A New Approach to Geometric Fitting

Bill Triggs

INRIA Rhône-Alpes, 655 avenue de l'Europe, 38330 Montbonnot St. Martin, France.

Bill.Triggs@inrialpes.fr ◇ *http://www.inrialpes.fr/movi/people/Triggs*

Abstract

Geometric fitting — parameter estimation for data subject to implicit parametric constraints — is a very common sub-problem in computer vision, used for curve, surface and 3D model fitting, matching constraint estimation and 3D reconstruction under constraints. Although many algorithms exist for specific cases, the general problem is by no means 'solved' and has recently become a subject of considerable debate among researchers in statistical vision. This paper describes a new, more direct approach to geometric fitting, formulating it as the explicit recovery of a coherent, statistically optimal set of estimates of the "underlying data points" that gave rise to the observations, together with the estimated constraints which these points exactly verify. The method is implemented using an efficient constrained numerical optimization technique, and is capable of handling large problems with complex, constrained parametrizations. As examples of such problems, we consider the optimal estimation of the fundamental and essential matrices and the trifocal tensor, subject to their full sets of algebraic constraints. We also describe how our approach 'reduces' to existing geometric fitting methods like gradient-weighted orthogonal least squares, and give a novel approach to robustness based on it.

Keywords: geometric fitting, orthogonal regression, statistical estimation, constrained optimization, matching tensors.

1 Introduction

Geometric fitting — statistical estimation for data subject to implicit parametric constraints — is a very common sub-problem in computer vision. Examples include: (*i*) curve, surface and 3D model fitting (*e.g.* conics [27, 19, 7, 34]); (*ii*) the estimation of homographies, epipolar or trifocal geometry from matched image tokens [35, 29, 2, 15]; (*iii*) 3D visual reconstruction under various types of calibration constraints. Although many algorithms exist for specific cases and Kanatani [20] has attempted a synthesis, the general problem is by no means 'solved' and has recently become a subject of considerable debate among statistical vision researchers [27, 18, 20, 29, 7, 34, 28]. Discussion has centred on three issues: (*i*) **statistical theory:** bias, error measures, decision rules, the nature of optimality; (*ii*) **robustification**; (*iii*) efficient **numerical methods** for frequent, large or complex problems.

Although theoretical, the research is being driven by practical problems: (*i*) the need to improve the precision and robustness of highly automated vision systems, *e.g.* for 3D model building [9, 21, 22, 6, 2]; (*ii*) the need to handle algebraically complex problems arising from multi-image geometry, with redundant constraints, gauge freedoms, and/or constrained parametrizations — *e.g.* matching tensor estimation, self calibration, reconstruction under constraints.

This paper describes a new approach to geometric fitting, which directly applies constrained numerical optimization to the "natural" underlying problem, rather than trying to "reduce" it to a "simpler" one. The result is easier to use and more efficient than some of the *ad hoc* algorithms that have been proposed. It also allows guarantees about things like rates of convergence, and automatically provides useful supplementary information — covariances and point position estimates — that would often have been needed in any case. We will also discuss how the method can be 'reduced' to existing ones, notably gradient-weighted least squares [27], a fundamental matrix method considered in [35], and Kanatani's 'optimal parametric fitting' [20].

As test cases, we give algorithms for the precise estimation of the fundamental matrix, the essential matrix, and the trifocal tensor from point matches, which exactly take into account all of the nonlinear constraints involved. (As far as I know, this is the first such algorithm for the trifocal tensor, if direct calculation from an intermediate projective reconstruction is excluded).

Sections 2 and 3 motivate the method by considering some complex real-world fitting problems, and describe the abstract framework that we will adopt. Sections 4 and 7 discuss the 'full' and 'reduced' methods, and relate these to current approaches. Section 6 considers covariance estimation. Sections 5 and 10 describe the numerical implementation and apply it to matching tensor estimation. Section 11 briefly describes some preliminary experimental results, and we finish with a conclusion.

Submitted to ICCV'98. This work was supported by INRIA Rhône-Alpes and Esprit LTR project CUMULI. I would like to thank L. Quan, P. Torr and P. Sturm for discussions and G. Csurka, K. Daumueller and R. Horaud for the calibration data.

