fairly economically. If '$*$' means 'select the k variables on which $\boldsymbol{\delta}\mathsf{H}, \boldsymbol{\delta}\mathsf{g}$ are non-zero', then $\boldsymbol{\delta}\mathsf{g}^\top \mathsf{H}^{-1}\boldsymbol{\delta}\mathsf{g} = (\boldsymbol{\delta}\mathsf{g}^*)^\top (\mathsf{H}^{-1})^* \boldsymbol{\delta}\mathsf{g}^*$ and[23] $(\mathsf{H}_\mp^{-1})^* = \big(\big((\mathsf{H}_\pm^{-1})^*\big)^{-1} \mp \boldsymbol{\delta}\mathsf{H}^*\big)^{-1} \approx (\mathsf{H}_\pm^{-1})^* \pm (\mathsf{H}_\pm^{-1})^*\, \boldsymbol{\delta}\mathsf{H}^*\, (\mathsf{H}_\pm^{-1})^*$. Even without the approximation, this involves at most a $k \times k$ factorization or inverse. Indeed, for least squares $\boldsymbol{\delta}\mathsf{H}$ is usually of even lower rank (= the number of independent observations in $\delta\mathsf{f}$), so the Woodbury formula (18) can be used to calculate the inverse even more efficiently.

---

[22] *Sketch proof*: From the Newton steps $\boldsymbol{\delta}\mathsf{x}_\pm \equiv \mathsf{x}_\pm - \mathsf{x}_0 \approx -\mathsf{H}_\pm^{-1}\mathsf{g}_\pm(\mathsf{x}_0)$ at x_0, we find that $\mathsf{f}_\pm(\mathsf{x}_\pm) - \mathsf{f}_\pm(\mathsf{x}_0) \approx -\frac{1}{2}\boldsymbol{\delta}\mathsf{x}_\pm^\top \mathsf{H}_\pm\, \boldsymbol{\delta}\mathsf{x}_\pm$ and hence $\nu \equiv \mathsf{f}_+(\mathsf{x}_+) - \mathsf{f}_-(\mathsf{x}_-) - \delta\mathsf{f}(\mathsf{x}_0) \approx \frac{1}{2}\big(\boldsymbol{\delta}\mathsf{x}_-^\top \mathsf{H}_-\, \boldsymbol{\delta}\mathsf{x}_- - \boldsymbol{\delta}\mathsf{x}_+^\top \mathsf{H}_+\, \boldsymbol{\delta}\mathsf{x}_+\big)$. ν is unbiased to relatively high order: by the central limit property of ML estimators, the asymptotic distributions of $\boldsymbol{\delta}\mathsf{x}_\pm$ are Gaussian $\mathcal{N}(\mathsf{0}, \mathsf{H}_\pm^{-1})$, so the expectation of both $\boldsymbol{\delta}\mathsf{x}_\pm^\top \mathsf{H}_\pm\, \boldsymbol{\delta}\mathsf{x}_\pm$ is asymptotically the number of free model parameters n_x. Expanding $\boldsymbol{\delta}\mathsf{x}_\pm$ and using $\mathsf{g}_+ = \mathsf{g}_- + \boldsymbol{\delta}\mathsf{g}$, the leading term is $\nu \;\approx\; -\boldsymbol{\delta}\mathsf{g}(\mathsf{x}_0)^\top \mathsf{x}_-$, which asymptotically has normal distribution $\nu \sim \mathcal{N}(0, \boldsymbol{\delta}\mathsf{g}(\mathsf{x}_0)^\top \mathsf{H}_-^{-1}\boldsymbol{\delta}\mathsf{g}(\mathsf{x}_0))$ with standard deviation of order $\mathcal{O}\big(\|\boldsymbol{\delta}\mathsf{g}\|/\sqrt{n_\mathsf{z} - n_\mathsf{x}}\big)$, as $\mathsf{x}_- \sim \mathcal{N}(\mathsf{0}, \mathsf{H}_-^{-1})$ and $\|\mathsf{H}_-\| \sim \mathcal{O}(n_\mathsf{z} - n_\mathsf{x})$. □

[23] *C.f.* the lower right corner of (17), where the '$*$' components correspond to block 2, so that $\big((\mathsf{H}_\pm^{-1})^*\big)^{-1}$ is 'D_2', the Schur complement of the remaining variables in $\mathsf{H}_\pm$. Adding $\boldsymbol{\delta}\mathsf{H}^*$ changes the 'D' term but not the Schur complement correction.

10.2 Inner Reliability and Outlier Detection

In robust cost models nothing special needs to be done with outliers — they are just normal measurements that happen be downweighted owing to their large deviations. But in non-robust models such as least squares, explicit outlier detection and removal are essential for inner reliability. An effective diagnostic is to estimate $\delta\mathsf{f}(\mathsf{x}_0)$ using (40, 41), and significance-test it against its distribution under the null hypothesis that the observation is an inlier. For the least squares cost model, the null distribution of $2\,\delta\mathsf{f}(\mathsf{x}_0)$ is χ^2_k where k is the number of independent observations contributing to $\delta\mathsf{f}$. So if α is a suitable χ^2_k significance threshold, the typical one-sided significance test is:

$$\alpha \stackrel{?}{\leq} 2\,(\mathsf{f}(\mathsf{x}_+) - \mathsf{f}(\mathsf{x}_-)) \approx 2\,\delta\mathsf{f}(\mathsf{x}_\pm) \pm \boldsymbol{\delta}\mathsf{g}(\mathsf{x}_\pm)^\top\, \mathsf{H}_\mp^{-1}\, \boldsymbol{\delta}\mathsf{g}(\mathsf{x}_\pm) \tag{42}$$

$$\approx \triangle\mathsf{z}_i(\mathsf{x}_\pm)^\top \left(\mathsf{W}_i \pm \mathsf{W}_i\, \mathsf{J}_i^\top\, \mathsf{H}_\mp^{-1}\, \mathsf{J}_i\, \mathsf{W}_i\right) \triangle\mathsf{z}_i(\mathsf{x}_\pm) \tag{43}$$

As usual we approximate $\mathsf{H}_\mp^{-1} \approx \mathsf{H}_\pm^{-1}$ and use x_- results for additions and x_+ ones for deletions. These tests require the fitted covariance matrix $\mathsf{H}_\pm^{-1}$ (or, if relatively few tests will be run, an equivalent factorization of $\mathsf{H}_\pm$), but given this they are usually fairly economical owing to the sparseness of the observation gradients $\boldsymbol{\delta}\mathsf{g}(\mathsf{x}_\pm)$. Equation (43) is for the nonlinear least squares model with residual error $\triangle\mathsf{z}_i(\mathsf{x}) \equiv \underline{\mathsf{z}}_i - \mathsf{z}_i(\mathsf{x})$, cost $\frac{1}{2}\triangle\mathsf{z}_i(\mathsf{x})^\top\, \mathsf{W}_i\, \triangle\mathsf{z}_i(\mathsf{x})$ and Jacobian $\mathsf{J}_i = \frac{\mathsf{dz}_i}{\mathsf{dx}}$. Note that even though $\underline{\mathsf{z}}_i$ induces a change in *all* components of the observation residual $\triangle\mathsf{z}$ via its influence on $\boldsymbol{\delta}\mathsf{x}$, only the immediately involved components $\triangle\mathsf{z}_i$ are required in (43). The bias-correction-induced change of weight matrix $\mathsf{W}_i \to \mathsf{W}_i \pm \mathsf{W}_i\, \mathsf{J}_i^\top\, \mathsf{H}_\mp^{-1}\, \mathsf{J}_i\, \mathsf{W}_i$ accounts for the others. For non-quadratic cost functions, the above framework still applies but the cost function's native distribution of negative log likelihood values must be used instead of the Gaussian's $\frac{1}{2}\,\chi^2$.

In principle, the above analysis is only valid when at most one outlier causes a relatively small perturbation $\boldsymbol{\delta}\mathsf{x}$. In practice, the observations are repeatedly scanned for outliers, at each stage removing any discovered outliers (and perhaps reinstating previously discarded observations that have become inliers) and refitting. The net result is a form of M-estimator routine with an abruptly vanishing weight function: *outlier deletion is just a roundabout way of simulating a robust cost function*. (Hard inlier/outlier rules correspond to total likelihood functions that become strictly constant in the outlier region).

The tests (42, 43) give what is needed for outlier decisions based on *fitted* state estimates $\mathsf{x}_\pm$, but for planning purposes it is also useful to know how large a gross error must typically be w.r.t. the *true* state x_0 before it is detected. Outlier detection is based on the uncertain fitted state estimates, so we can only give an average case result. No adjustment for $\mathsf{x}_\pm$ is needed in this case, so the average **minimum detectable gross error** is simply:

$$\alpha \stackrel{?}{\leq} 2\,\delta\mathsf{f}(\mathsf{x}_0) \approx \triangle\mathsf{z}(\mathsf{x}_0)^\top\, \mathsf{W}\, \triangle\mathsf{z}(\mathsf{x}_0) \tag{44}$$

10.3 Outer Reliability

Ideally, the state estimate should be as insensitive as possible to any remaining errors in the observations. To estimate how much a particular observation influences the final state estimate, we can directly monitor the displacement $\boldsymbol{\delta}\mathsf{x} \equiv \mathsf{x}_+ - \mathsf{x}_- \approx \mathsf{H}_\mp^{-1}\, \boldsymbol{\delta}\mathsf{g}_\pm(\mathsf{x}_\pm)$. For example, we might define an importance weighting on the state parameters with a criterion matrix U and monitor absolute displacements $\|\mathsf{U}\,\boldsymbol{\delta}\mathsf{x}\| \approx \|\mathsf{U}\,\mathsf{H}_\mp^{-1}\,\boldsymbol{\delta}\mathsf{g}(\mathsf{x}_\pm)\|$, or compare the displacement $\boldsymbol{\delta}\mathsf{x}$ to the covariance $\mathsf{H}_\pm^{-1}$ of $\mathsf{x}_\pm$ by monitoring $\boldsymbol{\delta}\mathsf{x}^\top\, \mathsf{H}_\mp\, \boldsymbol{\delta}\mathsf{x} \approx \boldsymbol{\delta}\mathsf{g}_\pm(\mathsf{x}_\pm)^\top\, \mathsf{H}_\mp^{-1}\, \boldsymbol{\delta}\mathsf{g}_\pm(\mathsf{x}_\pm)$. A bound on $\boldsymbol{\delta}\mathsf{g}(\mathsf{x}_\pm)$ of the form[24]

[24] This is a convenient intermediate form for deriving bounds. For positive semidefinite matrices A, B, we say that B **dominates** A, $\mathsf{B} \succeq \mathsf{A}$, if $\mathsf{B} - \mathsf{A}$ is positive semidefinite. It follows that $\mathcal{N}(\mathsf{U}\,\mathsf{A}\,\mathsf{U}^\top) \leq \mathcal{N}(\mathsf{U}\,\mathsf{B}\,\mathsf{U}^\top)$ for any matrix U and

$\boldsymbol{\delta}\mathsf{g}\,\boldsymbol{\delta}\mathsf{g}^\top \preceq \mathsf{V}$ for some positive semidefinite V implies a bound $\boldsymbol{\delta}\mathsf{x}\,\boldsymbol{\delta}\mathsf{x}^\top \preceq \mathsf{H}_\mp^{-1}\mathsf{V}\mathsf{H}_\mp^{-1}$ on $\boldsymbol{\delta}\mathsf{x}$ and hence a bound $\|\mathsf{U}\,\boldsymbol{\delta}\mathsf{x}\|^2 \leq \mathcal{N}(\mathsf{U}\mathsf{H}_\mp^{-1}\mathsf{V}\mathsf{H}_\mp^{-1}\mathsf{U}^\top)$ where $\mathcal{N}(\cdot)$ can be L_2 norm, trace or Frobenius norm. For a robust cost model in which $\boldsymbol{\delta}\mathsf{g}$ is globally bounded, this already gives asymptotic bounds of order $\mathcal{O}(\|\mathsf{H}^{-1}\|\|\boldsymbol{\delta}\mathsf{g}\|) \sim \mathcal{O}(\|\boldsymbol{\delta}\mathsf{g}\|/\sqrt{n_\mathsf{z}-n_\mathsf{x}})$ for the state perturbation, regardless of whether an outlier occurred. For non-robust cost models we have to use an inlier criterion to limit $\boldsymbol{\delta}\mathsf{g}$. For the least squares observation model with rejection test (43), $\triangle\mathsf{z}\,\triangle\mathsf{z}^\top \preceq \alpha\left(\mathsf{W}_i \pm \mathsf{W}_i\,\mathsf{J}_i^\top\mathsf{H}_\mp^{-1}\mathsf{J}_i\,\mathsf{W}_i\right)^{-1}$ and hence the maximum state perturbation due to a declared-inlying observation $\underline{\mathsf{z}}_i$ is:

$$\begin{aligned}
\boldsymbol{\delta}\mathsf{x}\,\boldsymbol{\delta}\mathsf{x}^\top &\preceq \alpha\,\mathsf{H}_\mp^{-1}\mathsf{J}_i\,\mathsf{W}_i\left(\mathsf{W}_i \pm \mathsf{W}_i\,\mathsf{J}_i^\top\mathsf{H}_\mp^{-1}\mathsf{J}_i\,\mathsf{W}_i\right)^{-1}\mathsf{W}_i\,\mathsf{J}_i^\top\mathsf{H}_\mp^{-1} \\
&= \alpha\left(\mathsf{H}_-^{-1} - \mathsf{H}_+^{-1}\right) \qquad (45)\\
&\approx \alpha\,\mathsf{H}_\pm^{-1}\mathsf{J}_i\,\mathsf{W}_i^{-1}\mathsf{J}_i^\top\mathsf{H}_\pm^{-1} \qquad (46)
\end{aligned}$$

so, *e.g.*, $\boldsymbol{\delta}\mathsf{x}^\top\mathsf{H}_\pm\,\boldsymbol{\delta}\mathsf{x} \leq \alpha\,\mathrm{Trace}\left(\mathsf{J}_i\,\mathsf{H}_\pm^{-1}\mathsf{J}_i^\top\,\mathsf{W}_i^{-1}\right)$ and $\|\mathsf{U}\,\boldsymbol{\delta}\mathsf{x}\|^2 \leq \alpha\,\mathrm{Trace}\left(\mathsf{J}_i\,\mathsf{H}_\pm^{-1}\mathsf{U}^\top\mathsf{U}\,\mathsf{H}_\pm^{-1}\mathsf{J}_i^\top\,\mathsf{W}_i^{-1}\right)$, where W_i^{-1} is the nominal covariance of $\underline{\mathsf{z}}_i$. Note that these bounds are based on changes in the *estimated* state $\mathsf{x}_\pm$. They do not directly control perturbations w.r.t. the *true* one x_0. The combined influence of several ($k \ll n_\mathsf{z} - n_\mathsf{x}$) observations is given by summing their $\boldsymbol{\delta}\mathsf{g}$'s.

10.4 Sensitivity Analysis

This section gives some simple figures of merit that can be used to quantify network redundancy and hence reliability. First, in $\delta\mathsf{f}(\mathsf{x}_0) \approx \delta\mathsf{f}(\mathsf{x}_+) + \frac{1}{2}\,\boldsymbol{\delta}\mathsf{g}(\mathsf{x}_+)^\top\mathsf{H}_-^{-1}\boldsymbol{\delta}\mathsf{g}(\mathsf{x}_+)$, each cost contribution $\delta\mathsf{f}(\mathsf{x}_0)$ is split into two parts: the *visible residual* $\delta\mathsf{f}(\mathsf{x}_+)$ at the fitted state x_+; and $\frac{1}{2}\,\boldsymbol{\delta}\mathsf{x}^\top\mathsf{H}_-\,\boldsymbol{\delta}\mathsf{x}$, the *change in the base cost* $\mathsf{f}_-(\mathsf{x})$ due to the state perturbation $\boldsymbol{\delta}\mathsf{x} = \mathsf{H}_-^{-1}\boldsymbol{\delta}\mathsf{g}(\mathsf{x}_+)$ induced by the observation. Ideally, we would like the state perturbation to be small (for stability) and the residual to be large (for outlier detectability). In other words, we would like the following **masking factor** to be small ($m_i \ll 1$) for each observation:

$$\begin{aligned}
m_i &\equiv \frac{\boldsymbol{\delta}\mathsf{g}(\mathsf{x}_+)^\top\mathsf{H}_-^{-1}\boldsymbol{\delta}\mathsf{g}(\mathsf{x}_+)}{2\,\delta\mathsf{f}(\mathsf{x}_+) + \boldsymbol{\delta}\mathsf{g}(\mathsf{x}_+)^\top\mathsf{H}_-^{-1}\boldsymbol{\delta}\mathsf{g}(\mathsf{x}_+)} \qquad (47)\\
&= \frac{\triangle\mathsf{z}_i(\mathsf{x}_+)^\top\mathsf{W}_i\,\mathsf{J}_i\,\mathsf{H}_-^{-1}\mathsf{J}_i^\top\,\mathsf{W}_i\,\triangle\mathsf{z}_i(\mathsf{x}_+)}{\triangle\mathsf{z}_i(\mathsf{x}_+)^\top\left(\mathsf{W}_i + \mathsf{W}_i\,\mathsf{J}_i\,\mathsf{H}_-^{-1}\mathsf{J}_i^\top\,\mathsf{W}_i\right)\triangle\mathsf{z}_i(\mathsf{x}_+)} \qquad (48)
\end{aligned}$$

(Here, $\delta\mathsf{f}$ should be normalized to have minimum value 0 for an exact fit). If m_i is known, the outlier test becomes $\delta\mathsf{f}(\mathsf{x}_+)/(1-m_i) \geq \alpha$. The masking m_i depends on the relative size of $\boldsymbol{\delta}\mathsf{g}$ and $\delta\mathsf{f}$, which in general depends on the functional form of $\delta\mathsf{f}$ and the specific deviation involved. For robust cost models, a bound on $\boldsymbol{\delta}\mathsf{g}$ may be enough to bound m_i for outliers. However, for least squares case ($\triangle\mathsf{z}$ form), and more generally for quadratic cost models (such as robust models near the origin), m_i depends only on the direction of $\triangle\mathsf{z}_i$, not on its size, and we have a global L_2 matrix norm based bound $m_i \leq \frac{\nu}{1+\nu}$ where $\nu = \|\mathsf{L}^\top\mathsf{J}_i\,\mathsf{H}_-^{-1}\mathsf{J}_i^\top\,\mathsf{L}\|_2 \leq \mathrm{Trace}\left(\mathsf{J}_i\,\mathsf{H}_-^{-1}\mathsf{J}_i^\top\,\mathsf{W}\right)$ and $\mathsf{L}\,\mathsf{L}^\top = \mathsf{W}_i$ is a Cholesky decomposition of W_i. (These bounds become equalities for scalar observations).

The stability of the state estimate is determined by the total cost Hessian (information matrix) H. A large H implies a small state estimate covariance H^{-1} and also small responses $\boldsymbol{\delta}\mathsf{x} \approx -\mathsf{H}^{-1}\boldsymbol{\delta}\mathsf{g}$

any matrix function $\mathcal{N}(\cdot)$ that is non-decreasing under positive additions. Rotationally invariant non-decreasing functions $\mathcal{N}(\cdot)$ include all non-decreasing functions of the eigenvalues, *e.g.* L_2 norm $\max\lambda_i$, trace $\sum\lambda_i$, Frobenius norm $\sqrt{\sum\lambda_i^2}$. For a vector a and positive B, $\mathsf{a}^\top\mathsf{B}\,\mathsf{a} \leq k$ if and only if $\mathsf{a}\,\mathsf{a}^\top \preceq k\,\mathsf{B}^{-1}$. (*Proof:* Conjugate by $\mathsf{B}^{1/2}$ and then by a $(\mathsf{B}^{1/2}\,\mathsf{a})$-reducing Householder rotation to reduce the question to the equivalence of $\mathsf{0} \preceq \mathrm{Diag}\left(k-u^2, k, \ldots, k\right)$ and $u^2 \leq k$, where $u^2 = \|\mathsf{B}^{1/2}\,\mathsf{a}\|^2$). Bounds of the form $\|\mathsf{U}\,\mathsf{a}\|^2 \leq k\,\mathcal{N}(\mathsf{U}\mathsf{B}^{-1}\mathsf{U}^\top)$ follow for any U and any $\mathcal{N}(\cdot)$ for which $\mathcal{N}(\mathsf{v}\,\mathsf{v}^\top) = \|\mathsf{v}\|^2$, *e.g.* L_2 norm, trace, Frobenius norm.

to cost perturbations $\delta\mathsf{g}$. The **sensitivity numbers** $s_i \equiv \text{Trace}\big(\mathsf{H}_+^{-1}\delta\mathsf{H}_i\big)$ are a useful measure of the relative amount of information contributed to H_+ by each observation. They sum to the model dimension — $\sum_i s_i = n_\mathsf{x}$ because $\sum_i \delta\mathsf{H}_i = \mathsf{H}_+$ — so they count "how many parameters worth" of the total information the observation contributes. Some authors prefer to quote **redundancy numbers** $r_i \equiv n_i - s_i$, where n_i is the effective number of independent observations contained in $\underline{\mathsf{z}}_i$. The redundancy numbers sum to $n_\mathsf{z} - n_\mathsf{x}$, the total redundancy of the system. In the least squares case, $s_i = \text{Trace}\big(\mathsf{J}_i\,\mathsf{H}_+^{-1}\,\mathsf{J}_i^\top\,\mathsf{W}\big)$ and $m_i = s_i$ for scalar observations, so the scalar outlier test becomes $\delta\mathsf{f}(\mathsf{x}_+)/r_i \geq \alpha$. Sensitivity numbers can also be defined for subgroups of the parameters in the form $\text{Trace}(\mathsf{U}\,\mathsf{H}^{-1}\,\delta\mathsf{H})$, where U is an orthogonal projection matrix that selects the parameters of interest. Ideally, the sensitivities of each subgroup should be spread evenly across the observations: a large s_i indicates a heavily weighted observation, whose incorrectness might significantly compromise the estimate.