Notation is introduced as needed, but briefly: $\mathbf{u}$ stands for parameters, $\mathbf{x}$ for true underlying data points, $\mathbf{y}$ for observations of these, $\mathbf{c}(\mathbf{x},\mathbf{u})$ for the constraints linking $\mathbf{x}$ to $\mathbf{u}$, and $\mathbf{k}(\mathbf{u})$ for constraints on $\mathbf{u}$ itself. Bold subscripts on functions denote derivatives, *e.g.* $\mathbf{c_x}$ and $\mathbf{c_u}$ are the $\dim(\mathbf{c}) \times \dim(\mathbf{x})$ and $\dim(\mathbf{c}) \times \dim(\mathbf{u})$ Jacobeans $\partial\mathbf{c}/\partial\mathbf{x}$ and $\partial\mathbf{c}/\partial\mathbf{u}$. Bold subscripts on non-functions are just labels, *e.g.* $\mathbf{\Lambda_x}$. $\mathbf{\Sigma}$ and $\mathbf{\Lambda}$ denote covariance and information (inverse covariance) matrices, *e.g.* $\mathbf{\Lambda_x} \sim \mathbf{\Sigma_x^{-1}}$. Hat $\hat{\mathbf{u}}$ denotes fitted estimates of true underlying values $\mathbf{u}^{\text{true}}$. On 3-vectors, $[\cdot]_\times$ denotes the cross product matrix $[\mathbf{a}]_\times \mathbf{b} = \mathbf{a} \wedge \mathbf{b}$.

2 Some Typical Fitting Problems

To motivate our method and relate it to existing ones, we sketch several common geometric fitting problems and the difficulties they create. **'Geometric'** means that: (*i*) there are constraints expressed as *implicit* functions $\mathbf{c}(\mathbf{x}_i,\mathbf{u}) = \mathbf{0}$ of the underlying data points $\mathbf{x}_i$ and parameters $\mathbf{u}$; (*ii*) all coordinates of the data or parameters are on an equal footing, and *all* are uncertain. For example, we can not (or choose not to) take some of the variables as independent, precisely known 'explananatory variables' and describe the others as functions (regressions) of these.

• **Conic fitting:** This well-studied problem [27, 19, 7, 34] is a prototype for other types of implicit curve and surface fitting, and also (modulo the difficult *data association* problem) for fitting object-level "CAD" models to 3D or image data. Projectively, conics can be parametrized by the 6 independent coordinates $\mathbf{u}$ of a 3×3 homogeneous symmetric matrix $\mathbf{A}$ and written implicitly as $\mathbf{c}(\mathbf{x}_i,\mathbf{u}) \equiv \mathbf{x}_i^\top \mathbf{A} \mathbf{x}_i = 0$. Owing to measurement error, these equations will not be exactly satisfied for observations $\mathbf{y}_i$. Fitting necessarily involves a trade-off between the different error terms.

A key insight is that it is *not* sufficient just to take arbitrary algebraic formulae $\mathbf{c}(\mathbf{y}_i,\mathbf{u})$ and minimize the **algebraic distance** $\sum_i \|\mathbf{c}(\mathbf{y}_i,\mathbf{u})\|^2$, as the resulting implicit least squares trade-off can lead to very significant bias [19, 7, 34]. Before attempting any such "algebraic" minimization, it is *essential* to **balance** or **standardize** the relative scales of both the coordinates and the different constraint equations. The Jacobean $\mathbf{c_x} \equiv \nabla_\mathbf{x}\mathbf{c}$ should have components of $\mathcal{O}(1)$. For example, pixel coordinates should be standardized to $\mathcal{O}(1)$: mixing $\mathcal{O}(512)$ pixel coordinates with $\mathcal{O}(1)$ homogenizers can give different terms in $\mathbf{c}$ very different scales, and hence very different weights in the minimization.

But even *with* standardization, algebraic distance is just a heuristic. A statistically more accurate error model can be obtained by locally linearizing the constraints (and hence the fitted curve or surface) and minimizing the Mahalanobis-orthogonal distance of the observations from the curve or surface. To first order, this is equivalent to minimizing the constraint-covariance-weighted error measure $\sum_i \mathbf{c}(\mathbf{y}_i,\mathbf{u})^\top \mathbf{\Sigma}_{\mathbf{c}i}^{-1} \mathbf{c}(\mathbf{y}_i,\mathbf{u})$, where $\mathbf{\Sigma}_{\mathbf{c}i} \equiv \mathbf{c_x}(\mathbf{y}_i,\mathbf{u})^\top \mathbf{\Sigma}_{\mathbf{x}i} \mathbf{c_x}(\mathbf{y}_i,\mathbf{u})$ is the predicted covariance of the constraint functions owing to $\mathbf{y}_i$ uncertainty, for fixed $\mathbf{u}$. We will rederive this below, but it is a standard result in the statistical theory of **weighted orthogonal regression** [10, 4, 3], advocated in vision contexts by (among others) [27, 20]. For scalar constraints, this reduces to **gradient weighted least squares** $\min_\mathbf{u} \sum_i |\mathbf{c}(\mathbf{y}_i,\mathbf{x})|^2/(\mathbf{c_x}^\top \mathbf{\Sigma}_{\mathbf{x}i} \mathbf{c_x})$.