10.5 Model Selection

It is often necessary to chose between several alternative models of the cameras or scene, *e.g.* additional parameters for lens distortion, camera calibrations that may or may not have changed between images, coplanarity or non-coplanarity of certain features. Over-special models give biased results, while over-general ones tend to be noisy and unstable. We will consider only **nested models**, for which a more general model is specialized to a more specific one by freezing some of its parameters at default values (*e.g.* zero skew or lens distortion, equal calibrations, zero deviation from a plane). Let: x be the parameter vector of the more general model; $\mathsf{f}(\mathsf{x})$ be its cost function; $\mathsf{c}(\mathsf{x}) = \mathsf{0}$ be the parameter freezing constraints enforcing the specialization; k be the number of parameters frozen; x_0 be the true underlying state; x_g be the optimal state estimate for the general model (*i.e.* the unconstrained minimum of $\mathsf{f}(\mathsf{x})$); and x_s be the optimal state estimate for the specialized one (*i.e.* the minimum of $\mathsf{f}(\mathsf{x})$ subject to the constraints $\mathsf{c}(\mathsf{x}) = \mathsf{0}$). Then, under the null hypothesis that the specialized model is correct, $\mathsf{c}(\mathsf{x}_0) = \mathsf{0}$, and in the asymptotic limit in which $\mathsf{x}_g - \mathsf{x}_0$ and $\mathsf{x}_s - \mathsf{x}_0$ become Gaussian and the constraints become locally approximately linear across the width of this Gaussian, the difference in fitted residuals $2\,(\mathsf{f}(\mathsf{x}_s) - \mathsf{f}(\mathsf{x}_g))$ has a χ^2_k distribution[25]. So if $2\,(\mathsf{f}(\mathsf{x}_s) - \mathsf{f}(\mathsf{x}_g))$ is less than some suitable χ^2_k decision threshold α, we can accept the hypothesis that the additional parameters take their default values, and use the specialized model rather than the more general one[26].

As before, we can avoid fitting one of the models by using a linearized analysis. First suppose that we start with a fit of the more general model x_g. Let the linearized constraints at x_g be $\mathsf{c}(\mathsf{x}_g + \delta\mathsf{x}) \approx \mathsf{c}(\mathsf{x}_g) + \mathsf{C}\,\delta\mathsf{x}$, where $\mathsf{C} \equiv \frac{\mathsf{dc}}{\mathsf{dx}}$. A straightforward Lagrange multiplier calculation gives:

$$\begin{aligned} 2\,(\mathsf{f}(\mathsf{x}_s) - \mathsf{f}(\mathsf{x}_g)) &\approx \mathsf{c}(\mathsf{x}_g)^\top\,(\mathsf{C}\,\mathsf{H}^{-1}\,\mathsf{C}^\top)^{-1}\,\mathsf{c}(\mathsf{x}_g) \\ \mathsf{x}_s &\approx \mathsf{x}_g - \mathsf{H}^{-1}\,\mathsf{C}^\top\,(\mathsf{C}\,\mathsf{H}^{-1}\,\mathsf{C}^\top)^{-1}\,\mathsf{c}(\mathsf{x}_g) \end{aligned} \tag{49}$$

Conversely, starting from a fit of the more specialized model, the unconstrained minimum is given by the Newton step: $\mathsf{x}_g \approx \mathsf{x}_s - \mathsf{H}^{-1}\mathsf{g}(\mathsf{x}_s)$, and $2\,(\mathsf{f}(\mathsf{x}_s) - \mathsf{f}(\mathsf{x}_g)) \approx \mathsf{g}(\mathsf{x}_s)^\top\,\mathsf{H}^{-1}\,\mathsf{g}(\mathsf{x}_s)$, where $\mathsf{g}(\mathsf{x}_s)$ is the residual cost gradient at x_s. This requires the general-model covariance H^{-1} (or an equivalent factorization of H), which may not have been worked out. Suppose that the additional parameters

[25] This happens irrespective of the observation distributions because — unlike the case of adding an observation — the same observations and cost function are used for both fits.

[26] In practice, small models are preferable as they have greater stability and predictive power and less computational cost. So the threshold α is usually chosen to be comparatively large, to ensure that the more general model will not be chosen unless there is strong evidence for it.

were simply appended to the model, $\mathsf{x} \rightarrow (\mathsf{x}, \mathsf{y})$ where x is now the reduced parameter vector of the specialized model and y contains the additional parameters. Let the general-model cost gradient at $(\mathsf{x}_s, \mathsf{y}_s)$ be $\left(\begin{smallmatrix} 0 \\ \mathsf{h} \end{smallmatrix}\right)$ where $\mathsf{h} = \frac{\mathsf{df}}{\mathsf{dy}}$, and its Hessian be $\left(\begin{smallmatrix} \mathsf{H} & \mathsf{A}^\top \\ \mathsf{A} & \mathsf{B} \end{smallmatrix}\right)$. A straightforward calculation shows that:

$$\begin{aligned} 2\,(\mathsf{f}(\mathsf{x}_s, \mathsf{y}_s) - \mathsf{f}(\mathsf{x}_g, \mathsf{y}_g)) \;&\approx\; \mathsf{h}^\top\,(\mathsf{B} - \mathsf{A}\,\mathsf{H}^{-1}\mathsf{A}^\top)^{-1}\,\mathsf{h} \\ \left(\begin{smallmatrix} \mathsf{x}_g \\ \mathsf{y}_g \end{smallmatrix}\right) \;&\approx\; \left(\begin{smallmatrix} \mathsf{x}_s \\ \mathsf{y}_s \end{smallmatrix}\right) + \left(\begin{smallmatrix} \mathsf{H}^{-1}\mathsf{A}^\top \\ -1 \end{smallmatrix}\right)(\mathsf{B} - \mathsf{A}\,\mathsf{H}^{-1}\mathsf{A}^\top)^{-1}\,\mathsf{h} \end{aligned} \tag{50}$$

Given H^{-1} or an equivalent factorization of H, these tests are relatively inexpensive for small k. They amount respectively to one step of Sequential Quadratic Programming and one Newton step, so the results will only be accurate when these methods converge rapidly.

Another, softer, way to handle nested models is to apply a prior $\delta\mathsf{f}_{\text{prior}}(\mathsf{x})$ peaked at the zero of the specialization constraints $\mathsf{c}(\mathsf{x})$. If this is weak the data will override it when necessary, but the constraints may not be very accurately enforced. If it is stronger, we can either apply an 'outlier' test (40, 42) to remove it if it appears to be incorrect, or use a **sticky prior** — a prior similar to a robust distribution, with a concentrated central peak and wide flat tails, that will hold the estimate near the constraint surface for weak data, but allow it to 'unstick' if the data becomes stronger.

Finally, more heuristic rules are often used for model selection in photogrammetry, for example deleting any additional parameters that are excessively correlated (correlation coefficient greater than ~ 0.9) with other parameters, or whose introduction appears to cause an excessive increase in the covariance of other parameters [49,50].

11 Network Design

Network design is the problem of planning camera placements and numbers of images before a measurement project, to ensure that sufficiently accurate and reliable estimates of everything that needs to be measured are found. We will not say much about design, merely outlining the basic considerations and giving a few useful rules of thumb. See [5, chapter 6], [79,78], [73, Vol.2 §4] for more information.

Factors to be considered in network design include: scene coverage, occlusion / visibility and feature viewing angle; field of view, depth of field, resolution and workspace constraints; and geometric strength, accuracy and redundancy. The basic quantitative aids to design are covariance estimation in a suitably chosen gauge (see §9) and the quality control tests from §10. Expert systems have been developed [79], but in practice most designs are still based on personal experience and rules of thumb.

In general, geometric stability is best for 'convergent' (close-in, wide baseline, high perspective) geometries, using wide angle lenses to cover as much of the object as possible, and large film or CCD formats to maximize measurement precision. The wide coverage maximizes the overlap between different sub-networks and hence overall network rigidity, while the wide baselines maximize the sub-network stabilities. The practical limitations on closeness are workspace, field of view, depth of field, resolution and feature viewing angle constraints.

Maximizing the overlap between sub-networks is very important. For objects with several faces such as buildings, images should be taken from corner positions to tie the face sub-networks together. For large projects, large scale overview images can be used to tie together close-in densifying ones. When covering individual faces or surfaces, overlap and hence stability are improved by taking images with a range of viewing angles rather than strictly fronto-parallel ones (*e.g.*, for the same number of images, pan-move-pan-move or interleaved left-looking and right-looking images are stabler than

a simple fronto-parallel track). Similarly, for buildings or turntable sequences, using a mixture of low and high viewpoints helps stability.

For reliability, one usually plans to see each feature point in at least four images. Although two images in principle suffice for reconstruction, they offer little redundancy and no resistance against feature extraction failures. Even with three images, the internal reliability is still poor: isolated outliers can usually be detected, but it may be difficult to say which of the three images they occurred in. Moreover, 3–4 image geometries with widely spaced (*i.e.* non-aligned) centres usually give much more isotropic feature error distributions than two image ones.

If the bundle adjustment will include self-calibration, it is important to include a range of viewing angles. For example for a flat, compact object, views might be taken at regularly spaced points along a 30–45° half-angle cone centred on the object, with 90° optical axis rotations between views.

12 Summary and Recommendations

This survey was written in the hope of making photogrammetric know-how about bundle adjustment — the simultaneous optimization of structure and camera parameters in visual reconstruction — more accessible to potential implementors in the computer vision community. Perhaps the main lessons are the extraordinary versatility of adjustment methods, the critical importance of exploiting the problem structure, and the continued dominance of second order (Newton) algorithms, in spite of all efforts to make the simpler first order methods converge more rapidly.

We will finish by giving a series of recommendations for methods. At present, these must be regarded as very provisional, and subject to revision after further testing.

Parametrization: (§2.2, 4.5) During step prediction, avoid parameter singularities, infinities, strong nonlinearities and ill-conditioning. Use well-conditioned local (current value + offset) parametrizations of nonlinear elements when necessary to achieve this: the local step prediction parametrization can be different from the global state representation one. The ideal is to make the parameter space error function as isotropic and as near-quadratic as possible. Residual rotation or quaternion parametrizations are advisable for rotations, and projective homogeneous parametrizations for distant points, lines and planes (*i.e.* 3D features near the singularity of their affine parametrizations, affine infinity).

Cost function: (§3) The cost should be a realistic approximation to the negative log likelihood of the total (inlier + outlier) error distribution. The exact functional form of the distribution is not too critical, however: (*i*) Undue weight should not be given to outliers by making the tails of the distribution (the predicted probability of outliers) unrealistically small. (NB: Compared to most real-world measurement distributions, the tails of a Gaussian *are* unrealistically small). (*ii*) The dispersion matrix or inlier covariance should be a realistic estimate of the actual inlier measurement dispersion, so that the transition between inliers and outliers is in about the right place, and the inlier errors are correctly weighted during fitting.

Optimization method: (§4, 6, 7) For **batch problems** use a second order Gauss-Newton method with sparse factorization (see below) of the Hessian, unless:

- The problem is so large that exact sparse factorization is impractical. In this case consider either iterative linear system solvers such as Conjugate Gradient for the Newton step, or related nonlinear iterations such as Conjugate Gradient, or preferably Limited Memory Quasi-Newton or (if memory permits) full Quasi-Newton (§7, [29, 93, 42]). (None of these methods require the Hessian). If you are in this case, it would pay to investigate professional large-scale optimization codes such as MINPACK-2, LANCELOT, or commercial methods from NAG or IMSL (see §C.2).

- If the problem is medium or large but dense (which is unusual), and if it has strong geometry, alternation of resection and intersection may be preferable to a second order method. However, in this case Successive Over-Relaxation (SOR) would be even better, and Conjugate Gradient is likely to be better yet.
- In all of the above cases, good preconditioning is critical (§7.3).

For **on-line problems** (rather than batch ones), use factorization updating rather than matrix inverse updating or re-factorization (§B.5). In time-series problems, investigate the effect of changing the time window (§8.2, [83, 84]), and remember that Kalman filtering is only the first half-iteration of a full nonlinear method.

Factorization method: (§6.2, B.1) For speed, preserve the symmetry of the Hessian during factorization by using: Cholesky decomposition for positive definite Hessians (*e.g.* unconstrained problems in a trivial gauge); pivoted Cholesky decomposition for positive semi-definite Hessians (*e.g.* unconstrained problems with gauge fixing by subset selection §9.5); and Bunch-Kauffman decomposition (§B.1) for indefinite Hessians (*e.g.* the augmented Hessians of constrained problems, §4.4). Gaussian elimination is stable but a factor of two slower than these.

Variable ordering: (§6.3) The variables can usually be ordered by hand for regular networks, but for more irregular ones (*e.g.* close range site-modelling) some experimentation may be needed to find the most efficient overall ordering method. If reasonably compact profiles can be found, profile representations (§6.3.3, B.3) are simpler to implement and faster than general sparse ones (§6.3).

- For dense networks use a profile representation and a "natural" variable ordering: either features then cameras, or cameras then features, with whichever has the fewest parameters last. An explicit reduced system based implementation such as Brown's method [19] can also be used in this case (§6.1, A).
- If the problem has some sort of 1D temporal or spatial structure (*e.g.* image streams, turntable problems), try a profile representation with natural (simple connectivity) or Snay's banker's (more complex connectivity) orderings (§6.3.3, [101, 24]). A recursive on-line updating method might also be useful in this case.
- If the problem has 2D structure (*e.g.* cartography and other surface coverage problems) try nested dissection, with hand ordering for regular problems (cartographic blocks), and a multilevel scheme for more complex ones (§6.3.2). A profile representation may or may not be suitable.
- For less regular sparse networks, the choice is not clear. Try minimum degree ordering with a general sparse representation, Snay's Banker's with a profile representation, or multilevel nested dissection.

For all of the automatic variable ordering methods, try to order any especially highly connected variables last by hand, before invoking the method.

Gauge fixing: (§9) For efficiency, use either a trivial gauge or a subset selection method as a working gauge for calculations, and project the results into whatever gauge you want later by applying a suitable gauge projector $\mathsf{P}_{\mathcal{G}}$ (33). Unless you have a strong reason to use an external reference system, the output gauge should probably be an inner gauge centred on the network elements you care most about, *i.e.* the observed features for a reconstruction problem, and the cameras for a navigation one.

Quality control and network design: (§10) A robust cost function helps, but for overall system reliability you still need to plan your measurements in advance (until you have developed a good intuition for this), and check the results afterwards for outlier sensitivity and over-modelling, using a

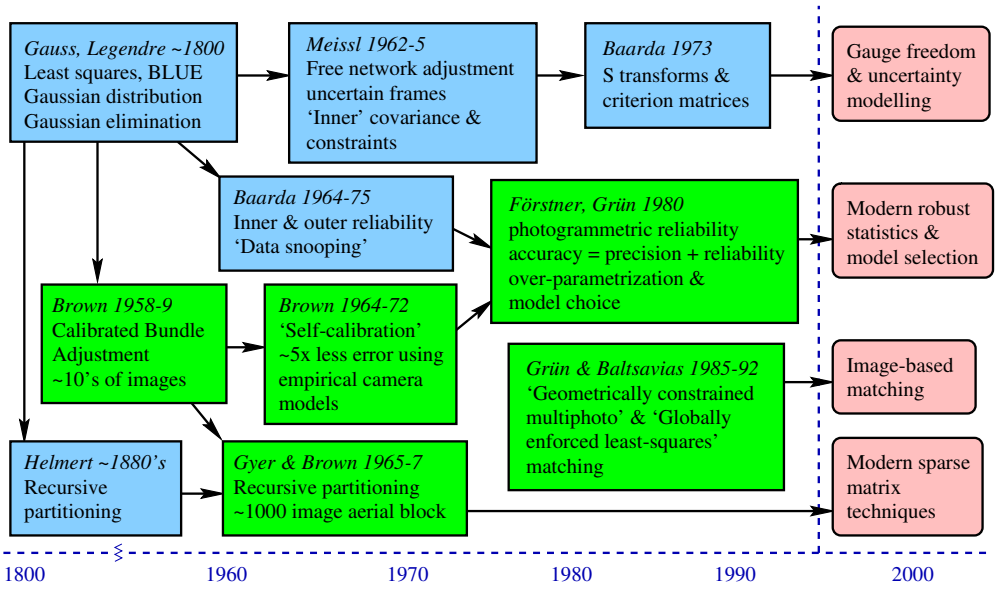


Figure 9: A schematic history of bundle adjustment.

suitable quality control procedure. Do not underestimate the extent to which either low redundancy, or weak geometry, or over-general models can make gross errors undetectable.

A Historical Overview

This appendix gives a brief history of the main developments in bundle adjustment, including literature references.

Least squares: The theory of combining measurements by minimizing the sum of their squared residuals was developed independently by Gauss and Legendre around 1795–1820 [37,74], [36, Vol.IV, 1–93], about 40 years *after* robust L_1 estimation [15]. Least squares was motivated by estimation problems in astronomy and geodesy and extensively applied to both fields by Gauss, whose remarkable 1823 monograph [37,36] already contains almost the complete modern theory of least squares including elements of the theory of probability distributions, the definition and properties of the Gaussian distribution, and a discussion of bias and the "Gauss-Markov" theorem, which states that least squares gives the Best Linear Unbiased Estimator (BLUE) [37,11]. It also introduces the $\mathsf{LDL}^\top$ form of symmetric Gaussian elimination and the Gauss-Newton iteration for nonlinear problems, essentially in their modern forms although without explicitly using matrices. The 1828 supplement on geodesy introduced the Gauss-Seidel iteration for solving large nonlinear systems. The economic and military importance of surveying lead to extensive use of least squares and several further developments: Helmert's nested dissection [64] — probably the first systematic sparse matrix method — in the 1880's, Cholesky decomposition around 1915, Baarda's theory of reliability of measurement networks in the 1960's [7,8], and Meissl [87,89] and Baarda's [6] theories of uncertain coordinate frames and free networks [22,25]. We will return to these topics below.

Second order bundle algorithms: Electronic computers capable of solving reasonably large least squares problems first became available in the late 1950's. The basic photogrammetric bundle method was developed for the U.S. Air Force by Duane C. Brown and his co-workers in 1957–9 [16,19].

The initial focus was aerial cartography, but by the late 1960's bundle methods were also being used for close-range measurements[27]. The links with geodesic least squares and the possibility of combining geodesic and other types of measurements with the photogrammetric ones were clear right from the start. Initially the cameras were assumed to be calibrated[28], so the optimization was over object points and camera poses only. **Self calibration** (the estimation of internal camera parameters during bundle adjustment) was first discussed around 1964 and implemented by 1968 [19]. Camera models were greatly refined in the early 1970's, with the investigation of many alternative sets of **additional (distortion) parameters** [17,18,19]. Even with stable and carefully calibrated aerial photogrammetry cameras, self calibration significantly improved accuracies (by factors of around 2–10). This lead to rapid improvements in camera design as previously unmeasurable defects like film platten non-flatness were found and corrected. Much of this development was lead by Brown and his collaborators. See [19] for more of the history and references.

Brown's initial 1958 bundle method [16,19] uses block matrix techniques to eliminate the structure parameters from the normal equations, leaving only the camera pose parameters. The resulting **reduced camera subsystem** is then solved by dense Gaussian elimination, and back-substitution gives the structure. For self-calibration, a second reduction from pose to calibration parameters can be added in the same way. Brown's method is probably what most vision researchers think of as 'bundle adjustment', following descriptions by Slama [100] and Hartley [58,59]. It is still a reasonable choice for small dense networks[29], but it rapidly becomes inefficient for the large sparse ones that arise in aerial cartography and large-scale site modelling.

For larger problems, more of the natural sparsity has to be exploited. In aerial cartography, the regular structure makes this relatively straightforward. The images are arranged in **blocks** — rectangular or irregular grids designed for uniform ground coverage, formed from parallel 1D **strips** of images with about 50–70% forward overlap giving adjacent stereo pairs or triplets, about 10–20% side overlap, and a few known ground control points sprinkled sparsely throughout the block. Features are shared only between neighbouring images, and images couple in the reduced camera subsystem only if they share common features. So if the images are arranged in strip or cross-strip ordering, the reduced camera system has a triply-banded block structure (the upper and lower bands representing, *e.g.*, right and left neighbours, and the central band forward and backward ones). Several efficient numerical schemes exist for such matrices. The first was Gyer & Brown's 1967 **recursive partitioning** method [57,19], which is closely related to Helmert's 1880 geodesic method [64]. (Generalizations of these have become one of the major families of modern sparse matrix methods [40,26,11]). The basic idea is to split the rectangle into two halves, recursively solving each half and gluing the two solutions together along their common boundary. Algebraically, the variables are reordered into left-half-only, right-half-only and boundary variables, with the latter (representing the only coupling between the two halves) eliminated last. The technique is extremely effective for aerial blocks and similar problems where small *separating sets* of variables can be found. Brown mentions adjusting a block of 162 photos on a machine with only 8k words of memory, and 1000 photo blocks were already feasible by mid-1967 [19]. For less regular networks such as site modelling ones it may not be feasible to choose

[27] **Close range** means essentially that the object has significant depth relative to the camera distance, *i.e.* that there is significant perspective distortion. For aerial images the scene is usually shallow compared to the viewing height, so focal length variations are very difficult to disentangle from depth variations.

[28] **Calibration** always denotes *internal* camera parameters ("interior orientation") in photogrammetric terminology. External calibration is called **pose** or **(exterior) orientation**.

[29] A photogrammetric network is **dense** if most of the 3D features are visible in most of the images, and **sparse** if most features appear in only a few images. This corresponds directly to the density or sparsity of the off-diagonal block (feature-camera coupling matrix) of the bundle Hessian.

an appropriate variable ordering beforehand, but efficient on-line ordering methods exist [40,26,11] (see §6.3).

Independent model methods: These approximate bundle adjustment by calculating a number of partial reconstructions independently and merging them by pairwise 3D alignment. Even when the individual models and alignments are separately optimal, the result is suboptimal because the the stresses produced by alignment are not propagated back into the individual models. (Doing so would amount to completing one full iteration of an optimal recursive decomposition style bundle method — see §8.2). Independent model methods were at one time the standard in aerial photogrammetry [95,2,100,73], where they were used to merge individual stereo pair reconstructions within aerial strips into a global reconstruction of the whole block. They are always less accurate than bundle methods, although in some cases the accuracy can be comparable.

First order & approximate bundle algorithms: Another recurrent theme is the use of approximations or iterative methods to avoid solving the full Newton update equations. Most of the plausible approximations have been rediscovered several times, especially variants of alternate steps of resection (finding the camera poses from known 3D points) and intersection (finding the 3D points from known camera poses), and the linearized version of this, the block Gauss-Seidel iteration. Brown's group had already experimented with Block Successive Over-Relaxation (BSOR — an accelerated variant of Gauss-Seidel) by 1964 [19], before they developed their recursive decomposition method. Both Gauss-Seidel and BSOR were also applied to the independent model problem around this time [95,2]. These methods are mainly of historical interest. For large sparse problems such as aerial blocks, they can not compete with efficiently organized second order methods. Because some of the inter-variable couplings are ignored, corrections propagate very slowly across the network (typically one step per iteration), and many iterations are required for convergence (see §7).

Quality control: In parallel with this algorithmic development, two important theoretical developments took place. Firstly, the Dutch geodesist W. Baarda led a long-running working group that formulated a theory of statistical reliability for least squares estimation [7,8]. This greatly clarified the conditions (essentially **redundancy**) needed to ensure that outliers could be detected from their residuals (**inner reliability**), and that any remaining undetected outliers had only a limited effect on the final results (**outer reliability**). A. Grün [49,50] and W. Förstner [30,33,34] adapted this theory to photogrammetry around 1980, and also gave some early correlation and covariance based model selection heuristics designed to control over-fitting problems caused by over-elaborate camera models in self calibration.

Datum / gauge freedom: Secondly, as problem size and sophistication increased, it became increasingly difficult to establish sufficiently accurate control points for large geodesic and photogrammetric networks. Traditionally, the network had been viewed as a means of 'densifying' a fixed control coordinate system — propagating control-system coordinates from a few known control points to many unknown ones. But this viewpoint is suboptimal when the network is intrinsically more accurate than the control, because most of the apparent uncertainty is simply due to the uncertain definition of the control coordinate system itself. In the early 1960's, Meissl studied this problem and developed the first **free network** approach, in which the reference coordinate system floated freely rather than being locked to any given control points [87,89]. More precisely, the coordinates are pinned to a sort of average structure defined by so-called **inner constraints**. Owing to the removal of control-related uncertainties, the nominal structure covariances become smaller and easier to interpret, and the numerical bundle iteration also converges more rapidly. Later, Baarda introduced another approach to this theory based on **S-transforms** — coordinate transforms between uncertain frames [6,21,22,25].

Least squares matching: All of the above developments originally used manually extracted image

points. Automated image processing was clearly desirable, but it only gradually became feasible owing to the sheer size and detail of photogrammetric images. Both feature based, *e.g.* [31,32], and direct (region based) [1,52,55,110] methods were studied, the latter especially for matching low-contrast natural terrain in cartographic applications. Both rely on some form of **least squares matching** (as image correlation is called in photogrammetry). Correlation based matching techniques remain the most accurate methods of extracting precise translations from images, both for high contrast photogrammetric targets and for low contrast natural terrain. Starting from around 1985, Grün and his co-workers combined region based least squares matching with various geometric constraints. **Multi-photo geometrically constrained matching** optimizes the match over multiple images simultaneously, subject to the inter-image matching geometry [52,55,9]. For each surface patch there is a single search over patch depth and possibly slant, which *simultaneously* moves it along epipolar lines in the other images. Initial versions assumed known camera matrices, but a full patch-based bundle method was later investigated [9]. Related methods in computer vision include [94,98,67]. **Globally enforced least squares matching** [53,97,76] further stabilizes the solution in low-signal regions by enforcing continuity constraints between adjacent patches. Patches are arranged in a grid and matched using local affine or projective deformations, with additional terms to penalize mismatching at patch boundaries. Related work in vision includes [104,102]. The inter-patch constraints give a sparsely-coupled structure to the least squares matching equations, which can again be handled efficiently by recursive decomposition.

B Matrix Factorization

This appendix covers some standard material on matrix factorization, including the technical details of factorization, factorization updating, and covariance calculation methods. See [44,11] for more details.

Terminology: Depending on the factorization, 'L' stands for lower triangular, 'U' or 'R' for upper triangular, 'D' or 'S' for diagonal, 'Q' or 'U','V' for orthogonal factors.

B.1 Triangular decompositions

Any matrix A has a family of block (lower triangular)*(diagonal)*(upper triangular) factorizations $\mathsf{A} = \mathsf{L}\,\mathsf{D}\,\mathsf{U}$:

$$\begin{array}{ccccc} \mathsf{A} & = & \mathsf{L} & \mathsf{D} & \mathsf{U} \\ \begin{pmatrix} \mathsf{A}_{11} & \mathsf{A}_{12} & \cdots & \mathsf{A}_{1n} \\ \mathsf{A}_{21} & \mathsf{A}_{22} & \cdots & \mathsf{A}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathsf{A}_{m1} & \mathsf{A}_{m2} & \cdots & \mathsf{A}_{mn} \end{pmatrix} & = & \begin{pmatrix} \mathsf{L}_{11} & & & \\ \mathsf{L}_{21} & \mathsf{L}_{22} & & \\ \vdots & \vdots & \ddots & \\ \vdots & \vdots & & \vdots \\ \mathsf{L}_{m1} & \mathsf{L}_{m2} & \cdots & \mathsf{L}_{mr} \end{pmatrix} & \begin{pmatrix} \mathsf{D}_1 & & & \\ & \mathsf{D}_2 & & \\ & & \ddots & \\ & & & \mathsf{D}_r \end{pmatrix} & \begin{pmatrix} \mathsf{U}_{11} & \mathsf{U}_{12} & \cdots & \cdots & \mathsf{U}_{1n} \\ & \mathsf{U}_{22} & \cdots & \cdots & \mathsf{U}_{2n} \\ & & \ddots & & \vdots \\ & & & \cdots & \mathsf{U}_{rn} \end{pmatrix} \end{array} \tag{51}$$

$$\left.\begin{array}{rcll} \mathsf{L}_{ii}\,\mathsf{D}_i\,\mathsf{U}_{ii} & = & \overline{\mathsf{A}}_{ii}\,, & i = j \\ \mathsf{L}_{ij} & \equiv & \overline{\mathsf{A}}_{ij}\,\mathsf{U}_{jj}^{-1}\,\mathsf{D}_j^{-1}, & i > j \\ \mathsf{U}_{ij} & \equiv & \mathsf{D}_i^{-1}\,\mathsf{L}_{ii}^{-1}\,\overline{\mathsf{A}}_{ij}\,, & i < j \end{array}\right\} \qquad \begin{array}{rcl} \overline{\mathsf{A}}_{ij} & \equiv & \mathsf{A}_{ij} - \sum_{k<\min(i,j)} \mathsf{L}_{ik}\,\mathsf{D}_k\,\mathsf{U}_{kj} \\ & = & \mathsf{A}_{ij} - \sum_{k<\min(i,j)} \overline{\mathsf{A}}_{ik}\,\overline{\mathsf{A}}_{kk}^{-1}\,\overline{\mathsf{A}}_{kj} \end{array} \tag{52}$$

Here, the diagonal blocks $\mathsf{D}_1 \ldots \mathsf{D}_{r-1}$ must be chosen to be square and invertible, and r is determined by the rank of A. The recursion (52) follows immediately from the product $\mathsf{A}_{ij} = (\mathsf{L}\,\mathsf{D}\,\mathsf{U})_{ij} = \sum_{k\le\min(i,j)} \mathsf{L}_{ik}\,\mathsf{D}_k\,\mathsf{U}_{kj}$. Given such a factorization, linear equations can be solved by forwards and backwards substitution as in (22–24).

The diagonal blocks of $\mathsf{L}, \mathsf{D}, \mathsf{U}$ can be chosen freely subject to $\mathsf{L}_{ii}\,\mathsf{D}_{ii}\,\mathsf{U}_{ii} = \overline{\mathsf{A}}_{ii}$, but once this is done the factorization is uniquely defined. Choosing $\mathsf{L}_{ii} = \mathsf{D}_{ii} = \mathsf{1}$ so that $\mathsf{U}_{ii} = \overline{\mathsf{A}}_{ii}$ gives the (block) **LU decomposition** $\mathsf{A} = \mathsf{L}\,\mathsf{U}$, the matrix representation of (block) Gaussian elimination. Choosing $\mathsf{L}_{ii} = \mathsf{U}_{ii} = \mathsf{1}$ so that $\mathsf{D}_i = \overline{\mathsf{A}}_{ii}$ gives the **LDU decomposition**. If A is symmetric, the LDU decomposition preserves the symmetry and becomes the **LDL$^\top$ decomposition** $\mathsf{A} = \mathsf{L}\,\mathsf{D}\,\mathsf{L}^\top$ where $\mathsf{U} = \mathsf{L}^\top$ and $\mathsf{D} = \mathsf{D}^\top$. If A is symmetric positive definite we can set $\mathsf{D} = \mathsf{1}$ to get the **Cholesky decomposition** $\mathsf{A} = \mathsf{L}\,\mathsf{L}^\top$, where $\mathsf{L}_{ii}\,\mathsf{L}_{ii}^\top = \overline{\mathsf{A}}_{ii}$ (recursively) defines the Cholesky factor L_{ii} of the positive definite matrix $\overline{\mathsf{A}}_{ii}$. (For a scalar, $\mathrm{Chol}(a) = \sqrt{a}$). If all of the blocks are chosen to be 1×1, we get the conventional scalar forms of these decompositions. These decompositions are obviously equivalent, but for speed and simplicity it is usual to use the most specific one that applies: LU for general matrices, LDL$^\top$ for symmetric ones, and Cholesky for symmetric positive definite ones. For symmetric matrices such as the bundle Hessian, LDL$^\top$ / Cholesky are 1.5–2 times faster than LDU / LU. We will use the general form (51) below as it is trivial to specialize to any of the others.

Loop ordering: From (52), the ij block of the decomposition depends only on the the upper left $(m-1) \times (m-1)$ submatrix and the first m elements of row i and column j of A, where $m = \min(i,j)$. This allows considerable freedom in the ordering of operations during decomposition, which can be exploited to enhance parallelism and improve memory cache locality.

Fill in: If A is sparse, its L and U factors tend to become ever denser as the decomposition progresses. Recursively expanding $\overline{\mathsf{A}}_{ik}$ and $\overline{\mathsf{A}}_{kj}$ in (52) gives contributions of the form $\pm\mathsf{A}_{ik}\,\overline{\mathsf{A}}_{kk}^{-1}\,\mathsf{A}_{kl}\cdots\mathsf{A}_{pq}\,\overline{\mathsf{A}}_{qq}^{-1}\,\mathsf{A}_{qj}$ for $k, l \ldots p, q < \min(i,j)$. So even if A_{ij} is zero, if there is any path of the form $i \to k \to l \to \ldots \to p \to q \to j$ via non-zero A_{kl} with $k, l \ldots p, q < \min(i,j)$, the ij block of the decomposition will generically **fill-in** (become non-zero). The amount of fill-in is strongly dependent on the ordering of the variables (*i.e.* of the rows and columns of A). Sparse factorization methods (§6.3) manipulate this ordering to minimize either fill-in or total operation counts.

Pivoting: For positive definite matrices, the above factorizations are very stable because the **pivots** $\overline{\mathsf{A}}_{ii}$ must themselves remain positive definite. More generally, the pivots may become ill-conditioned causing the decomposition to break down. To deal with this, it is usual to search the undecomposed part of the matrix for a large pivot at each step, and permute this into the leading position before proceeding. The stablest policy is **full pivoting** which searches the whole submatrix, but usually a less costly **partial pivoting** search over just the current column (**column pivoting**) or row (**row pivoting**) suffices. Pivoting ensures that L and/or U are relatively well-conditioned and postpones ill-conditioning in D for as long as possible, but it can not ultimately make D any better conditioned than A is. Column pivoting is usual for the LU decomposition, but if applied to a symmetric matrix it destroys the symmetry and hence doubles the workload. **Diagonal pivoting** preserves symmetry by searching for the largest remaining diagonal element and permuting both its row and its column to the front. This suffices for positive semidefinite matrices (*e.g.* gauge deficient Hessians). For general symmetric indefinite matrices (*e.g.* the augmented Hessians $\left(\begin{smallmatrix}\mathsf{H} & \mathsf{C}\\ \mathsf{C}^\top & \mathsf{0}\end{smallmatrix}\right)$ of constrained problems (12)), off-diagonal pivots can not be avoided[30], but there are fast, stable, symmetry-preserving pivoted LDL$^\top$ decompositions with block diagonal D having 1×1 and 2×2 blocks. Full pivoting is possible (**Bunch-Parlett decomposition**), but **Bunch-Kaufman decomposition** which searches the diagonal and only one or at most two columns usually suffices. This method is nearly as fast as pivoted Cholesky decomposition (to which it reduces for positive matrices), and as stable LU decomposition with partial pivoting. **Åsen's method** has similar speed and stability but produces a tridiagonal D.

[30] The archetypical failure is the unstable LDL$^\top$ decomposition of the well-conditioned symmetric indefinite matrix $\left(\begin{smallmatrix}\epsilon & 1\\ 1 & 0\end{smallmatrix}\right) = \left(\begin{smallmatrix}1 & 0\\ 1/\epsilon & 1\end{smallmatrix}\right)\left(\begin{smallmatrix}\epsilon & 0\\ 0 & -1/\epsilon\end{smallmatrix}\right)\left(\begin{smallmatrix}1 & 1/\epsilon\\ 0 & 1\end{smallmatrix}\right)$, for $\epsilon \to 0$. Fortunately, for small diagonal elements, permuting the dominant off-diagonal element next to the diagonal and leaving the resulting 2×2 block undecomposed in D suffices for stability.

The constrained Hessian $\left(\begin{smallmatrix} \mathsf{H} & \mathsf{C} \\ \mathsf{C}^\top & 0 \end{smallmatrix}\right)$ has further special properties owing to its zero block, but we will not consider these here — see [44, §4.4.6 Equilibrium Systems].

B.2 Orthogonal decompositions

For least squares problems, there is an alternative family of decompositions based on orthogonal reduction of the Jacobian $\mathsf{J} = \frac{\mathsf{dz}}{\mathsf{dx}}$. Given any rectangular matrix A, it can be decomposed as $\mathsf{A} = \mathsf{Q}\,\mathsf{R}$ where R is upper triangular and Q is orthogonal (*i.e.*, its columns are orthonormal unit vectors). This is called the **QR decomposition** of A. R is identical to the right Cholesky factor of $\mathsf{A}^\top\mathsf{A} = (\mathsf{R}^\top\mathsf{Q}^\top)(\mathsf{Q}\,\mathsf{R}) = \mathsf{R}^\top\mathsf{R}$. The solution of the linear least squares problem $\min_{\mathsf{x}} \|\mathsf{A}\,\mathsf{x} - \mathsf{b}\|^2$ is $\mathsf{x} = \mathsf{R}^{-1}\mathsf{Q}^\top\mathsf{b}$, and $\mathsf{R}^{-1}\mathsf{Q}^\top$ is the Moore-Penrose pseduo-inverse of A. The QR decomposition is calculated by finding a series of simple rotations that successively zero below diagonal elements of A to form R, and accumulating the rotations in Q, $\mathsf{Q}^\top\mathsf{A} = \mathsf{R}$. Various types of rotations can be used. **Givens rotations** are the fine-grained extreme: one-parameter 2×2 rotations that zero a single element of A and affect only two of its rows. **Householder reflections** are coarser-grained reflections in hyperplanes $\mathsf{1} - 2\frac{\mathsf{v}\,\mathsf{v}^\top}{\|\mathsf{v}\|^2}$, designed to zero an entire below-diagonal column of A and affecting all elements of A in or below the diagonal row of that column. Intermediate sizes of Householder reflections can also be used, the 2×2 case being computationally equivalent, and equal up to a sign, to the corresponding Givens rotation. This is useful for sparse QR decompositions, *e.g.* multifrontal methods (see §6.3 and [11]). The Householder method is the most common one for general use, owing to its speed and simplicity. Both the Givens and Householder methods calculate R explicitly, but Q is not calculated directly unless it is explicitly needed. Instead, it is stored in factorized form (as a series of 2×2 rotations or Householder vectors), and applied piecewise when needed. In particular, $\mathsf{Q}^\top\mathsf{b}$ is needed to solve the least squares system, but it can be calculated progressively as part of the decomposition process. As for Cholesky decomposition, QR decomposition is stable without pivoting so long as A has full column rank and is not too ill-conditioned. For degenerate A, Householder QR decomposition with column exchange pivoting can be used. See [11] for more information about QR decomposition.

Both QR decomposition of A and Cholesky decomposition of the normal matrix $\mathsf{A}^\top\mathsf{A}$ can be used to calculate the Cholesky / QR factor R and to solve least squares problems with design matrix / Jacobian A. The QR method runs about as fast as the normal / Cholesky one for square A, but becomes twice as slow for long thin A (*i.e.* many observations in relatively few parameters). However, the QR is numerically much stabler than the normal / Cholesky one in the following sense: if A has condition number (ratio of largest to smallest singular value) c and the machine precision is ϵ, the QR solution has relative error $\mathcal{O}(c\epsilon)$, whereas the normal matrix $\mathsf{A}^\top\mathsf{A}$ has condition number c^2 and its solution has relative error $\mathcal{O}\big(c^2\epsilon\big)$. This matters only if $c^2\epsilon$ approaches the relative accuracy to which the solution is required. For example, even in accurate bundle adjustments, we do not need relative accuracies greater than about $1 : 10^6$. As $\epsilon \sim 10^{-16}$ for double precision floating point, we can safely use the normal equation method for $c(\mathsf{J}) \lesssim 10^5$, whereas the QR method is safe up to $c(\mathsf{J}) \lesssim 10^{10}$, where J is the bundle Jacobian. In practice, the Gauss-Newton / normal equation approach is used in most bundle implementations.

Individual Householder reflections are also useful for projecting parametrizations of geometric entities orthogonal to some constraint vector. For example, for quaternions or homogeneous projective vectors $\mathbf{X}$, we often want to enforce spherical normalization $\|\mathbf{X}\|^2 = 1$. To first order, only displacements $\boldsymbol{\delta}\mathbf{X}$ orthogonal to $\mathbf{X}$ are allowed, $\mathbf{X}^\top\boldsymbol{\delta}\mathbf{X} = 0$. To parametrize the directions we can move in, we need a basis for the vectors orthogonal to $\mathbf{X}$. A Householder reflection Q based on $\mathbf{X}$

$$
\begin{array}{l}
\mathsf{L} \;=\; \mathbf{profile\_cholesky\_decomp}(\mathsf{A}) \\
\quad \textbf{for } i = 1 \textbf{ to } n \textbf{ do} \\
\quad\quad \textbf{for } j = \mathrm{first}(i) \textbf{ to } i \textbf{ do} \\
\quad\quad\quad a \;=\; \mathsf{A}_{ij} \;-\; \sum_{k=\max(\mathrm{first}(i),\mathrm{first}(j))}^{j-1} \mathsf{L}_{ik}\,\mathsf{L}_{jk} \\
\quad\quad\quad \mathsf{L}_{ij} \;=\; (j < i) \;?\; a \,/\, \mathsf{L}_{jj} \;:\; \sqrt{a}
\end{array}
$$

$$
\begin{array}{l}
\mathsf{x} \;=\; \mathbf{profile\_cholesky\_forward\_subs}(\mathsf{A}, \mathsf{b}) \\
\quad \textbf{for } i = \mathrm{first}(\mathsf{b}) \textbf{ to } n \textbf{ do} \\
\quad\quad \mathsf{x}_i \;=\; \left(\mathsf{b}_i \;-\; \sum_{k=\max(\mathrm{first}(i),\mathrm{first}(\mathsf{b}))}^{i-1} \mathsf{L}_{ik}\,\mathsf{x}_k \right) / \mathsf{L}ii
\end{array}
$$

$$
\begin{array}{l}
\mathsf{y} \;=\; \mathbf{profile\_cholesky\_back\_subs}(\mathsf{A}, \mathsf{x}) \\
\quad \mathsf{y} \;=\; \mathsf{x} \\
\quad \textbf{for } i = \mathrm{last}(\mathsf{b}) \textbf{ to } 1 \textbf{ step } -1 \textbf{ do} \\
\quad\quad \textbf{for } k = \max(\mathrm{first}(i), \mathrm{first}(\mathsf{y})) \textbf{ to } i \textbf{ do} \\
\quad\quad\quad \mathsf{y}_k \;=\; \mathsf{y}_k - \mathsf{y}_i\,\mathsf{L}_{ik} \\
\quad\quad \mathsf{y}_i \;=\; \mathsf{y}_i \,/\, \mathsf{L}_{ii}
\end{array}
$$

Figure 10: A complete implementation of profile Cholesky decomposition.

converts $\mathbf{X}$ to $(1\ 0 \ldots 0)^\top$ and hence the orthogonal directions to vectors of the form $(0\ *\ldots *)^\top$. So if U contains rows 2–n of Q, we can reduce Jacobians $\frac{\mathsf{d}}{\mathsf{d}\mathbf{X}}$ to the $n-1$ independent parameters $\boldsymbol{\delta}\mathsf{u}$ of the orthogonal subspace by post-multiplying by $\mathsf{U}^\top$, and once we have solved for $\boldsymbol{\delta}\mathsf{u}$, we can recover the orthogonal $\boldsymbol{\delta}\mathbf{X} \approx \mathsf{U}\,\boldsymbol{\delta}\mathsf{u}$ by premultiplying by U. Multiple constraints can be enforced by successive **Householder reductions** of this form. This corresponds exactly to the **LQ method** for solving constrained least squares problems [11].

B.3 Profile Cholesky Decomposition

One of the simplest sparse methods suitable for bundle problems is **profile Cholesky decomposition**. With natural (features then cameras) variable ordering, it is as efficient as any method for dense networks (*i.e.* most features visible in most images, giving dense camera-feature coupling blocks in the Hessian). With suitable variable ordering[31], it is also efficient for some types of sparse problems, particularly ones with chain-like connectivity.

Figure 10 shows the complete implementation of profile Cholesky, including decomposition $\mathsf{L}\,\mathsf{L}^\top = \mathsf{A}$, forward substitution $\mathsf{x} = \mathsf{L}^{-1}\mathsf{b}$, and back substitution $\mathsf{y} = \mathsf{L}^{-\top}\mathsf{x}$. $\mathrm{first}(\mathsf{b}), \mathrm{last}(\mathsf{b})$ are the indices of the first and last nonzero entries of b, and $\mathrm{first}(i)$ is the index of the first nonzero entry in row i of A and hence L. If desired, $\mathsf{L}, \mathsf{x}, \mathsf{y}$ can overwrite $\mathsf{A}, \mathsf{b}, \mathsf{x}$ during decomposition to save storage. As always with factorizations, the loops can be reordered in several ways. These have the same operation counts but different access patterns and hence memory cache localities, which on modern machines can lead to significant performance differences for large problems. Here we store and access A and L consistently by rows.

B.4 Matrix Inversion and Covariances

When solving linear equations, forward-backward substitutions (22, 24) are much faster than explicitly calculating and multiplying by A^{-1}, and numerically stabler too. Explicit inverses are only rarely needed, *e.g.* to evaluate the dispersion ("covariance") matrix H^{-1}. Covariance calculation is expensive for bundle adjustment: no matter how sparse H may be, H^{-1} is always dense. Given a triangular decomposition $\mathsf{A} = \mathsf{L}\,\mathsf{D}\,\mathsf{U}$, the most obvious way to calculate A^{-1} is via the product $\mathsf{A}^{-1} = \mathsf{U}^{-1}\,\mathsf{D}^{-1}\,\mathsf{L}^{-1}$,

[31] Snay's Banker's strategy (§6.3.3, [101,24]) seems to be one of the most effective ordering strategies.

where L^{-1} (which is lower triangular) is found using a recurrence based on either $\mathsf{L}^{-1}\mathsf{L} = \mathsf{1}$ or $\mathsf{L}\mathsf{L}^{-1} = \mathsf{1}$ as follows (and similarly but transposed for U):

$$(\mathsf{L}^{-1})_{ii} = (\mathsf{L}_{ii})^{-1}, \qquad \underset{i=1\ldots n\ ,\ j=i+1\ldots n}{(\mathsf{L}^{-1})_{ji} = -\mathsf{L}_{jj}^{-1}\left(\sum_{k=i}^{j-1}\mathsf{L}_{jk}\,(\mathsf{L}^{-1})_{ki}\right)} = \underset{i=n\ldots 1\ ,\ j=n\ldots i+1}{-\left(\sum_{k=i+1}^{j}(\mathsf{L}^{-1})_{jk}\,\mathsf{L}_{ki}\right)\mathsf{L}_{ii}^{-1}} \tag{53}$$

Alternatively [45, 11], the diagonal and the (zero) upper triangle of the linear system $\mathsf{U}\,\mathsf{A}^{-1} = \mathsf{D}^{-1}\,\mathsf{L}^{-1}$ can be combined with the (zero) lower triangle of $\mathsf{A}^{-1}\,\mathsf{L} = \mathsf{U}^{-1}\,\mathsf{D}^{-1}$ to give the direct recursion ($i = n\ldots 1$ and $j = n\ldots i+1$):

$$\begin{aligned}
(\mathsf{A}^{-1})_{ji} &= -\left(\sum_{k=i+1}^{n}(\mathsf{A}^{-1})_{jk}\,\mathsf{L}_{ki}\right)\mathsf{L}_{ii}^{-1}, \qquad (\mathsf{A}^{-1})_{ij} = -\mathsf{U}_{ii}^{-1}\left(\sum_{k=i+1}^{n}\mathsf{U}_{ik}\,(\mathsf{A}^{-1})_{kj}\right) \\
(\mathsf{A}^{-1})_{ii} &= \mathsf{U}_{ii}^{-1}\left(\mathsf{D}_i^{-1}\mathsf{L}_{ii}^{-1} - \sum_{k=i+1}^{n}\mathsf{U}_{ik}\,(\mathsf{A}^{-1})_{ki}\right) = \left(\mathsf{U}_{ii}^{-1}\,\mathsf{D}_i^{-1} - \sum_{k=i+1}^{n}(\mathsf{A}^{-1})_{ik}\,\mathsf{L}_{ki}\right)\mathsf{L}_{ii}^{-1}
\end{aligned} \tag{54}$$

In the symmetric case $(\mathsf{A}^{-1})_{ji} = (\mathsf{A}^{-1})_{ij}$ so we can avoid roughly half of the work. If only a few blocks of A^{-1} are required (*e.g.* the diagonal ones), this recursion has the property that the blocks of A^{-1} associated with the filled positions of L and U can be calculated without calculating any blocks associated with unfilled positions. More precisely, to calculate $(\mathsf{A}^{-1})_{ij}$ for which L_{ji} ($j > i$) or U_{ji} ($j < i$) is non-zero, we do not need any block $(\mathsf{A}^{-1})_{kl}$ for which $\mathsf{L}_{lk} = \mathsf{0}$ ($l > k$) or $\mathsf{U}_{lk} = \mathsf{0}$ ($l < k$) [32]. This is a significant saving if L, U are sparse, as in bundle problems. In particular, given the covariance of the reduced camera system, the 3D feature variances and feature-camera covariances can be calculated efficiently using (54) (or equivalently (17), where $\mathsf{A} \leftarrow \mathsf{H}_{ss}$ is the block diagonal feature Hessian and D_2 is the reduced camera one).

B.5 Factorization Updating

For on-line applications (§8.2), it is useful to be able to **update** the decomposition $\mathsf{A} = \mathsf{L}\,\mathsf{D}\,\mathsf{U}$ to account for a (usually low-rank) change $\mathsf{A} \to \overline{\mathsf{A}} \equiv \mathsf{A} \pm \mathsf{B}\,\mathsf{W}\,\mathsf{C}$. Let $\overline{\mathsf{B}} \equiv \mathsf{L}^{-1}\,\mathsf{B}$ and $\overline{\mathsf{C}} \equiv \mathsf{C}\,\mathsf{U}^{-1}$ so that $\mathsf{L}^{-1}\,\overline{\mathsf{A}}\,\mathsf{U}^{-1} = \mathsf{D} \pm \overline{\mathsf{B}}\,\mathsf{W}\,\overline{\mathsf{C}}$. This low-rank update of D can be LDU decomposed efficiently. Separating the first block of D from the others we have:

$$\begin{aligned}
\begin{pmatrix}\mathsf{D}_1 & \\ & \mathsf{D}_2\end{pmatrix} \pm \begin{pmatrix}\overline{\mathsf{B}}_1 \\ \overline{\mathsf{B}}_2\end{pmatrix}\mathsf{W}\begin{pmatrix}\overline{\mathsf{C}}_1 & \overline{\mathsf{C}}_2\end{pmatrix} &= \begin{pmatrix}\mathsf{1} & \\ \pm\overline{\mathsf{B}}_2\,\mathsf{W}\,\overline{\mathsf{C}}_1\,\overline{\mathsf{D}}_1^{-1} & \mathsf{1}\end{pmatrix}\begin{pmatrix}\overline{\mathsf{D}}_1 & \\ & \overline{\mathsf{D}}_2\end{pmatrix}\begin{pmatrix}\mathsf{1} & \pm\overline{\mathsf{D}}_1^{-1}\,\overline{\mathsf{B}}_1\,\mathsf{W}\,\overline{\mathsf{C}}_2 \\ & \mathsf{1}\end{pmatrix} \\
\overline{\mathsf{D}}_1 &\equiv \mathsf{D}_1 \pm \overline{\mathsf{B}}_1\,\mathsf{W}\,\overline{\mathsf{C}}_1 \qquad\qquad \overline{\mathsf{D}}_2 \equiv \mathsf{D}_2 \pm \overline{\mathsf{B}}_2\left(\mathsf{W} \mp \mathsf{W}\,\overline{\mathsf{C}}_1\,\overline{\mathsf{D}}_1^{-1}\,\overline{\mathsf{B}}_1\,\mathsf{W}\right)\overline{\mathsf{C}}_2
\end{aligned} \tag{55}$$

[32] This holds because of the way fill-in occurs in the LDU decomposition. Suppose that we want to find $(\mathsf{A}^{-1})_{ij}$, where $j > i$ and $\mathsf{L}_{ji} \neq \mathsf{0}$. For this we need $(\mathsf{A}^{-1})_{kj}$ for all non-zero U_{ik}, $k > i$. But for these $\overline{\mathsf{A}}_{jk} = \mathsf{L}_{ji}\,\mathsf{D}_i\,\mathsf{U}_{ik} + \ldots + \mathsf{A}_{jk} \neq \mathsf{0}$, so $(\mathsf{A}^{-1})_{kj}$ is associated with a filled position and will already have been evaluated.

$\overline{\mathsf{D}}_2$ is a low-rank update of D_2 with the same $\overline{\mathsf{C}}_2$ and $\overline{\mathsf{B}}_2$ but a different W. Evaluating this recursively and merging the resulting L and U factors into L and U gives the updated decomposition[33] $\overline{\mathsf{A}} = \overline{\mathsf{L}}\,\overline{\mathsf{D}}\,\overline{\mathsf{U}}$:

$$\begin{array}{l}
\mathsf{W}^{(1)} \leftarrow \pm\mathsf{W}\,; \quad \mathsf{B}^{(1)} \leftarrow \mathsf{B}\,; \quad \mathsf{C}^{(1)} \leftarrow \mathsf{C}\,; \\
\textbf{for } i = 1 \textbf{ to } n \textbf{ do} \\
\qquad \overline{\mathsf{B}}_i \leftarrow \mathsf{B}_i^{(i)}\,; \quad \overline{\mathsf{C}}_i \leftarrow \mathsf{C}_i^{(i)}\,; \quad \overline{\mathsf{D}}_i \leftarrow \mathsf{D}_i + \overline{\mathsf{B}}_i\,\mathsf{W}^{(i)}\,\overline{\mathsf{C}}_i\,; \\
\qquad \mathsf{W}^{(i+1)} \leftarrow \mathsf{W}^{(i)} - \mathsf{W}^{(i)}\,\overline{\mathsf{C}}_i\,\overline{\mathsf{D}}_i^{-1}\,\overline{\mathsf{B}}_i\,\mathsf{W}^{(i)} = \left((\mathsf{W}^{(i)})^{-1} + \overline{\mathsf{C}}_i\,\mathsf{D}_i^{-1}\overline{\mathsf{B}}_i\right)^{-1}\,; \\
\qquad \textbf{for } j = i+1 \textbf{ to } n \textbf{ do} \\
\qquad\qquad \mathsf{B}_j^{(i+1)} \leftarrow \mathsf{B}_j^{(i)} - \mathsf{L}_{ji}\,\overline{\mathsf{B}}_i\,; \quad \overline{\mathsf{L}}_{ji} \leftarrow \mathsf{L}_{ji} + \mathsf{B}_j^{(i+1)}\,\mathsf{W}^{(i+1)}\,\overline{\mathsf{C}}_i\,\mathsf{D}_i^{-1}\,; \\
\qquad\qquad \mathsf{C}_j^{(i+1)} \leftarrow \mathsf{C}_j^{(i)} - \overline{\mathsf{C}}_i\,\mathsf{U}_{ij}\,; \quad \overline{\mathsf{U}}_{ij} \leftarrow \mathsf{U}_{ij} + \mathsf{D}_i^{-1}\,\overline{\mathsf{B}}_i\,\mathsf{W}^{(i+1)}\,\mathsf{C}_j^{(i+1)}\,;
\end{array} \tag{56}$$

The W^{-1} form of the W update is numerically stabler for additions ('+' sign in $\mathsf{A}\pm\mathsf{B}\,\mathsf{W}\,\mathsf{C}$ with positive W), but is not usable unless $\mathsf{W}^{(i)}$ is invertible. In either case, the update takes time $\mathcal{O}\big((k^2+b^2)N^2\big)$ where A is $N\times N$, W is $k\times k$ and the D_i are $b\times b$. So other things being equal, k should be kept as small as possible (*e.g.* by splitting the update into independent rows using an initial factorization of W, and updating for each row in turn). The scalar Cholesky form of this method for a rank one update $\mathsf{A} \rightarrow \mathsf{A} + w\,\mathsf{b}\,\mathsf{b}^\top$ is:

$$\begin{array}{l}
w^{(1)} \leftarrow w\,; \quad \mathsf{b}^{(1)} \leftarrow \mathsf{b}\,; \\
\textbf{for } i = 1 \textbf{ to } n \textbf{ do} \\
\qquad \overline{\mathsf{b}}_i \leftarrow \mathsf{b}_i^{(i)}/\mathsf{L}_{ii}\,; \quad \overline{d}_i \leftarrow 1 + w^{(i)}\,\overline{\mathsf{b}}_i^2\,; \quad \overline{\mathsf{L}}_{ii} \leftarrow \mathsf{L}_{ii}\sqrt{\overline{d}_i}\,; \\
\qquad w^{(i+1)} \leftarrow w^{(i)}/\overline{d}_i\,; \\
\qquad \textbf{for } j = i+1 \textbf{ to } n \textbf{ do} \\
\qquad\qquad \mathsf{b}_j^{(i+1)} \leftarrow \mathsf{b}_j^{(i)} - \mathsf{L}_{ji}\,\overline{\mathsf{b}}_i\,; \quad \overline{\mathsf{L}}_{ji} \leftarrow \left(\mathsf{L}_{ji} + \mathsf{b}_j^{(i+1)}\,w^{(i+1)}\,\overline{\mathsf{b}}_i\right)\sqrt{\overline{d}_i}\,;
\end{array} \tag{57}$$

This takes $\mathcal{O}\big(n^2\big)$ operations. The same recursion rule (and several equivalent forms) can be derived by reducing $(\mathsf{L}\ \ \mathsf{b})^\top$ to an upper triangular matrix using Givens rotations or Householder transformations [43, 11].

C Software

C.1 Software Organization

For a general purpose bundle adjustment code, an extensible object-based organization is natural. The measurement network can be modelled as a network of objects, representing *measurements and their error models* and the different types of *3D features* and *camera models* that they depend on. It is obviously useful to allow the measurement, feature and camera types to be open-ended. Measurements may be 2D or 3D, implicit or explicit, and many different robust error models are possible. Features may range from points through curves and homographies to entire 3D object models. Many types of camera and lens distortion models exist. If the scene is dynamic or articulated, additional nodes representing 3D transformations (kinematic chains or relative motions) may also be needed.

The main purpose of the network structure is to predict observations and their Jacobians w.r.t. the free parameters, and then to integrate the resulting first order parameter updates back into the internal

[33] Here, $\mathsf{B}_j^{(i)} = \mathsf{B}_j - \sum_{k=1}^{i-1}\mathsf{L}_{jk}\,\overline{\mathsf{B}}_k = \sum_{k=i}^{j}\mathsf{L}_{jk}\,\overline{\mathsf{B}}_k$ and $\mathsf{C}_j^{(i)} = \mathsf{C}_j - \sum_{k=1}^{i-1}\overline{\mathsf{C}}_k\,\mathsf{L}_{kj} = \sum_{k=i}^{j}\overline{\mathsf{C}}_k\,\mathsf{U}_{kj}$ accumulate $\mathsf{L}^{-1}\,\mathsf{B}$ and $\mathsf{C}\,\mathsf{U}^{-1}$. For the L, U updates one can also use $\mathsf{W}^{(i+1)}\,\overline{\mathsf{C}}_i\,\mathsf{D}_i^{-1} = \mathsf{W}^{(i)}\,\overline{\mathsf{C}}_i\,\overline{\mathsf{D}}_i^{-1}$ and $\mathsf{D}_i^{-1}\,\overline{\mathsf{B}}_i\,\mathsf{W}^{(i+1)} = \overline{\mathsf{D}}_i^{-1}\,\overline{\mathsf{B}}_i\,\mathsf{W}^{(i)}$.

3D feature and camera state representations. Prediction is essentially a matter of systematically propagating values through the network, with heavy use of the chain rule for derivative propagation. The network representation must interface with a numerical linear algebra one that supports appropriate methods for forming and solving the sparse, damped Gauss-Newton (or other) step prediction equations. A fixed-order sparse factorization may suffice for simple networks, while automatic variable ordering is needed for more complicated networks and iterative solution methods for large ones.

Several extensible bundle codes exist, but as far as we are aware, none of them are currently available as freeware. Our own implementations include:

- CARMEN [59] is a program for camera modelling and scene reconstruction using iterative nonlinear least squares. It has a modular design that allows many different feature, measurement and camera types to be incorporated (including some quite exotic ones [56, 63]). It uses sparse matrix techniques similar to Brown's reduced camera system method [19] to make the bundle adjustment iteration efficient.

- HORATIO (http://www.ee.surrey.ac.uk/Personal/P.McLauchlan/horatio/html, [85, 86, 83, 84]) is a C library supporting the development of efficient computer vision applications. It contains support for image processing, linear algebra and visualization, and will soon be made publicly available. The bundle adjustment methods in Horatio, which are based on the Variable State Dimension Filter (VSDF) [83, 84], are being commercialized. These algorithms support sparse block matrix operations, arbitrary gauge constraints, global and local parametrizations, multiple feature types and camera models, as well as batch and sequential operation.

- VXL: This modular C++ vision environment is a new, lightweight version of the TargetJr/IUE environment, which is being developed mainly by the Universities of Oxford and Leuven, and General Electric CRD. The initial public release on http://www.robots.ox.ac.uk/∼vxl will include an OpenGL user interface and classes for multiple view geometry and numerics (the latter being mainly C++ wrappers to well established routines from Netlib — see below). A bundle adjustment code exists for it but is not currently planned for release [28, 62].

C.2 Software Resources

A great deal of useful numerical linear algebra and optimization software is available on the Internet, although more commonly in FORTRAN than in C/C++. The main repository is NETLIB at http://www.netlib.org/. Other useful sites include: the 'Guide to Available Mathematical Software' GAMS at http://gams.nist.gov; the NEOS guide http://www-fp.mcs.anl.gov/otc/Guide/, which is based in part on Moré & Wright's guide book [90]; and the Object Oriented Numerics page http://oonumerics.org. For large-scale dense linear algebra, LAPACK (http://www.netlib.org/lapack, [3]) is the best package available. However it is optimized for relatively large problems (matrices of size 100 or more), so if you are solving many small ones (size less than 20 or so) it may be faster to use the older LINPACK and EISPACK routines. These libraries all use the BLAS (Basic Linear Algebra Subroutines) interface for low level matrix manipulations, optimized versions of which are available from most processor vendors. They are all FORTRAN based, but C/C++ versions and interfaces exist (CLAPACK, http://www.netlib.org/clapack; LAPACK++, http://math.nist.gov/lapack++). For sparse matrices there is a bewildering array of packages. One good one is Boeing's SPOOLES (http://www.netlib.org/linalg/spooles/spooles.2.2.html) which implements sparse Bunch-Kaufman decomposition in C with several ordering methods. For iterative linear system solvers implementation is seldom difficult, but there are again many methods and implementations. The 'Templates'

book [10] contains potted code. For nonlinear optimization there are various older codes such as MINPACK, and more recent codes designed mainly for very large problems such as MINPACK-2 (ftp://info.mcs.anl.gov/pub/MINPACK-2) and LANCELOT (http://www.cse.clrc.ac.uk/Activity/LANCELOT). (Both of these latter codes have good reputations for other large scale problems, but as far as we are aware they have not yet been tested on bundle adjustment). All of the above packages are freely available. Commercial vendors such as NAG (ttp://www.nag.co.uk) and IMSL (www.imsl.com) have their own optimization codes.

Glossary

This glossary includes a few common terms from vision, photogrammetry, numerical optimization and statistics, with their translations.

Additional parameters: Parameters added to the basic perspective model to represent lens distortion and similar small image deformations.

α-distribution: A family of wide tailed probability distributions, including the **Cauchy distribution** ($\alpha = 1$) and the Gaussian ($\alpha = 2$).

Alternation: A family of simplistic and largely outdated strategies for nonlinear optimization (and also iterative solution of linear equations). Cycles through variables or groups of variables, optimizing over each in turn while holding all the others fixed. Nonlinear alternation methods usually relinearize the equations after each group, while **Gauss-Seidel** methods propagate first order corrections forwards and relinearize only at the end of the cycle (the results are the same to first order). **Successive over-relaxation** adds momentum terms to speed convergence. See **separable problem**. Alternation of **resection** and **intersection** is a naïve and often-rediscovered bundle method.

Asymptotic limit: In statistics, the limit as the number of independent measurements is increased to infinity, or as the second order moments dominate all higher order ones so that the posterior distribution becomes approximately Gaussian.

Asymptotic convergence: In optimization, the limit of small deviations from the solution, *i.e*. as the solution is reached. **Second order** or **quadratically convergent** methods such as **Newton's method** *square* the norm of the residual at each step, while **first order** or **linearly convergent** methods such as **steepest descent** and **alternation** only reduce the error by a constant factor at each step.

Banker's strategy: See **fill in**, §6.3.3.

Block: A (possibly irregular) grid of overlapping photos in aerial cartography.

Bunch-Kauffman: A numerically efficient factorization method for symmetric indefinite matrices, $\mathsf{A} = \mathsf{L}\,\mathsf{D}\,\mathsf{L}^\top$ where L is lower triangular and D is block diagonal with 1×1 and 2×2 blocks (§6.2, B.1).

Bundle adjustment: Any refinement method for visual reconstructions that aims to produce jointly optimal structure and camera estimates.

Calibration: In photogrammetry, this always means *internal* calibration of the cameras. See **inner orientation**.

Central limit theorem: States that maximum likelihood and similar estimators asymptotically have Gaussian distributions. The basis of most of our perturbation expansions.

Cholesky decomposition: A numerically efficient factorization method for symmetric positive definite matrices, $\mathsf{A} = \mathsf{L}\,\mathsf{L}^\top$ where L is lower triangular.

Close Range: Any photogrammetric problem where the scene is relatively close to the camera, so that it has significant depth compared to the camera distance. Terrestrial photogrammetry as opposed to **aerial cartography**.

Conjugate gradient: A cleverly accelerated first order iteration for solving positive definite linear systems or minimizing a nonlinear cost function. See **Krylov subspace**.

Cost function: The function quantifying the total residual error that is minimized in an adjustment computation.

Cramér-Rao bound: See **Fisher information**.

Criterion matrix: In network design, an ideal or desired form for a covariance matrix.

Damped Newton method: **Newton's method** with a stabilizing step control policy added. See **Levenberg-Marquardt**.

Data snooping: Elimination of outliers based on examination of their residual errors.

Datum: A reference coordinate system, against which other coordinates and uncertainties are measured. Our principle example of a **gauge**.

Dense: A matrix or system of equations with so few known-zero elements that it may as well be treated as having none. The opposite of **sparse**. For photogrammetric networks, **dense** means that the off-diagonal structure-camera block of the Hessian is dense, *i.e.* most features are seen in most images.

Descent direction: In optimization, any search direction with a downhill component, *i.e.* that locally reduces the cost.

Design: The process of defining a measurement network (placement of cameras, number of images, *etc.*) to satisfy given accuracy and quality criteria.

Design matrix: The observation-state Jacobian $\mathsf{J} = \frac{\mathsf{dz}}{\mathsf{dx}}$.

Direct method: Dense correspondence or reconstruction methods based directly on cross-correlating photometric intensities or related descriptor images, without extracting geometric features. See **least squares matching**, **feature based method**.

Dispersion matrix: The inverse of the cost function **Hessian**, a measure of distribution spread. In the **asymptotic limit**, the covariance is given by the dispersion.

Downdating: On-the-fly removal of observations, without recalculating everything from scratch. The inverse of **updating**.

Elimination graph: A graph derived from the **network graph**, describing the progress of **fill in** during sparse matrix factorization.

Empirical distribution: A set of samples from some probability distribution, viewed as an sum-of-delta-function approximation to the distribution itself. The **law of large numbers** asserts that the approximation asymptotically converges to the true distribution in probability.

Fill-in: The tendency of zero positions to become nonzero as sparse matrix factorization progresses. **Variable ordering strategies** seek to minimize fill-in by permuting the variables before factorization. Methods include **minimum degree**, **reverse Cuthill-McKee**, **Banker's strategies**, and **nested dissection**. See §6.3.

Fisher information: In parameter estimation, the mean curvature of the posterior log likelihood function, regarded as a measure of the certainty of an estimate. The **Cramér-Rao bound** says that any unbiased estimator has covariance $\geq$ the inverse of the Fisher information.

Free gauge / free network: A **gauge** or **datum** that is defined internally to the measurement network, rather than being based on predefined reference features like a **fixed gauge**.

Feature based: Sparse correspondence / reconstruction methods based on geometric image features (points, lines, homographies...) rather than direct photometry. See **direct method**.

Filtering: In sequential problems such as time series, the estimation of a current value using all of the previous measurements. **Smoothing** can correct this afterwards, by integrating also the information from future measurements.

First order method / convergence: See **asymptotic convergence**.

Gauge: An internal or external reference coordinate system defined for the current state and (at least) small variations of it, against which other quantities *and their uncertainties* can be measured. The 3D coordinate gauge is also called the **datum**. A **gauge constraint** is any constraint fixing a specific gauge, *e.g.* for the current state and arbitrary (small) displacements of it. The fact that the gauge can be chosen arbitrarily without changing the underlying structure is called **gauge freedom** or **gauge invariance**. The rank-deficiency that this transformation-invariance of the cost function induces on the Hessian is called **gauge deficiency**. Displacements that violate the gauge constraints can be corrected by applying an **S-transform**, whose linear form is a **gauge projection matrix** $\mathsf{P}_{\mathcal{G}}$.

Gauss-Markov theorem: This says that for a linear system, least squares weighted by the true measurement covariances gives the Best (minimum variance) Linear Unbiased Estimator or BLUE.

Gauss-Newton method: A Newton-like method for nonlinear least squares problems, in which the Hessian is approximated by the Gauss-Newton one $\mathsf{H} \approx \mathsf{J}^\top \mathsf{W} \mathsf{J}$ where J is the **design matrix** and W is a weight matrix. The **normal equations** are the resulting Gauss-Newton step prediction equations $(\mathsf{J}^\top \mathsf{W} \mathsf{J})\, \delta\mathsf{x} = -(\mathsf{J} \mathsf{W} \triangle\mathsf{z})$.

Gauss-Seidel method: See **alternation**.

Givens rotation: A 2×2 rotation used to as part of orthogonal reduction of a matrix, *e.g*. QR, SVD. See **Householder reflection**.

Gradient: The derivative of the cost function w.r.t. the parameters $\mathsf{g} = \frac{\mathsf{df}}{\mathsf{dx}}$.

Gradient descent: See **steepest descent**.

Hessian: The second derivative matrix of the cost function $\mathsf{H} = \frac{\mathsf{d}^2\mathsf{f}}{\mathsf{dx}^2}$. Symmetric and positive (semi-)definite at a cost minimum. Measures how 'stiff' the state estimate is against perturbations. Its inverse is the **dispersion matrix**.

Householder reflection: A matrix representing reflection in a hyperplane, used as a tool for orthogonal reduction of a matrix, *e.g*. QR, SVD. See **Givens rotation**.

Independent model method: A suboptimal approximation to bundle adjustment developed for aerial cartography. Small local 3D models are reconstructed, each from a few images, and then glued together via **tie features** at their common boundaries, without a subsequent adjustment to relax the internal stresses so caused.

Inner: Internal or intrinsic.

Inner constraints: **Gauge constraints** linking the gauge to some weighted average of the reconstructed features and cameras (rather than to an externally supplied reference system).

Inner orientation: Internal camera calibration, including lens distortion, *etc*.

Inner reliability: The ability to either resist outliers, or detect and reject them based on their residual errors.

Intersection: (of optical rays). Solving for 3D feature positions given the corresponding image features and known 3D camera poses and calibrations. See **resection**, **alternation**.

Jacobian: See **design matrix**.

Krylov subspace: The linear subspace spanned by the iterated products $\{\mathsf{A}^k\, \mathsf{b} | k = 0 \ldots n\}$ of some square matrix A with some vector b, used as a tool for generating linear algebra and nonlinear optimization iterations. **Conjugate gradient** is the most famous Krylov method.

Kullback-Leibler divergence: See **relative entropy**.

Least squares matching: Image matching based on photometric intensities. See **direct method**.

Levenberg-Marquardt: A common damping (step control) method for nonlinear least squares problems, consisting of adding a multiple $\lambda\mathsf{D}$ of some positive definite weight matrix D to the Gauss-Newton Hessian before solving for the step. Levenberg-Marquardt uses a simple rescaling based heuristic for setting λ, while **trust region** methods use a more sophisticated step-length based one. Such methods are called **damped Newton** methods in general optimization.

Local model: In optimization, a local approximation to the function being optimized, which is easy enough to optimize that an iterative optimizer for the original function can be based on it. The second order Taylor series model gives **Newton's method**.

Local parametrization: A parametrization of a nonlinear space based on offsets from some current point. Used during an optimization step to give better local numerical conditioning than a more global parametrization would.

LU decomposition: The usual matrix factorization form of Gaussian elimination.

Minimum degree ordering: One of the most widely used automatic variable ordering methods for sparse matrix factorization.

Minimum detectable gross error: The smallest outlier that can be detected on average by an outlier detection

method.

Nested dissection: A top-down divide-and-conquer **variable ordering method** for sparse matrix factorization. Recursively splits the problem into disconnected halves, dealing with the **separating set** of connecting variables last. Particularly suitable for surface coverage problems. Also called **recursive partitioning**.

Nested models: Pairs of models, of which one is a specialization of the other obtained by freezing certain parameters(s) at prespecified values.

Network: The interconnection structure of the 3D features, the cameras, and the measurements that are made of them (image points, *etc*.). Usually encoded as a graph structure.

Newton method: The basic iterative second order optimization method. The **Newton step** state update $\delta \mathsf{x} = -\mathsf{H}^{-1}\mathsf{g}$ minimizes a local quadratic Taylor approximation to the cost function at each iteration.

Normal equations: See **Gauss-Newton method**.

Nuisance parameter: Any parameter that had to be estimated as part of a nonlinear parameter estimation problem, but whose value was not really wanted.

Outer: External. See **inner**.

Outer orientation: Camera pose (position and angular orientation).

Outer reliability: The influence of unremoved outliers on the final parameter estimates, *i.e*. the extent to which they are reliable even though some (presumably small or lowly-weighted) outliers may remain undetected.

Outlier: An observation that deviates significantly from its predicted position. More generally, any observation that does not fit some preconceived notion of how the observations should be distributed, and which must therefore be removed to avoid disturbing the parameter estimates. See **total distribution**.

Pivoting: Row and/or column exchanges designed to promote stability during matrix factorization.

Point estimator: Any estimator that returns a single "best" parameter estimate, *e.g*. maximum likelihood, maximum a posteriori.

Pose: 3D position and orientation (angle), *e.g*. of a camera.

Preconditioner: A linear change of variables designed to improve the accuracy or convergence rate of a numerical method, *e.g*. a first order optimization iteration. **Variable scaling** is the diagonal part of preconditioning.

Primary structure: The main decomposition of the bundle adjustment variables into structure and camera ones.

Profile matrix: A storage scheme for sparse matrices in which all elements between the first and the last nonzero one in each row are stored, even if they are zero. Its simplicity makes it efficient even if there are quite a few zeros.

Quality control: The monitoring of an estimation process to ensure that accuracy requirements were met, that outliers were removed or down-weighted, and that appropriate models were used, *e.g*. for **additional parameters**.

Radial distribution: An observation error distribution which retains the Gaussian dependence on a squared residual error $r = \mathsf{x}^\top \mathsf{W}\mathsf{x}$, but which replaces the exponential $e^{-r/2}$ form with a more robust long-tailed one.

Recursive: Used of filtering-based reconstruction methods that handle sequences of images or measurements by successive updating steps.

Recursive partitioning: See **nested dissection**.

Reduced problem: Any problem where some of the variables have already been eliminated by partial factorization, leaving only the others. The **reduced camera system** (20) is the result of reducing the bundle problem to only the camera variables. (§6.1, 8.2, 4.4).

Redundancy: The extent to which any one observation has only a small influence on the results, so that it could be incorrect or missing without causing problems. Redundant consenses are the basis of reliability. **Redundancy numbers** r are a heuristic measure of the amount of redundancy in an estimate.

Relative entropy: An information-theoretic measure of how badly a model probability density p_1 fits an actual

one p_0: the mean (w.r.t. p_0) log likelihood contrast of p_0 to p_1, $\langle \log(p_0/p_1) \rangle_{p_0}$.

Resection: (of optical rays). Solving for 3D camera poses and possibly calibrations, given image features and the corresponding 3D feature positions. See **intersection**.

Resection-intersection: See **alternation**.

Residual: The error $\triangle \mathsf{z}$ in a predicted observation, or its cost function value.

S-transformation: A transformation between two **gauges**, implemented locally by a **gauge projection matrix** $\mathsf{P}_{\mathcal{G}}$.

Scaling: See **preconditioner**.

Schur complement: Of A in $\left(\begin{smallmatrix} \mathsf{A} & \mathsf{B} \\ \mathsf{C} & \mathsf{D} \end{smallmatrix}\right)$ is $\mathsf{D} - \mathsf{C}\,\mathsf{A}^{-1}\mathsf{B}$. See §6.1.

Second order method / convergence: See **asymptotic convergence**.

Secondary structure: Internal structure or sparsity of the off-diagonal feature-camera coupling block of the bundle Hessian. See **primary structure**.

Self calibration: Recovery of camera (internal) calibration during bundle adjustment.

Sensitivity number: A heuristic number s measuring the sensitivity of an estimate to a given observation.

Separable problem: Any optimization problem in which the variables can be separated into two or more subsets, for which optimization over each subset given all of the others is significantly easier than simultaneous optimization over all variables. Bundle adjustment is separable into 3D structure and cameras. **Alternation** (successive optimization over each subset) is a naïve approach to separable problems.

Separating set: See **nested dissection**.

Sequential Quadratic Programming (SQP): An iteration for constrained optimization problems, the constrained analogue of **Newton's method**. At each step optimizes a **local model** based on a quadratic model function with linearized constraints.

Sparse: "Any matrix with enough zeros that it pays to take advantage of them" (Wilkinson).

State: The bundle adjustment parameter vector, including all scene and camera parameters to be estimated.

Steepest descent: Naïve optimization method which consists of descent directly (in some given coordinate system) down the gradient of the cost function.

Sticky prior: A robust prior with a central peak but wide tails, designed to let the estimate 'unstick' from the peak if there is strong evidence against it.

Subset selection: The selection of a stable subset of 'live' variables on-line during pivoted factorization. *E.g.*, used as a method for selecting variables to constrain with trivial gauge constraints (§9.5).

Successive Over-Relaxation (SOR): See **alternation**.

Sum of Squared Errors (SSE): The nonlinear least squares cost function. The (possibly weighted) sum of squares of all of the residual feature projection errors.

Total distribution: The error distribution expected for *all* observations of a given type, including both inliers and outliers. *I.e.* the distribution that should be used in maximum likelihood estimation.

Trivial gauge: A **gauge** that fixes a small set of predefined reference features or cameras at given coordinates, irrespective of the values of the other features.

Trust region: See **Levenberg-Marquardt**.

Updating: Incorporation of additional observations without recalculating everything from scratch.

Variable ordering strategy: See **fill-in**.

Weight matrix: An information (inverse covariance) like matrix matrix W, designed to put the correct relative statistical weights on a set of measurements.

Woodbury formula: The matrix inverse updating formula (18).

References

[1] F. Ackermann. Digital image correlation: Performance and potential applications in photogrammetry. *Photogrammetric Record*, 11(64):429–439, 1984.

[2] F. Amer. Digital block adjustment. *Photogrammetric Record*, 4:34–47, 1962.

[3] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide, Third Edition*. SIAM Press, Philadelphia, 1999. LAPACK home page: http://www.netlib.org/lapack.

[4] C. Ashcraft and J. W.-H. Liu. Robust ordering of sparse matrices using multisection. *SIAM J. Matrix Anal. Appl.*, 19:816–832, 1998.

[5] K. B. Atkinson, editor. *Close Range Photogrammetry and Machine Vision*. Whittles Publishing, Roseleigh House, Latheronwheel, Caithness, Scotland, 1996.

[6] W. Baarda. S-transformations and criterion matrices. *Netherlands Geodetic Commission, Publications on Geodesy*, New Series, Vol.5, No.1 (168 pages), 1967.

[7] W. Baarda. Statistical concepts in geodesy. *Netherlands Geodetic Commission Publications on Geodesy*, New Series, Vol.2, No.4 (74 pages), 1967.

[8] W. Baarda. A testing procedure for use in geodetic networks. *Netherlands Geodetic Commission Publications on Geodesy*, New Series, Vol.2, No.5 (97 pages), 1968.

[9] E. P. Baltsavias. *Multiphoto Geometrically Constrained Matching*. PhD thesis, ETH-Zurich, 1992.

[10] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM Press, Philadelphia, 1993.

[11] Åke Björck. *Numerical Methods for Least Squares Problems*. SIAM Press, Philadelphia, PA, 1996.

[12] J. A. R. Blais. Linear least squares computations using Givens transformations. *Canadian Surveyor*, 37(4):225–233, 1983.

[13] P. T. Boggs, R. H. Byrd, J. E. Rodgers, and R. B. Schnabel. Users reference guide for ODRPACK 2.01: Software for weighted orthogonal distance regression. Technical Report NISTIR 92-4834, NIST, Gaithersburg, MD, June 1992.

[14] P. T. Boggs, R. H. Byrd, and R. B. Schnabel. A stable and efficient algorithm for nonlinear orthogonal regression. *SIAM J. Sci. Statist. Comput*, 8:1052–1078, 1987.

[15] R. J. Boscovich. De litteraria expeditione per pontificiam ditionem, et synopsis amplioris operis, ac habentur plura ejus ex exemplaria etiam sensorum impressa. *Bononiensi Scientarum et Artum Instituto Atque Academia Commentarii*, IV:353–396, 1757.

[16] D. C. Brown. A solution to the general problem of multiple station analytical stereotriangulation. Technical Report RCA-MTP Data Reduction Technical Report No. 43 (or AFMTC TR 58-8), Patrick Airforce Base, Florida, 1958.

[17] D. C. Brown. Close range camera calibration. *Photogrammetric Engineering*, XXXVII(8), August 1971.

[18] D. C. Brown. Calibration of close range cameras. *Int. Archives Photogrammetry*, 19(5), 1972. Unbound paper (26 pages).

[19] D. C. Brown. The bundle adjustment — progress and prospects. *Int. Archives Photogrammetry*, 21(3), 1976. Paper number 3–03 (33 pages).

[20] Q. Chen and G. Medioni. Efficient iterative solutions to m-view projective reconstruction problem. In *Int. Conf. Computer Vision & Pattern Recognition*, pages II:55–61. IEEE Press, 1999.

[21] M. A. R. Cooper and P. A. Cross. Statistical concepts and their application in photogrammetry and surveying. *Photogrammetric Record*, 12(71):637–663, 1988.

[22] M. A. R. Cooper and P. A. Cross. Statistical concepts and their application in photogrammetry and surveying (continued). *Photogrammetric Record*, 13(77):645–678, 1991.

[23] D. R. Cox and D. V. Hinkley. *Theoretical Statistics*. Chapman & Hall, 1974.

[24] P. J. de Jonge. A comparative study of algorithms for reducing the fill-in during Cholesky factorization. *Bulletin Géodésique*, 66:296–305, 1992.

[25] A. Dermanis. The photogrammetric inner constraints. *J. Photogrammetry & Remote Sensing*, 49(1):25–39, 1994.

[26] I. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices*. Oxford University Press, 1986.

[27] O. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In G. Sandini, editor, *European Conf. Computer Vision*, Santa Margherita Ligure, Italy, May 1992. Springer-Verlag.

[28] A. W. Fitzgibbon and A. Zisserman. Automatic camera recovery for closed or open image sequences. In *European Conf. Computer Vision*, pages 311–326, Freiburg, 1998.

[29] R. Fletcher. *Practical Methods of Optimization*. John Wiley, 1987.

[30] W. Förstner. Evaluation of block adjustment results. *Int. Arch. Photogrammetry*, 23-III, 1980.

[31] W. Förstner. On the geometric precision of digital correlation. *Int. Arch. Photogrammetry & Remote Sensing*, 24(3):176–189, 1982.

[32] W. Förstner. A feature-based correspondence algorithm for image matching. *Int. Arch. Photogrammetry & Remote Sensing*, 26 (3/3):150–166, 1984.

[33] W. Förstner. The reliability of block triangulation. *Photogrammetric Engineering & Remote Sensing*, 51(8):1137–1149, 1985.

[34] W. Förstner. Reliability analysis of parameter estimation in linear models with applications to mensuration problems in computer vision. *Computer Vision, Graphics & Image Processing*, 40:273–310, 1987.

[35] D. A. Forsyth, S. Ioffe, and J. Haddon. Bayesian structure from motion. In *Int. Conf. Computer Vision*, pages 660–665, Corfu, 1999.

[36] C. F. Gauss. *Werke*. Königlichen Gesellschaft der Wissenschaften zu Göttingen, 1870–1928.

[37] C. F. Gauss. *Theoria Combinationis Observationum Erroribus Minimis Obnoxiae (Theory of the Combination of Observations Least Subject to Errors)*. SIAM Press, Philadelphia, PA, 1995. Originally published in *Commentatines Societas Regiae Scientarium Gottingensis Recentiores* 5, 1823 (*Pars prior, Pars posterior*), 6, 1828 (*Supplementum*). Translation and commentary by G. W. Stewart.

[38] J. A. George. Nested dissection of a regular finite element mesh. *SIAM J. Numer. Anal.*, 10:345–363, 1973.

[39] J. A. George, M. T. Heath, and E. G. Ng. A comparison of some methods for solving sparse linear least squares problems. *SIAM J. Sci. Statist. Comput.*, 4:177–187, 1983.

[40] J. A. George and J. W.-H. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, 1981.

[41] J. A. George and J. W.-H. Liu. Householder reflections versus Givens rotations in sparse orthogonal decomposition. *Lin. Alg. Appl.*, 88/89:223–238, 1987.

[42] P. Gill, W. Murray, and M. Wright. *Practical Optimization*. Academic Press, 1981.

[43] P. E. Gill, G. H. Golub, W. Murray, and M. Saunders. Methods for modifying matrix factorizations. *Math. Comp.*, 28:505–535, 1974.

[44] G. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3^{rd} edition, 1996.

[45] G. Golub and R. Plemmons. Large-scale geodetic least squares adjustment by dissection and orthogonal decomposition. *Linear Algebra Appl.*, 34:3–28, 1980.

[46] S. Granshaw. Bundle adjustment methods in engineering photogrammetry. *Photogrammetric Record*, 10(56):181–207, 1980.

[47] A. Greenbaum. Behaviour of slightly perturbed Lanczos and conjugate-gradient recurrences. *Linear Algebra Appl.*, 113:7–63, 1989.

[48] A. Greenbaum. *Iterative Methods for Solving Linear Systems*. SIAM Press, Philadelphia, 1997.

[49] A. Grün. Accuracy, reliability and statistics in close range photogrammetry. In *Inter-Congress Symposium of ISP Commission V*, page Presented paper. Unbound paper No.9 (24 pages), Stockholm, 1978.

[50] A. Grün. Precision and reliability aspects in close range photogrammetry. *Int. Arch. Photogrammetry*, 11(23B):378–391, 1980.

[51] A. Grün. An optimum algorithm for on-line triangulation. In *Symposium of Commission III of the ISPRS*, Helsinki, 1982.

[52] A. Grün. Adaptive least squares correlation — concept and first results. Intermediate Research Report to Helava Associates, Ohio State University. 13 pages, March 1984.

[53] A. Grün. Adaptive kleinste Quadrate Korrelation and geometrische Zusatzinformationen. *Vermessung, Photogrammetrie, Kulturtechnik*, 9(83):309–312, 1985.

[54] A. Grün. Algorithmic aspects of on-line triangulation. *Photogrammetric Engineering & Remote Sensing*, 4(51):419–436, 1985.

[55] A. Grün and E. P. Baltsavias. Adaptive least squares correlation with geometrical constraints. In *SPIE Computer Vision for Robots*, volume 595, pages 72–82, Cannes, 1985.

[56] R. Gupta and R. I. Hartley. Linear pushbroom cameras. *IEEE Trans. Pattern Analysis & Machine Intelligence*, September 1997.

[57] M. S. Gyer. The inversion of the normal equations of analytical aerotriangulation by the method of recursive partitioning. Technical report, Rome Air Development Center, Rome, New York, 1967.

[58] R. Hartley. Euclidean reconstruction from multiple views. In 2^{nd} *Europe-U.S. Workshop on Invariance*, pages 237–56, Ponta Delgada, Azores, October 1993.

[59] R. Hartley. An object-oriented approach to scene reconstruction. In *IEEE Conf. Systems, Man & Cybernetics*, pages 2475–2480, Beijing, October 1996.

[60] R. Hartley. Lines and points in three views and the trifocal tensor. *Int. J. Computer Vision*, 22(2):125–140, 1997.

[61] R. Hartley, R. Gupta, and T. Chang. Stereo from uncalibrated cameras. In *Int. Conf. Computer Vision & Pattern Recognition*, pages 761–4, Urbana-Champaign, Illinois, 1992.

[62] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[63] R. I. Hartley and T. Saxena. The cubic rational polynomial camera model. In *Image Understanding Workshop*, pages 649–653, 1997.

[64] F. Helmert. *Die Mathematischen und Physikalischen Theorien der höheren Geodäsie*, volume 1 Teil. Teubner, Leipzig, 1880.

[65] B. Hendrickson and E. Rothberg. Improving the run time and quality of nested dissection ordering. *SIAM J. Sci. Comput.*, 20:468–489, 1998.

[66] K. R. Holm. Test of algorithms for sequential adjustment in on-line triangulation. *Photogrammetria*, 43:143–156, 1989.

[67] M. Irani, P. Anadan, and M. Cohen. Direct recovery of planar-parallax from multiple frames. In *Vision Algorithms: Theory and Practice*. Springer-Verlag, 2000.

[68] K. Kanatani and N. Ohta. Optimal robot self-localization and reliability evaluation. In *European Conf. Computer Vision*, pages 796–808, Freiburg, 1998.

[69] H. M. Karara. *Non-Topographic Photogrammetry*. Americal Society for Photogrammetry and Remote Sensing, 1989.

[70] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *J. Parallel & Distributed Computing*, 48:96–129, 1998.

[71] G. Karypis and V. Kumar. A fast and highly quality multilevel scheme for partitioning irregular graphs. *SIAM J. Scientific Computing*, 20(1):359–392, 1999. For METIS code see http://www-users.cs.umn.edu/ karypis/.

[72] I. P. King. An automatic reordering scheme for simultaneous equations derived from network systems. *Int. J. Numer. Meth. Eng.*, 2:479–509, 1970.

[73] K. Kraus. *Photogrammetry*. Dümmler, Bonn, 1997. Vol.1: Fundamentals and Standard Processes. Vol.2: Advanced Methods and Applications. Available in German, English & several other languages.

[74] A. M. Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes*. Courcier, Paris, 1805. Appendix on least squares.

[75] R. Levy. Restructuring the structural stiffness matrix to improve computational efficiency. *Jet Propulsion Lab. Technical Review*, 1:61–70, 1971.

[76] M. X. Li. *Hierarchical Multi-point Matching with Simultaneous Detection and Location of Breaklines*. PhD thesis, KTH Stockholm, 1989.

[77] Q.-T. Luong, R. Deriche, O. Faugeras, and T. Papadopoulo. On determining the fundamental matrix: Analysis of different methods and experimental results. Technical Report RR-1894, INRIA, Sophia Antipolis, France, 1993.

[78] S. Mason. Expert system based design of close-range photogrammetric networks. *J. Photogrammetry & Remote Sensing*, 50(5):13–24, 1995.

[79] S. O. Mason. *Expert System Based Design of Photogrammetric Networks*. Ph.D. Thesis, Institut für Geodäsie und Photogrammetrie, ETH Zürich, May 1994.

[80] B. Matei and P. Meer. Bootstrapping a heteroscedastic regression model with application to 3D rigid motion evaluation. In *Vision Algorithms: Theory and Practice*. Springer-Verlag, 2000.

[81] P. F. McLauchlan. Gauge independence in optimization algorithms for 3D vision. In *Vision Algorithms: Theory and Practice*, Lecture Notes in Computer Science, Corfu, September 1999. Springer-Verlag.

[82] P. F. McLauchlan. Gauge invariance in projective 3D reconstruction. In *Multi-View Modeling and Analysis of Visual Scenes*, Fort Collins, CO, June 1999. IEEE Press.

[83] P. F. McLauchlan. The variable state dimension filter. Technical Report VSSP 5/99, University of Surrey, Dept of Electrical Engineering, December 1999.

[84] P. F. McLauchlan. A batch/recursive algorithm for 3D scene reconstruction. In *Int. Conf. Computer Vision & Pattern Recognition*, Hilton Head, South Carolina, 2000.

[85] P. F. McLauchlan and D. W. Murray. A unifying framework for structure and motion recovery from image sequences. In E. Grimson, editor, *Int. Conf. Computer Vision*, pages 314–20, Cambridge, MA, June 1995.

[86] P. F. McLauchlan and D. W. Murray. Active camera calibration for a Head-Eye platform using the Variable State-Dimension filter. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 18(1):15–22, 1996.

[87] P. Meissl. Die innere Genauigkeit eines Punkthaufens. *Österreichische Zeitschrift für Vermessungswesen*, 50(5): 159–165 and 50(6): 186–194, 1962.

[88] E. Mikhail and R. Helmering. Recursive methods in photogrammetric data reduction. *Photogrammetric Engineering*, 39(9):983–989, 1973.

[89] E. Mittermayer. Zur Ausgleichung freier Netze. *Zeitschrift für Vermessungswesen*, 97(11):481–489, 1962.

[90] J. J. Moré and S. J. Wright. *Optimization Software Guide*. SIAM Press, Philadelphia, 1993.

[91] D. D. Morris and T. Kanade. A unified factorization algorithm for points, line segments and planes with uncertainty. In *Int. Conf. Computer Vision*, pages 696–702, Bombay, 1998.

[92] D. D. Morris, K. Kanatani, and T. Kanade. Uncertainty modelling for optimal structure and motion. In *Vision Algorithms: Theory and Practice*. Springer-Verlag, 2000.

[93] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, 1999.

[94] M. Okutomi and T. Kanade. A multiple-baseline stereo. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 15(4):353–363, 1993.

[95] D. W. Proctor. The adjustment of aerial triangulation by electronic digital computers. *Photogrammetric Record*, 4:24–33, 1962.

[96] B. D. Ripley. *Pattern Recongition and Neural Networks*. Cambridge University Press, 1996.

[97] D. Rosenholm. Accuracy improvement of digital matching for elevation of digital terrain models. *Int. Arch. Photogrammetry & Remote Sensing*, 26(3/2):573–587, 1986.

[98] S. Roy and I. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *Int. Conf. Computer Vision*, Bombay, 1998.

[99] Y. Saad. On the rates of convergence of Lanczos and block-Lanczos methods. *SIAM J. Numer. Anal.*, 17:687–706, 1980.

[100] C. C. Slama, editor. *Manual of Photogrammetry*. American Society of Photogrammetry and Remote Sensing, Falls Church, Virginia, USA, 1980.

[101] R. A. Snay. Reducing the profile of sparse symmetric matrices. *Bulletin Géodésique*, 50:341–352, 1976. Also NOAA Technical Memorandum NOS NGS-4, National Geodetic Survey, Rockville, MD.

[102] R. Szeliski, S. B. Kang, and H. Y. Shum. A parallel feature tracker for extended image sequences. Technical Report CRL 95/2, DEC Cambridge Research Labs, May 1995.

[103] R. Szeliski and S.B. Kang. Shape ambiguities in structure from motion. In *European Conf. Computer Vision*, pages 709–721, Cambridge, 1996.

[104] R. Szeliski and H.Y. Shum. Motion estimation with quadtree splines. In *Int. Conf. Computer Vision*, pages 757–763, Boston, 1995.

[105] B. Triggs. A new approach to geometric fitting. Available from http://www.inrialpes.fr/movi/people/Triggs, 1997.

[106] B. Triggs. Optimal estimation of matching constraints. In R. Koch and L. Van Gool, editors, *3D Structure from Multiple Images of Large-scale Environments SMILE'98*, Lecture Notes in Computer Science. Springer-Verlag, 1998.

[107] G. L. Strang van Hees. Variance-covariance transformations of geodetic networks. *Manuscripta Geodaetica*, 7:1–20, 1982.

[108] X. Wang and T. A. Clarke. Separate adjustment of close range photogrammetric measurements. *Int. Symp. Photogrammetry & Remote Sensing*, XXXII, part 5:177–184, 1998.

[109] P. R. Wolf and C. D. Ghilani. *Adjustment Computations: Statistics and Least Squares in Surveying and GIS*. John Wiley & Sons, 1997.

[110] B. P. Wrobel. Facets stereo vision (FAST vision) — a new approach to computer stereo vision and to digital photogrammetry. In *ISPRS Intercommission Conf. Fast Processing of Photogrammetric Data*, pages 231–258, Interlaken, Switzerland, June 1987.

Chapter 6

Pattern Recognition & Statistics

This chapter contains two papers on statistical modelling and pattern recognition, written jointly with my PhD student Guillaume Bouchard.

Summary of paper 16, "The Trade-off between Generative and Discriminative Classifiers"

This paper was presented at COMPSTAT'04, the IASC International Symposium on Computational Statistics [BT04b]. It gives a likelihood-based derivation of the difference between discriminatively-trained and generatively-trained classifiers, and proposes an intermediate approach that is sometimes better than both. The basic idea is that discriminatively-trained classifiers are trained to maximize the expected log-likelihood of $p(\text{class} \mid \text{descriptors})$, while generatively-trained ones maximize $p(\text{class}, \text{descriptors}) = p(\text{class} \mid \text{descriptors})\, p(\text{descriptors})$. So we can interpolate between the two by modulating the underlying class-independent likelihood $p(\text{descriptors})$.

Summary of paper 17, "Hierarchical Part-Based Visual Object Categorization"

This paper has been submitted to the 2005 Conference on Computer Vision and Pattern Recognition (CVPR) [BT04a]. It describes a statistically motivated, spatially structured local feature based approach to visual object recognition. Local features are small distinctive image patches representing possible fragments of a detected object. Several types of local feature detectors exist. Currently, two of the most successful approaches to visual object recognition are: (*i*) "bag of local features" approaches that treat the detected features essentially as independent weak classifiers for object presence, and combine them using voting, naive Bayes, boosting, and similar ensemble classifier schemes; and (*ii*) "constellation" models, that try to capture spatial dependencies by using a few specific local features, and building a joint spatial density model for them. The bag model is spatially weak but able to exploit large numbers (hundreds) of weakly-informative features, while the constellation model is spatially strong but limited to only a few (6 or so) relatively informative ones. The current paper adopts an intermediate approach, using hundreds of relatively weak local features, but enforcing at least some spatial coherence by learning hierarchical, spatially organized,

part-subpart based object models. The overall model is expressed as a Gaussian mixture over appearances and positions and learned using EM. The method is tested on several classes from the popular Caltech test sets.

THE TRADE-OFF BETWEEN GENERATIVE AND DISCRIMINATIVE CLASSIFIERS

Guillaume Bouchard and Bill Triggs

Key words: Statistical computing, numerical algorithms.

COMPSTAT 2004 section: Classification.

Abstract: Given any generative classifier based on an inexact density model, we can define a discriminative counterpart that reduces its asymptotic error rate. We introduce a family of classifiers that interpolate the two approaches, thus providing a new way to compare them and giving an estimation procedure whose classification performance is well balanced between the bias of generative classifiers and the variance of discriminative ones. We show that an intermediate trade-off between the two strategies is often preferable, both theoretically and in experiments on real data.

1 Introduction

In supervised classification, inputs x and their labels y arise from an unknown joint probability $p(x, y)$. If we can approximate $p(x, y)$ using a parametric family of models $\mathcal{G} = \{p_\theta(x, y), \theta \in \Theta\}$, then a natural classifier is obtained by first estimating the class-conditional densities, then classifying each new data point to the class with highest posterior probability. This approach is called *generative* classification.

However, if the overall goal is to find the classification rule with the smallest error rate, this depends only on the conditional density $p(y|x)$. *Discriminative* methods directly model the conditional distribution, without assuming anything about the input distribution $p(x)$. Well known generative-discriminative pairs include Linear Discriminant Analysis (LDA) vs. Linear logistic regression and naive Bayes vs. Generalized Additive Models (GAM). Many authors have already studied these models e.g. [5,6]. Under the assumption that the underlying distributions are Gaussian with equal covariances, it is known that LDA requires less data than its discriminative counterpart, linear logistic regression [3]. More generally, it is known that generative classifiers have a smaller variance than.

Conversely, the generative approach converges to the best model for the joint distribution $p(x, y)$ but the resulting conditional density is usually a biased classifier unless its $p_\theta(x)$ part is an accurate model for $p(x)$. In real world problems the assumed generative model is rarely exact, and asymptotically, a discriminative classifier should typically be preferred [9, 5]. The key argument is that the discriminative estimator converges to the conditional density that minimizes the negative log-likelihood classification loss against the true density $p(x, y)$ [2]. For finite sample sizes, there is a bias-variance tradeoff and it is less obvious how to choose between generative and discriminative classifiers.

In this paper, we will first consider the parameter estimation problem, focusing on the theoretical distinction between generative and discriminative classifiers. Then we propose a new technique for combining the two classifiers: the Generative-Discriminative Trade-off (GDT) estimate. It is based on a continuous class of

Published in the Symposium of the Int. Assoc. for Statistical Computing, CompStat 2004.

cost functions that interpolate smoothly between the generative strategy and the discriminative one. Our method assumes a joint density based parametrization $p_\theta(x, y)$, but uses this to model the conditional density $p(x|y)$. The goal is to find the parameters that maximize classification performance on the underlying population, but we do this by defining a cost function that is intermediate between the joint and the conditional log-likelihoods and optimizing this on training and validation sets.

Given that the generative model based on maximum likelihood (ML) produces minimum variance — but possibly biased — parameter estimates, while the discriminative one gives the best asymptotic classification performance, there are good reasons for thinking that an intermediate method such as the GDT estimate should be preferred. We illustrate this on simulations and on real datasets.

2 Preliminaries

Using independent training samples $\{x_i, y_i\}, i = 1, \ldots, n$, $x_i \in \mathbb{R}^d$,and $y_i \in \{1, \ldots, K\}$ sampled from the unknown distribution $p(x, y)$, we aim to find the rule that gives the lowest error rate on new data. This is closely related to estimating the conditional probability $p(y|x)$.

For each of the K classes, the class-conditional probability $p(x|y = k)$ is modeled by a parametric model f_k with parameters θ_k. The y follows a multinomial distribution with parameters $p_1, \ldots, p_K$. The full parametrization of the joint density is $\theta = (p_1, \ldots, p_K, \theta_1, \ldots, \theta_K)$. Given θ, new data points x are classified to the group k giving the highest posterior probability

$$P_\theta(Y = k|X = x) = \frac{p_k f_k(x; \theta_k)}{\sum_{l=1}^{K} p_l f_l(x_i; \theta_l)}. \tag{1}$$

The generative and the discriminative approaches differ only in the estimation of θ.

Generative classifier. Given data $\{x_i, y_i\}$, $i = 1, \ldots, n$, a standard way to estimate the parameters of densities is the Maximum Likelihood (ML) estimate (we assume that the solution is unique):

$$\hat{\theta}_{GEN} = \arg\max_{\theta \in \Theta} \mathcal{L}_{GEN}(\theta), \qquad \mathcal{L}_{GEN}(\theta) = \sum_{i=1}^{n} \log p_{y_i} f_{y_i}(x_i; \theta). \tag{2}$$

Discriminative classifier. Let $\mathcal{D} = \{p_\theta(y|x) = p_\theta(x, y)/\sum_z p_\theta(x, z), \theta \in \Theta\}$ be the set of conditional densities derived from the generative model. Our aim is to find the conditional density in $\mathcal{D}$ that minimizes a classification loss function on the training set. Here, we consider only the negative conditional log-likelihood $-\mathcal{L}_{DISC}$, which can be viewed as a convex approximation to the error rate:

$$\hat{\theta}_{DISC} = \arg\max_{\theta \in \Theta} \mathcal{L}_{DISC}(\theta), \qquad \mathcal{L}_{DISC}(\theta) = \sum_{i=1}^{n} \log \frac{p_{y_i} f_{y_i}(x_i; \theta)}{\sum_k p_k f_k(x_i; \theta)}. \tag{3}$$

The discriminative approach allows to eliminate parameters that influence only $p(x)$,not $p(y|x)$ (e.g. shared covariance matrix in Gaussian distributions), leading to logistic regression over lower dimensional parameter spaces. However, we will not use this reduction, as we need to maintain a common parametrization for the

discriminative and generative cases. thus, the solution (3) of the discriminative classifier may not be unique — there may exist infinitely many parameters that give the same conditional distribution $p_\theta(x|y)$. However, the classification performance is the same for all such solutions.

Relationship. The quantity $\mathcal{L}_{DISC}$ can be expanded as follows:

$$\mathcal{L}_{DISC}(\theta) = \underbrace{\sum_{i=1}^{n} \log p_{y_i} f_{y_i}(x_i;\theta)}_{\mathcal{L}_{GEN}(\theta)} - \underbrace{\sum_{i=1}^{n} \log \sum_{k=1}^{K} p_k f_k(x_i;\theta)}_{\mathcal{L}_x(\theta)} \tag{4}$$

The difference between the generative and discriminative objective functions $\mathcal{L}_{GEN}$ and $\mathcal{L}_{DISC}$ is thus $\sum_{i=1}^{n}\sum_k \log p_\theta(x_i,k)$, the log-likelihood of the input space probability model $p_\theta(x)$. Equation (4) shows that compared to the discriminative approach, the generative strategy tends to favor parameters that give high likelihood on the training data.

3 Between Generative and Discriminative classifiers

To get a natural trade-off between the two approaches, we can introduce a new objective function $\mathcal{L}_\lambda$ based on a parameter $\lambda \in [0,1]$ that interpolates continuously between the discriminative and generative objective functions:

$$\begin{aligned} \mathcal{L}_\lambda(\theta;\boldsymbol{x},\boldsymbol{y}) &= \mathcal{L}_{GEN}(\theta;\boldsymbol{x},\boldsymbol{y}) - (1-\lambda)\mathcal{L}_x(\theta;\boldsymbol{x}) &(5)\\ &= \lambda\mathcal{L}_{GEN}(\theta) + (1-\lambda)\mathcal{L}_{DISC}(\theta). &(6)\end{aligned}$$

For $\lambda \in [0,1]$, the GDT estimate is

$$\hat{\theta}_\lambda = \arg\max_{\theta\in\Theta} \mathcal{L}_\lambda(\theta). \tag{7}$$

Taking $\lambda = 0$ leads to the discriminative estimate $\hat{\theta}_{DISC}$, while $\lambda = 1$ leads to the generative one $\hat{\theta}_{GEN}$. We expect that the GDT estimates $\hat{\theta}_\lambda$ $(0 < \lambda < 1)$ will sometimes have better generalization performances than these two extremes. Even if the discriminative estimate (3) is not unique, the maximum of (7) is unique for all $\lambda \in [0,1)$ if the ML estimate $\hat{\theta}_{GEN}$ is unique.

Computation of $\hat{\theta}_\lambda$. Since we use a differentiable classification loss, the maximization problem (7) can be solved by any gradient ascent method. The Newton algorithm converges rapidly, but requires the computation of the Hessian matrix, The Conjugate Gradient (CG) algorithm may be more suitable for large scale problems: it needs only the first derivative and it is possible to avoid the storage of the quasi-Hessian matrix which can be huge when the number of parameters is large.

For simplicity, we assume that the parameters θ_k of the different class densities are independent. Taking the derivative of (5) with respect to θ_k and π_k, we get

$$\begin{cases} \frac{\partial}{\partial\theta_k}\mathcal{L}_\lambda(\theta_k) = \sum_{i=1}^{n}(\mathbf{I}_{\{y_i=k\}} - (1-\lambda)\tau_{ki})\frac{\partial \log f_k(x_i;\theta_k)}{\partial\theta_k} \\ \frac{\partial}{\partial\pi_k}\mathcal{L}_\lambda(\theta_k) = \frac{1}{\pi_k}\left(n_k - (1-\lambda)\sum_{i=1}^{n}\tau_{ki}\right) \end{cases} \tag{8}$$

with $n_k = \sum_{i=1}^{n} \mathbf{I}_{\{y_i=k\}}$ and $\tau_{ki} = \frac{\pi_k f_k(x_i;\theta_k)}{\sum_{l=1}^{K} \pi_l f_l(x_i;\theta_l)}$. The optimal parameters are zeros of the equations (8) for $k = 1, \ldots, K$.

For a given class k, these equations are analogous to the ML equations on weighted data, although unlike ML, the weights can be negative here Each point has a weight $\mathbf{I}_{\{y_i=k\}} - (1-\lambda)\tau_{ki}$. The examples that have most influence on the θ_k-gradient are those that belong to the class k but have a low probability to be in it (τ_{ki} is small), and conversly those that do not belong to the class k but that are assigned to it with a high probability. The influence of the assignment probabilities is controlled by the parameter λ. This remark may ultimately help us to link our approach to boosting, and similar algorithms that iteratively reweight misclassified data. It also shows that the generative estimator (λ=1) is not affected by the classification rate of the data points.

Choice of λ. The GDT estimate contains a tuning parameter to set, which functions like the smoothing parameter in regularization methods. λ cannot be set on the basis of minimum classification loss on the training set, since by definition, $\lambda = 0$ gives the optimal θ for training set classification. Instead, λ is set to the value $\hat{\lambda}$ that minimizes the cross-validation error rate.

If the optimal $\hat{\lambda}$ is close to one, the generative classifier is preferred. This suggests that the bias in $p_\theta(x, y)$ (if any) does not affect the discrimination of the model too much. Similarly, if $\hat{\lambda}$ is close to 1, it suggests that the model $p_\theta(x, y)$ does not fit the data well, and the bias of the generative classifier is too high to provide good classification results. In this case, a more complex model — i.e. with more parameters, or less constrained — may be needed to reduce the bias. For other $\hat{\lambda}$, there is an equilibrium between the bias and the variance, meaning that the model complexity is well adapted to the amount of training data.

4 Simulations

To illustrate the behavior of the GDT method, we study its performance on two synthetic test problems. We define the true distributions of the data as follows: In the first experiment, the class conditional probabilities are gaussian with identity covariance matrix and means $m_1 = (1.25, 0, 0, 0)$ and $m_0 = (-1.25, 0, 0, 0)$. In the second case, we simulate x according to a uniform density with correlated covariates : $x^{(1)} \sim \mathcal{U}[0; 1]$ and $x^{(d)} \sim \mathcal{U}[x^{(d-1)}; 1 + x^{(d-1)}]$ with $d \in \{2, 3, 4\}$ and $x^{(i)}$ denotes the i^{th} covariate. Then $y|x$ is simulated according to a Bernoulli distribution with parameter $1/\exp(-2.5x^{(1)})$. Note that the linear logistic model is true in the two experiments.

The assumed model is a Gaussian distribution for each class with shared diagonal covariance matrices and prior probabilities equal to $\frac{1}{K}$. Hence, the model does not correspond exactly to the true density in the second experiment, but it can provide a good approximation when the differences between the variances are small.

In each case, we estimated the true error rate of the classifiers learned on training samples of size 50, 100 and 200. The results are plotted in figure (1). We used standard plug-in estimates for $\lambda = 1$ and closed form logistic regression for $\lambda = 0$. For intermediate estimates, the conjugate gradient method was used. The first row illustrates the fact that the generative classifier performs better than the other estimates, but this difference tends to decrease when the sample size

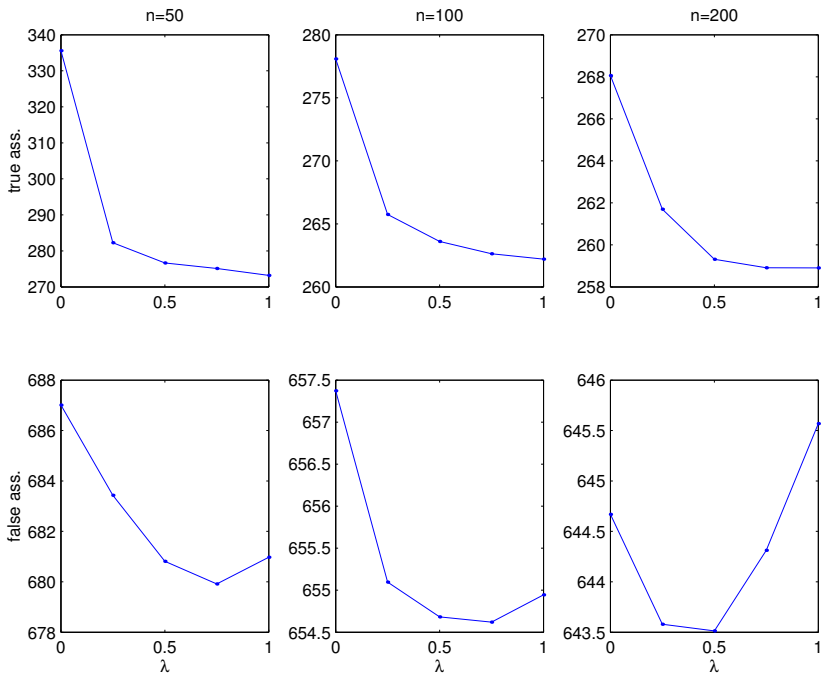


Figure 1: The full lines plot logistic loss computed on test sets of size 10^5 against the tuning parameter λ. Each plotted value is the median of 200 experiments. The rows correspond to the first and second simultations. The columns correspond to different training sample sizes.

increases. In the second row, the best performance is from the BDG estimate for all training set sizes, and the optimal value of λ (the one that minimizes the expected loss) decreases with n since we know that the discriminative approach becomes optimal when n tends to infinity.

5 Experiments

We tried our classification method on some of the publically available Statlog datasets. In our implementation of the GDT estimates, the parameter dimension is limited due to the size of the optimization problem (7). To make the computation feasible, we reduced the dimension of the data by computing the first four Fisher discriminant variables and using them as inputs (when the number of classes was less than 5, so that there were fewer than four discriminant directions, we computed the remaining directions by PCA on the residuals). These directions are computed using the training data and do not involve the test data.

We tried four types of density for the class-conditional distributions: 1. Gaussian densities with common covariance matrix (LDA), 2. Gaussian densities with unconstrained covariance (QDA), 3. Gaussian densities with spherical covariance matrix (Balls1), 4. Mixture of two Gaussian densities with spherical covariance matrix (Balls2). These distributions do not exactly fit the data, but they are distributions that are often used to approximate real datasets. Therefore, when the training sample is small, the generative approach may still behave better than

Dataset	australian	diabetes	heart	satimage	vehicle
Training size	100	100	100	300	200
LDA GEN	**0.143**	0.253	0.178	0.188	0.237
LDA GDT0.75	0.144	0.252	0.178	0.187	0.235
LDA GDT0.5	0.144	**0.249**	0.179	0.186	0.235
LDA GDT0.25	0.144	0.250	0.182	0.185	0.236
LDA DISC	0.145	0.249	0.185	0.191	0.243
QDA GEN	0.149	0.262	0.181	0.181	0.235
QDA GDT0.75	0.151	0.261	0.182	**0.179**	0.234
QDA GDT0.5	0.150	0.262	0.181	0.180	0.235
QDA GDT0.25	0.151	0.262	0.182	0.181	0.234
QDA DISC	0.168	0.270	0.204	0.215	0.267
Balls1 GEN	0.146	0.262	0.168	0.185	0.318
Balls1 GDT0.75	0.145	0.260	0.167	0.183	0.293
Balls1 GDT0.5	0.144	0.259	**0.165**	0.182	0.271
Balls1 GDT0.25	0.144	0.257	0.169	0.181	0.254
Balls1 DISC	0.150	0.253	0.190	0.194	0.242
Balls2 GEN	0.146	0.266	0.181	0.185	0.239
Balls2 GDT0.75	0.145	0.265	0.180	0.185	0.239
Balls2 GDT0.5	0.146	0.265	0.180	0.184	0.236
Balls2 GDT0.25	0.146	0.268	0.181	0.183	**0.232**
Balls2 DISC	0.166	0.279	0.211	0.210	0.250

Table 1: Test error rate on real datasets, averaged over 100 trials. For each trial, training data were randomly chosen and the error rate was computed on the remaining data. In the *heart* dataset, a misclassified heart disease has a cost of 5 instead of 1.

the discriminative one. Training sample sizes were set to 50 times the number of classes so the discriminative classifiers should not have reached their asymptotic behavior.

We used a Cholesky-based parametrization of the inverse covariance matrix, so there was no need for a separate positivity constraint on the parameters. Derivatives with respect to this parametrization were obtained for each density, and we used the generative solution — which is explicit for densities 1-3 and obtained by the EM algorithm for the densities 4 — to initialize the CG algorithm.

Table (1) shows the generalization performance for each dataset and each model with different values of λ. These results show that substantial improvements in the classification rate can be obtained for intermediate values of λ. However, they do not directly show the performance of the GDT estimate because we fixed λ rather than selecting it by cross-validation on the training set. The evaluation of the cost as a function of λ could be used as a model selection criterion. For example, on the vehicle dataset, the simple Gaussian model (Balls1) gives an optimal λ equal to 0. This suggests that the bias is dominating the error, and indeed the results are improved by using two Gaussian densities for each class (Balls2).

One can object that the gain in error rate in these experiments is not sufficient to really conclude the usefulness of the GDT estimator.

6 Conclusion

In this study, the relationship between generative and discriminative classifiers has been clarified: they correspond to two different maximizations in the parameter space. By interpolating linearly between the two objective functions, we introduced the GDT estimator. This can be seen either as a less biased variant version of the discriminative solution, or as an improvement of the generative classifier. The regularization is "natural" in the sense that the parameters are encouraged to fit the inputs. Our preliminary results on real data showed that the intermediate model often gives better classification performances than the discriminative and generative classifiers.

The real interest of the GDT estimate resides in its application to generative models. Probabilistic models already exist in many areas: time series models, mixed models and graphical models — including Markov Random Fields and Hidden Markov Models — are examples of widely used generative models. When class-conditional probabilities are modelled generatively, then the GDT estimator should often improve the classification performances.

Currently, the main difficulty with the GDT method is the choice of the tuning parameter, as this requires an expensive cross-validation computation. We believe that more computationally efficient criteria can be developed by analyzing the solutions on the training set, in the spirit of the Bayesian Information Criterion [7].

References

[1] Devroye L., Györfi L. & Lugosi L. (1997) *A* probabilistic Theory of Pattern Recognition. pp. 270-276 New York: Springer-Verlag.

[2] Efron. B. (1975) The efficiency of logistic regression compared to Normal Discriminant Analysis. *Journ. of the Amer. Statist. Assoc.*, 70:892-898.

[3] Ng A.Y. & Jordan M.I. (2002) On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In T. Dietterich, S. Becker and Z. Ghahramani (Eds.), *A*dvances in Neural Information Processing Systems 14, pp. 609-616. Cambridge, MA: MIT Press.

[4] Rubinstein Y.D. & Hastie T. (1997) Discriminative vs. informative learning. *I*n Proc. of the Third International Conference on Knowledge and Data Mining, pp. 49-53. AAAI Press.

[5] Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics 6,* pp. 461-464.

[6] Vapnick, V.N. (1998) *Statistical Learning Theory.* John Wiley & Sons.

Acknowledgement: We thank G. Celeux, for valuable discussions. This work was supported by the European LAVA project. *Address*: IS2 project
INRIA
38334 Saint-Ismier Cedex
`Guillaume.Bouchard@inria.fr`
E-mail: `Guillaume.Bouchard@inria.fr`

Hierarchical Part-Based Visual Object Categorization

Guillaume Bouchard and Bill Triggs
LEAR, GRAVIR-INRIA, 655 av. de l'Europe, 38330 Montbonnot, France
Guillaume.Bouchard@inria.fr, Bill.Triggs@inrialpes.fr

Abstract

We propose a generative model that codes the geometry and appearance of generic visual object categories as a loose hierarchy of parts, with probabilistic spatial relations linking parts to subparts, soft assignment of subparts to parts, and scale invariant keypoint based local features at the lowest level of the hierarchy. The method is designed to efficiently handle categories containing hundreds of redundant local features, such as those returned by current keypoint detectors. This robustness allows it to outperform constellation style models, despite their stronger spatial models. The model is initialized by robust bottom-up voting over location-scale pyramids, and optimized by Expectation-Maximization. Training is rapid, and objects do not need to be marked in the training images. Experiments on several popular datasets show the method's ability to capture complex natural object classes.

Keywords: *visual categorization, object recognition, generative models, local features.*

1 Introduction

In object categorization from digital images, existing geometrical models are typically very specific to a particular object class (for example 3D human body models). There is a need for generic models that are suitable for more general object categories. "Part" or "fragment" based models that combine local image features or regions into loose geometric assemblies offer one possible solution to this [9,11,5,4,8]. Constellation models [5,4] provide a probabilistic way to mix the appearance and location of local descriptors. One of their major limitations is the fact that they require an explicit enumeration over possible matchings of model features to image ones. This optimal, but combinatorially expensive, step limits the model to relatively few detected features ('parts'), typically 6 or at most 7. This in turn means that a good deal of the available image information must often be ignored, especially in cases where the objects have many parts, either naturally, or because fine grained local visual features are being used to characterize them. Indeed, such structural approaches often fail to compete with geometry-free "bag of features" style approaches because the latter make better use of the available image information [9,10,1]. Hence it is useful to investigate structural models that can handle models with hundreds of local features efficiently.

Secondly, many natural object categories (humans and animals, man made classes with variable forms) have relatively rigid local shape, but significant large scale shape variability, so that nearby object features have strongly correlated positions while more distant ones are much more weakly correlated. But these correlations are not always local and can be very complex, as it can be seen in human face expressions and 3D objects having small pose variations, for which a part-based model can approximate the pixels displacement at different depths. Another advantage of part-based models is that they can easily represent this kind of covariance structure. But to do this well, it is natural to include some levels of part hierarchy, with loosely connected parts containing more tightly connected subparts. Hence the overall model becomes a tree-structured graphical model [7].

In this paper, we propose a hierarchical model that is capable of handling hundreds of feature classes efficiently, so that the model is suitable for use with very basic feature detectors. The position of the object in the training images and the model structure are unknown and treated as hidden variables, to be estimated using E-M after a suitable initialization. The method is totally scale-invariant, and all of the model parameters are learned by maximum likelihood, so the only tuning needed is the number of parts at each level. Cross-validation shows that using multi-part models is often advantageous.

Below, we first present the probabilistic model. Then the learning method, including including initialization and EM steps, is explained. Finally experiments on real images show that the model is effective for object categorization.

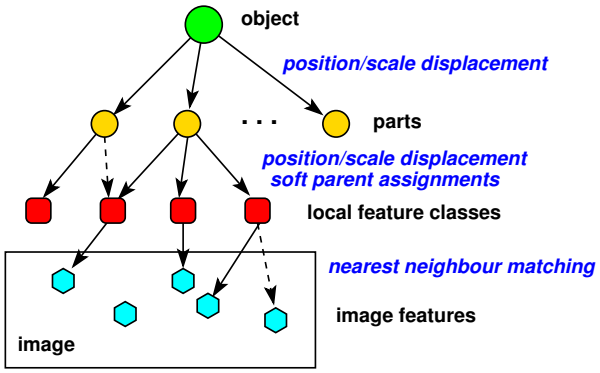


Figure 1: The overall structure of our hierarchical object model.

2 Model Structure

Our model (see figure 1) is a hierarchy of parts and subparts, with the object at the top level, and position-appearance classes of local image features that will be attached to observed image features at the bottom. In each layer of the hierarchy, the parts are softly assigned to parents from the preceding layer. Soft assignment is included mainly to help the model structure adapt to the object class during training: once the models have been learned, most of the parts tend to have relatively certain parent assignments.

Spatial structure: Parts and their sub-trees are attached to their parents by uncertain spatial transformations. In the experiments below, we have used translations and rescalings, *i.e.* transformations of the form $\mathbf{T}_{qp} = \begin{pmatrix} s & 0 & u \\ 0 & s & v \\ 0 & 0 & 1 \end{pmatrix}$ where s is the relative scale and (u, v) is the relative translation of part p relative to its parent q from the previous layer[1]. We assume that $\mathbf{T}_{qp}$ is sampled from a Normal distribution over translations and a log-Normal distribution over relative scales. We write the corresponding mean and variance symbolically as $\overline{\mathbf{T}}_{qp}$ and $\mathbf{Var}(\mathbf{T}_{qp})$. These are model parameters that need to be learned. Formally, $\overline{\mathbf{T}}_{qp}$ is a non-random transformation and $\mathbf{Var}(\mathbf{T}_{qp})$ can be thought of as a 3×3 covariance matrix for $(u, v, \log s)$, which is assumed to be diagonal below.

There is a minor complexity relating to the fact that we use soft parent assignments. We do *not* introduce separate model parameters for the transformation of each part relative to each of its possible parents. This would be a huge number of parameters, and those with low parenting probabilities would not be estimated stably. Instead, we expect parts to represent stable, identifiable regions of the object class with their own identities and positions. Parent attributions are uncertain only because it is unclear before training which parent best codes the target part's overall position variability, *i.e.* parts are essentially assigned to the parent whose spatial position variations best explain (covary most strongly over the training set with) their own. To capture this notion, each part p is given just *one* set of mean transformation parameters $\overline{\mathbf{T}}_p$, representing the mean position of p relative the root of the object frame, and a corresponding set of (reduced) variance parameters[2] $\mathbf{Var}(\mathbf{T}_p)$. Given a parent attribution q for p, the uncertain transformation $\mathbf{T}_{qp}$ is then sampled with mean $\overline{\mathbf{T}}_{qp} \equiv \overline{\mathbf{T}}_q^{-1} \overline{\mathbf{T}}_p$ and the correspondingly back-transformed variance, which we can denote by $\overline{\mathbf{T}}_q^{-1}(\mathbf{Var}(\mathbf{T}_p))$ say. (In our case, this is just the 3×3 $(u, v, \log s)$ covariance $\mathbf{Var}(\mathbf{T}_p)$ with its (u, v) block scaled by $1/\overline{s}_q^2$). In this way, the same few parameters control the part's position, whatever its parent assignment. If we suppose that the (random) parent locations $\mathbf{T}_q$ are already known, the part location relative to the object frame is a mixture of random transformations $\mathbf{T}_q \mathbf{T}_{qp}$, where $\mathbf{T}_{qp}$ is a random transformation (Gaussian in $(u, v, \log s)$) and the mixture weights are $\boldsymbol{\tau}_p(q)$, the model parameters representing the prior probabilities of p's parent being q:

$$\mathbf{p}_p^{\text{loc}}(\mathbf{T}_p \,|\, \{\mathbf{T}_q\}) = \sum_q \boldsymbol{\tau}_p(q)\, \mathcal{N}(\mathbf{T}_q \mathbf{T}_p^{-1} \,|\, \overline{\mathbf{T}}_{qp}, \mathbf{Var}(\mathbf{T}_{qp})) \tag{1}$$

This mixture has the peculiarity that if all of the possible parents q are in their mean positions $\overline{\mathbf{T}}_q$, all of its components coincide exactly — it becomes multimodal only when several parents have nonzero mixing proportions $\boldsymbol{\tau}_p(q)$ and deviate from their means.

Image correspondence: The lowest level of the spatial hierarchy contains elementary parts representing appearance classes of scale-invariant local features, similar to those used in other constellation and bag of features models [12,5,4,3,9,10,1]. When the model is in use, each elementary part acts as a "bonding site" for a nearby image feature of similar appearance. Image features are characterized by their locations (positions and scales) and their appearance vectors $\mathbf{a}$. In the experiments below they are SIFT descriptors calculated over scale-invariant Harris keypoints [9,10], but any other feature / descriptor combination could be used. Each elementary part p has the usual location model $\mathbf{T}_p$, and also a corresponding feature appearance model — here, a Gaussian with

[1] $\mathbf{T}_{qp}$ is the transformation taking point coordinates in the frame of p to point coordinates in the frame of q, *e.g.* a point at the origin of p with scale 1 has scale s and position (u, v) with respect to q.

[2] This is *not* the variance of the full uncertain transformation $\mathbf{T}_p$, just the part of this variance that is introduced at the level of part p.

model parameters $\overline{\mathbf{a}}_p$ and $\mathbf{Var}(\mathbf{a}_p)$. When an image feature is bound to an elementary part, the part's location is instantiated to the feature's location and scale, and its appearance is instantiated to the feature's appearance. The model is designed to support large numbers (hundreds) of elementary parts, only some of which are seen in any given image. So it is important to allow parts to remain effectively unassigned. In practice we nominally assign every part to some feature, but we use a robust assignment probability that effectively discounts any overly distant assignments:

$$\mathbf{p}_p(\mathbf{a}, \mathbf{T}) = (1 - \boldsymbol{\pi}_p)\, \mathbf{p}_{\mathrm{bkgd}}(\mathbf{a}, \mathbf{T}) + \boldsymbol{\pi}_p\, \mathbf{p}_p^{\mathrm{app}}(\mathbf{a})\, \mathbf{p}_p^{\mathrm{loc}}(\mathbf{T}) \tag{2}$$

Here: $\mathbf{a}$, $\mathbf{T}$ are the appearance and location of the assigned feature f; $\boldsymbol{\pi}_p$ is a learned inlier proportion for elementary part / feature class p; $\mathbf{p}_{\mathrm{bkgd}}$ is a background model, uniform in appearance and position; $\mathbf{p}_p^{\mathrm{app}}$ is p's appearance model, a Gaussian with mean $\overline{\mathbf{a}}_p$ and variance $\mathbf{Var}(\mathbf{a}_p)$; and $\mathbf{p}_p^{\mathrm{loc}}$ is the above mentioned spatial mixture over p's location, parametrized by $\overline{\mathbf{T}}_p, \mathbf{Var}(\mathbf{T}_p)$, the corresponding parameters of all p's parents, grandparents, *etc*., and the corresponding mixing proportions $\boldsymbol{\tau}_p(q)$ for all parents q, *etc*.

When the model is in use, each elementary model part is bound to the single observed image feature that is most probable according to the above likelihood model given the current model parameters.

One could also use soft assignments to several nearby image features. This would be more consistent with our overall philosophy, but at present we do not do it, mainly because it would make part-feature matching much less efficient.

During testing, we do nothing to prevent several elementary parts from binding to the same image feature. This is again for efficiency reasons — otherwise a combinatorial matching process would be needed to find the best set of correspondences. However during model training we *do* enforce unique assignments by greedy matching, as otherwise the learned appearance classes of nearby parts tend to merge.

We effectively ignore any unbound features (sometimes even the majority of the features detected). This prevents problems when there are multiple detections of essentially the same feature, but it also means that the current model has no efficient means of representing textured regions. We are currently investigating the use of a Poisson field binding model, where elementary parts can bind to many similar features at once. This also suggests that the model may have problems with hallucinations in very cluttered regions where many types of features occur.

3 Training

The model is fitted to a given image, and also trained over the full training set, using Expectation-Maximization. The model parameters to be adjusted during training are: for each part, the mean and variance of its location and scale, $\overline{\mathbf{T}}_p, \mathbf{Var}(\mathbf{T}_p)$, and its vector of parent assignment probabilities $\boldsymbol{\tau}_p$; and for each elementary part, the mean and variance of its feature appearance $\overline{\mathbf{a}}_p, \mathbf{Var}(\mathbf{a}_p)$, and its probability of occurrence $\boldsymbol{\pi}_p$. In addition, in each image there are continuous hidden variables to be estimated for the part locations $\mathbf{T}_p$; and discrete ones for the elementary part to feature bindings, the background / foreground decision for each bound feature, and the parent assignment for each part.

The E-M processes are straightforward to implement. Once initialized, the method converges in about 5–10 iterations. Training takes around 1 second per image in MATLAB, most of this time being spent in the (currently unoptimized) part to feature assignment process. Note that every parameter of the model is learned using E-M: apart from the number of parts in each layer and some thresholds used during initialization, the model has no hand-set parameters, regularization terms, *etc*.

3.1 Instantiating the Model in an Image

The effective cost function has many local minima so a robust instantiation method is needed. We use a hierarchical, Hough transform like heuristic voting method, based on voting into a position/scale pyramid of possible locations ($\mathbf{T}_p$ values) for each part.

1. For each part q in the penultimate layer of the hierarchy (the direct parents of the elementary parts), each image feature f (with appearance $\mathbf{a}_f$ and location $\mathbf{T}_f$) votes into a position/scale pyramid for q's location $\mathbf{T}_q$, essentially by taking the expected mixture distribution over feature appearances and locations generated by q's elementary subparts, and using it backwards as a likelihood for voting:

$$\mathrm{Vote}_q(\mathbf{T}_q) \;=\; \sum_f \max_p \frac{\boldsymbol{\tau}_p(q)}{w_p}\, \mathbf{p}_p^{\mathrm{app}}(\mathbf{a}_f)\, \mathbf{p}_p^{\mathrm{loc}}(\mathbf{T}_f \mid \mathbf{T}_q) \tag{3}$$

$$w_p \;\equiv\; \sum_f \mathbf{p}_p^{\mathrm{app}}(\mathbf{a}_f) \tag{4}$$

   Here the sum is over features f, and the maximum is over the elementary parts p whose best parent is q. For speed, the vote uses just the best elementary part attribution p for f. Note that we re-weight each elementary part's vote by the estimated number

of image features assigned to its appearance class, $w_p = \sum_f \mathbf{p}_p^{\text{app}}(\mathbf{a}_f)$. This helps to suppress common background features and enhance rarer object ones.

2. We work up the spatial tree, combining votes for subpart locations into votes for their parent's locations using the mean location offsets learned for the model:

$$\text{Vote}_q(\mathbf{T}_q) \;=\; S(\sum_p \log(1 + \text{Vote}_p(\mathbf{T}_q\, \overline{\mathbf{T}}_{qp})) \tag{5}$$

Here, the sum is over subparts p for which q is the most probable parent ($\arg\max_{q'} \boldsymbol{\tau}_p(q') = q$) — again we use such hard assignments for speed. The $\log(1 + \ldots)$ nonlinearity makes it harder for high peaks in outlier subparts to dominate the valid contributions of the other subparts. S is a heuristic smoothing function, currently a Gaussian convolution. To be more rigorous, we should smooth by using $\mathbf{T}_p = \mathbf{T}_q\, \mathbf{T}_{qp}$ as the argument and integrating over samples from the uncertain transform $\mathbf{T}_{qp}$.

3. Maxima in the voting pyramid for the top-level part 0 give potential object placements $\mathbf{T}_0$.

4. For the best (or eventually, each) maximum, work back down the tree assigning part positions. If the part's voting pyramid has a good maximum near the resulting expected part position, use this value. Otherwise, assume that the part was not seen and use its default offset $\overline{\mathbf{T}}_{qp}$.

This procedure gives reasonably reliable automatic model initialization results on test sets such as the Caltech ones, even when the object has unknown position and scale and is surrounded by a moderate amount of background clutter. However it can not replace a fully-fledged object detector.

3.2 Training — Model Initialization

The above procedure assumes a fully trained model. We also automatically initialize the entire training process, so that no manual location or pre-segmentation of the objects in the training images is required. The method assumes that each training image contains an instance of the class, but the instance's position and scale can be unknown and background clutter is tolerated. It also assumes that the number of parts in each layer of the hierarchy has been fixed by hand. It works as follows:

1. Heuristically rank the training images according to their expected quality as training examples (see below), and use just the best image to estimate the initial model parameters. If this fails we could potentially use the second, third,... images, but this has not been necessary in our experiments.

2. Using K-means, cluster all of the features in the initial image into n location-appearance classes, where n is the desired number of elementary parts, and initialize an elementary part at each cluster center. The experiments below actually assume that n is the number of observed features, so that there is one elementary part per observed feature. Some of the elementary parts will correspond to background clutter. We do not currently attempt to remove these. Some feature classes may have the same appearance but different locations. This is intentional: it allows for (small numbers of) repeated features such as eyes and car wheels.

3. Work up the hierarchy, clustering the subpart centers into the desired number of parent part centres, and initializing one parent for each cluster. Cluster membership gives initial (hard) parent assignments for the subparts. The corresponding $\boldsymbol{\tau}$ matrix is initialized to a slightly softer version of these. The cluster centre gives the part location, and the median scale of the part's children gives its initial scale estimate.

The use of a single initial image in the first step could certainly be criticised, but so far our attempts to initialize from averages over several or many images have given much worse results. The critical point seems to be to provide an initial model with cleanly separated appearances and parts, from which a relatively unambiguous training phase can proceed. Averaging tends to confuse the part relationships and produce a less effective overall model.

Our method of ranking the training images by their probable quality as model initializers is as follows:

1. Use K-means to cluster the features from all (positive) training images into about 500 classes. Encode each image as a 500-D signature vector $\mathbf{S}$ (the vector of class counts).

2. Rank the feature classes by an informativeness measure (see below) and select about the 30 most informative ones. Rank the images according to the number of these classes that they contain (*i.e.* the number of classes c for which $\mathbf{S}_c \neq 0$).

For the feature informativeness ranking, we have studied two methods, one supervised, the other unsupervised.

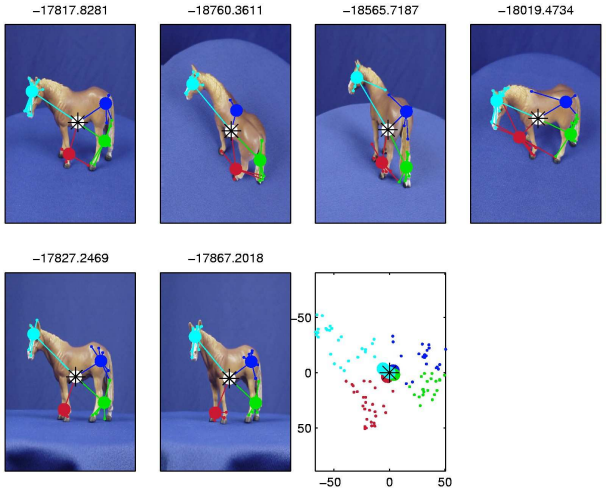


Figure 2: Model fitting on test horse toy images. The bottom right points are the model average positions of the subparts.

The supervised method requires a negative training set of non-class images as well as the positive one of class images. It trains a linear classifier to predict the image class (± 1 for positive or negative) from the binarized signature vector ($\mathbf{S} \neq 0$). The features with the highest weights are chosen as the most informative ones. Any appropriate classification method can be used: linear SVM or RVM, LASSO, *etc*.

The unsupervised method is somewhat more heuristic, but it seems to work equally well and it requires no negative images. For each feature, it counts the number of (positive) images in which it occurs *exactly once* (or alternatively, exactly 1–2 times), and chooses the features with the highest counts as the most informative. This works because it selects *distinctive features representing unique object parts*. Many object classes contain such features, whereas background features are much more variable and seldom occur exactly once per image. This method would fail for object classes dominated by repetitive texture though.

4 Experiments

In this paper we consider only a three-layer, object - part - feature class model. Figure 2 illustrates this model's ability to handle local image deformations, for which rigid matching would fail. We learned the model from 6 images of the same toy horse seen from different viewing positions, using 100 feature classes and 4 parts. The model was then instantiated on 6 test images with the method described earlier. The change of viewing angle between views is considerable, but the model still finds and locks on to the correct object parts, even if only a few points points are found on a given part.

	F.	L.	M.	A.	C.
Faces	198	12	5	1	1
Leopards	0	92	8	0	0
Motorbikes	0	6	383	10	0
Airplanes	0	4	15	351	30
Car sides	0	0	0	1	60

Table 1: The confusion matrix for part based multiclass categorization on the original Caltech 7 class dataset.

one-level model					
	Acc	Air	Anc	Ant	Bar
Accordion	18	0	0	9	0
Airplanes	0	359	6	35	0
Anchor	0	1	4	12	4
Ant	0	2	1	17	1
Barrel	0	3	1	9	10
Two-level, three part model					
Accordion	25	0	1	1	0
Airplanes	1	384	0	12	0
Anchor	0	3	6	12	0
Ant	0	4	1	18	1
Barrel	0	6	0	8	9

Table 2: Confusion matrix for best-class classifiers based on 80 feature classes, on the first few classes of the Caltech 101 class dataset.

Datasets: We used five different image classes from the "Caltech 101 Object Categories" dataset[3] [4], which contains many example images from 101 objects categories, including for example faces (435 images), leopards (200), motorbikes (800), aeroplanes (800) and side views of cars (123). These datasets have already been used by several groups [12,6,4,3]. Half of the images in each class were held out for testing.

Some examples of learned models are shown in figure 3.

To test whether the models really managed to learn the most important appearance parameters and spatial interrelationships, and whether they were sufficiently selective for a given object category, we assessed their discriminative power by fitting several class models to unseen test images, using model likelihoods as decision variables. For each class, a decision threshold was computed to minimize the average training error rate. We used 10 EM iterations during training and 5 during testing. Con-

[3] Available at http://www.vision.caltech.edu/feifeili/Datasets.htm

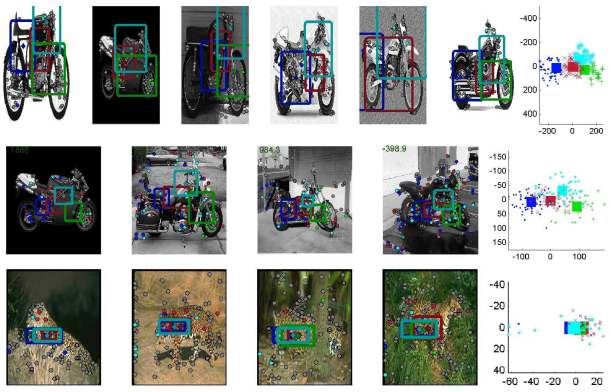


Figure 3: Examples of fits to images from the motorbikes and leopard datasets. The first line shows a close-up of the initialisation based on location/scale voting.

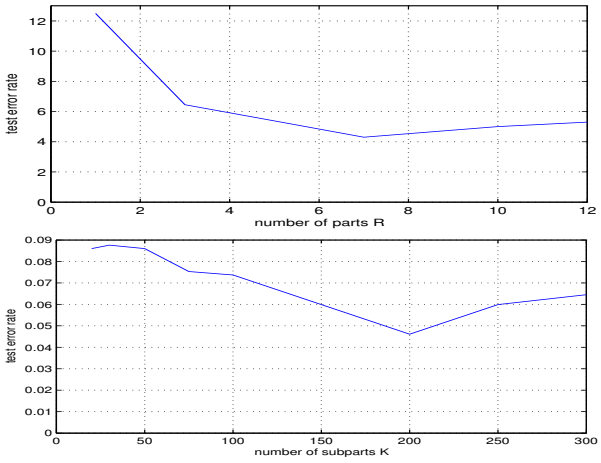


Figure 4: The test error rate of the leaves/faces classifier against the number of parts (top) and feature classes (bottom).

fusion matrices are given in table 1 for the original 7 class Caltech dataset using 200 feature classes, and in table 2 for the one-level and two level hierarchical models on the first few classes of the Caltech 101 dataset using just 80 feature classes. The number of errors depends on the class, but the results seem to be competitive with the state of the art on these datasets [2,5]. The basic rigid model is already highly discriminative for these data sets, but using a 3 part model still reduces the error rates by a factor of about two.

Figure 4 shows that the results are not too sensitive to the number of parts, although over-fitting starts to worsen the results beyond about 8–10 parts. Relatively large numbers of elementary parts are needed to get optimal results — about 200 in this case.

Soft *vs*. hard assignments: The matrix τ coding for the structure can be constrained to have only ones and zeros, so that a given part can only be generated by a single parent. To illustrate the advantages of soft parenting, for binary classification of motorbikes against background images using 40 training images, 200 feature classes and 4 parts, hard assignment based learning produces a test-set classification rate of 83%, while our standard soft assignments gave 88%. Similar results occur for other datasets and training parameters.

5 Conclusions and Future Work

We have described a multi-layered part-based generative model for category-level visual object recognition using large numbers of local features. The model managed to adapt very well to the object categories tested in supervised classification experiments. Reasons for this are its well-graded spatial flexibility, and the fact that it can efficiently incorporate a large number of interest points, each carrying a worthwhile amount of discriminant information. This lead to a full multiclass object classifier that reaches state of the art performances on benchmark databases. We also showed experimentally that so long as the model uses sufficiently many detected points, the matching of elementary parts to image features does not need to be very accurate. We showed how a simple three-layer hierarchy of object, parts and features can give satisfying visual intuition and probabilistic accuracy.

Future work: The model applies to arbitrary spatial transformations between parts and their subparts, and arbitrary numbers of layers, although here we applied it only with translation-scale transformations and 3 layers. Future work will study the advantages of more general transformations and additional layers. The main difficulty is likely to be getting a good initialization for these more complex models. Another promising direction is to learn mixtures of such generative models, for image clustering or to handle more complex classes such as 3D models viewed from all possible directions.

[1] G. Csurka, C. Dance, L. Fan, J. Williamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV'04 workshop on Statistical Learning in Computer Vision*, pages 59–74, Prague, 2004.

[2] G. Dorko and C. Schmid. Selection of scale-invariant parts for object class recognition. In *Proceedings of the 9th International Conference on Computer Vision, Nice, France*, 2003.

[3] Gy. Dorko and C. Schmid. Object class recognition using discriminative local features. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 2004. submitted.

[4] Li Fei-Fei, Rob Fergus, and Pietro Perona. A Bayesian approach to unsupervised one-shot learning of object categories. In *Proceedings of the 9th International Conference on Computer Vision, Nice, France*, pages 1134–1141, Nice, France, 2003.

[5] R. Fergus, P. Perona, and A.Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Madison, Wisconsin, USA*, 2003.

[6] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Madison, Wisconsin, USA*, 2003.

[7] William T. Freeman Kevin Murphy, Antonio Torralba. Using the forest to see the trees: A graphical model relating features, objects, and scenes. In *Neural Info. Processing Systems*, 2003.

[8] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with and implicit shape model. In *ECCV'04 workshop on Statistical Learning in Computer Vision*, pages 17–32, Prague, 2004.

[9] D. G. Lowe. Local feature view clustering for 3D object recognition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Kauai, Hawaii, USA*, pages 682–688, December 2001.

[10] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Madison, Wisconsin, USA*, June 2003.

[11] S. Ullman, E. Sali, and M. Vidal-Naquet. A fragment-based approach to object representation and classification. In *4th International Workshop on Visual Form, Capri, Italy*, May 2001.

[12] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *Proceedings of the 6th European Conference on Computer Vision, Dublin, Ireland*, pages 18–32, 2000.

References

This bibliography only includes works cited in the chapter introductions and the paper summaries. For works cited in the papers themselves, see the corresponding paper's bibliography.

[AQT02] M.-A. Ameller, L. Quan, and B. Triggs. Le calcul de pose : de nouvelles méthodes matricielles. In *Reconnaissance des Formes et Intelligence Artificielle*, Angers, January 2002.

[AT04a] A. Agarwal and B. Triggs. 3d human pose from silhouettes by relevance vector regression. In *Int. Conf. Computer Vision & Pattern Recognition*, pages II 882–888, Washington, June 2004.

[AT04b] A. Agarwal and B. Triggs. Learning to track 3d human motion from silhouettes. In *Int. Conf. Machine Learning*, pages 9–16, Banff, July 2004.

[AT04c] A. Agarwal and B. Triggs. Tracking articulated motion with piecewise learned dynamical models. In *European Conf. Computer Vision*, pages III 54–65, Prague, May 2004.

[BT04a] G. Bouchard and B. Triggs. Hierarchical part-based visual object categorization. Submitted to CVPR'05, November 2004.

[BT04b] G. Bouchard and B. Triggs. The tradeoff between generative and discriminative classifiers. In *COMPSTAT — IASC Int. Symp. Computational Statistics*, pages 721–728, Prague, August 2004.

[CR99] T. Cham and J. Rehg. A multiple hypothesis approach to figure tracking. In *Int. Conf. Computer Vision & Pattern Recognition*, pages II 239–245, 1999.

[F86] W. Förstner. A feature-based correspondence algorithm for image matching. *Int. Arch. Photogrammetry & Remote Sensing*, 26 (3/3):150–166, 1986.

[F94] W. Förstner. A framework for low-level feature extraction. In *European Conf. Computer Vision*, pages II 383–394, Stockholm, 1994.

[FE73] M.A. Fischler and R. Elschlager. The representation and matching of pictorial structures. *IEEE Trans. Computing*, C-22:67–92, 1973.

[FG87] W. Förstner and E. Gülch. A fast operator for detection and precise location of distinct points, corners and centres of circular features. In *ISPRS Intercommission Workshop*, Interlaken, June 1987.

[FH00] P. Felzenszwalb and D. Huttenlocher. Efficient matching of pictorial structures. In *Int. Conf. Computer Vision & Pattern Recognition*, pages 66–75, 2000.

[HS88] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 147–151, 1988.

[IF01] S. Ioffe and D. Forsyth. Probabilistic methods for finding people. *Int. J. Computer Vision*, 43(1):45–68, 2001.

[JBY96] S. Ju, M. Black, and Y. Yacoob. Cardboard people: A parameterized model of articulated motion. In *Int. Conf. Automatic Face & Gesture Recognition*, pages 38–44, 1996.

[KT99] F. Kahl and B. Triggs. Critical motions in euclidean structure from motion. In *Int. Conf. Computer Vision & Pattern Recognition*, Fort Collins, Colorado, 1999.

[KTÅ00] F. Kahl, B. Triggs, and K. Åström. Critical motions for auto-calibration when some intrinsic parameters can vary. *J. Mathematical Imaging & Vision*, 13(2):131–146, October 2000.

[Mor77] H.P. Moravec. Towards automatic visual obstacle avoidance. In *IJCAI*, page 584, 1977.

[RST02] R. Ronfard, C. Schmid, and B. Triggs. Learning to parse pictures of people. In *European Conf. Computer Vision*, Copenhagen, 2002.

[Smi02] C. Sminchisescu. *Estimation Algorithms for Ambiguous Visual Models — Three Dimensional Human Modeling and Motion Reconstruction in Monocular Video Sequences*. PhD thesis, INPG, July 2002.

[ST01a] C. Sminchisescu and B. Triggs. Covariance scaled sampling for monocular 3D body tracking. In *Int. Conf. Computer Vision & Pattern Recognition*, pages I 447–454, Hawaii, 2001.

[ST01b] C. Sminchisescu and B. Triggs. A robust multiple hypothesis approach to monocular human motion tracking. Research Report 4208, INRIA, June 2001.

[ST02a] C. Sminchisescu and B. Triggs. Hyperdynamic sampling for articulated estimation. In *European Conf. Computer Vision*, Copenhagen, 2002.

[ST02b] C. Sminchisescu and B. Triggs. Mapping minima and transitions of visual models. In *European Conf. Computer Vision*, Copenhagen, 2002.

[ST03a] C. Sminchisescu and B. Triggs. Estimating articulated human motion with covariance scaled sampling. *Int. J. Robotics Research*, 22(6):371–391, June 2003. Special issue on Visual Analysis of Human Movement.

[ST03b] C. Sminchisescu and B. Triggs. Kinematic jump processes for monocular 3d human tracking. In *Int. Conf. Computer Vision & Pattern Recognition*, pages I 69–76, June 2003.

[TMHF00] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment — a modern synthesis. In B. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, pages 298–372. Springer-Verlag, 2000.

[Tri00] B. Triggs. Plane + parallax, tensors and factorization. In *European Conf. Computer Vision*, pages 522–538, Dublin, 2000.

[Tri01a] B. Triggs. Empirical filter estimation for subpixel interpolation and matching. In *Int. Conf. Computer Vision*, pages II 550–557, Vancouver, 2001.

[Tri01b] B. Triggs. Joint feature distributions for image correspondence. In *Int. Conf. Computer Vision*, pages II 201–208, Vancouver, 2001.

[Tri04a] B. Triggs. Boundary conditions for Young - van Vliet recursive filtering. Under review for IEEE Trans. Image Processing, June 2004.

[Tri04b] B. Triggs. Detecting keypoints with stable position, orientation and scale under illumination changes. In *European Conf. Computer Vision*, pages IV 100–113, May 2004.

[TZS00] B. Triggs, A. Zisserman, and R. Szeliski, editors. *Vision Algorithms: Theory and Practice*, volume 1883 of *LNCS*. Springer-Verlag, 2000.

[vVYV98] L.J. van Vliet, I.T. Young, and P.W. Verbeek. Recursive Gaussian derivative filters. In *Int. Conf. Pattern Recognition*, pages 509–514, Brisbane, 1998.

[YvV95] I.T. Young and L.J. van Vliet. Recursive implementation of the Gaussian filter. *Signal Processing*, 44:139–151, 1995.

[YvVvG02] I.T. Young, L.J. van Vliet, and M. van Ginkel. Recursive Gabor filtering. *IEEE Trans. Sig. Proc.*, 50(11):2799–2805, 2002.