• **Matching tensors:** These are the objects that generate the algebraic constraints between corresponding tokens in different images. For two images we have the 3×3 **fundamental matrix** $\mathbf{F}$ and its calibrated cousin the **essential matrix** $\mathbf{E}$ [23, 5], while for three images there are three distinct $3 \times 3 \times 3$ **trifocal tensors** $\mathbf{G}$ [24, 14]. Besides their use in image matching, matching tensors are important because they characterize the camera geometry and generate an implicit 3D scene reconstruction [26, 32]. However they are an over-parametrization. For m images the camera geometry has $6m - 7$ parameters for calibrated cameras (*i.e.* $5, 11$ for $m = 2, 3$), or $11m - 15$ for uncalibrated ones ($7, 18$ for $m = 2, 3$). In contrast $\mathbf{F}$, $\mathbf{E}$ and $\mathbf{G}$ have respectively 8, 8 and 26 components (up to scale), and hence must be subject to 1, 3, and 8 supplementary constraints. For $\mathbf{F}$ the constraint is $\det(\mathbf{F}) = 0$, while for $\mathbf{E}$ the constraints can be written in several forms [5], in particular as a subset of 3 of the 9 cubic Demazure constraints $(\mathbf{E}\mathbf{E}^\top - \frac{1}{2}\,\text{trace}(\mathbf{E}\mathbf{E}^\top) \cdot \mathbf{I})\ \mathbf{E} = \mathbf{0}$. For the trifocal tensor $\mathbf{G}$ there are 10 cubic 'determinantal' constraints of which only 8 are linearly independent for any given tensor (details below).

To estimate the matching tensors accurately, it is essential to take these nonlinear constraints into account. For example, the "linear" (algebraic distance based) methods for estimating $\mathbf{F}$ from at least 8 matched point pairs [23, 5] or $\mathbf{G}$ from 7 matched point triplets [15] are reputed to lose both accuracy and stability because they ignore the constraints (not to mention their suboptimal error metrics).

There are basically two ways to incorporate the constraints. One is to develop methods capable of fitting models whose parameters $\mathbf{u}$ are subject to arbitrary constraints $\mathbf{k}(\mathbf{u}) = \mathbf{0}$. This is the route we will take below. The other is to attempt to remove the redundancy by finding a **minimal parametrization** that automatically satisfies the constraints. For example Zhang [35] gives several such parametrizations for the fundamental matrix $\mathbf{F}$. In theory this is always possible, but in practice it is often infeasible or undesirable. For one thing, it essentially amounts to algebraic elimination of variables using the constraints. This tends to significantly increase the degree and complexity of the equations involved, leading to complicated code and much poorer numerical conditioning. Secondly, the resulting formulae are usually only valid on a limited 'coordinate patch' (range of the remaining variables). Often they have singularities on the patch boundaries and numerical ill-behaviour extending well into the interior. Restricting attention to a single patch (assumptions like 'reconstructed points will not be near infinity' or 'rotations will be much less than 90°') may exclude the true solution, and often leads to sluggish or unreliable convergence. To counter this, several parametrizations must be implemented, along with code to detect problems and switch between patches when necessary. This can easily become unmanageable: we take the view that it is simpler and often faster and more reliable to use a redundant parametrization and a formalism designed to handle constraints, than to try to eliminate them.

For the trifocal tensor there is a further complication. Not only is there no simple minimal parametrization (that I am aware of), but the constraints themselves are redundant in that it is not clear how to systematically choose 8 linearly independent representatives among the 10 available cubic constraints. So we must deal with a redundant parametrization subject to redundant constraints. In fact, even the trifocal *point* matching constraints are redundant in that they give $3 \times 3 = 9$ tensorial equations of which only 3 are linearly independent, although here it is at least possible to systematically choose 3 independent representatives (see below).

• **Reconstruction under constraints.** Another common problem is **gauge freedom**: combinations of parameters which have no effect at all on the overall fitting error. Examples include the choice of coordinate frame for 3D reconstruction, and the choice of scale factors for homogeneous projective point vectors, projections, *etc.*. ('Gauge' means 'coordinate system'). In theory these redundant degrees of freedom can always be eliminated, either by a clever choice of parametrization or by adding explicit **gauge fixing** constraints. However in practice it is often advisable to leave non-trivial gauge freedoms such as coordinate frames free, as this can simplify the formulation and significantly improve numerical conditioning and reliability by giving the algorithm "more room to manoeuvre". The idea is that a redundant method does not have to "crawl along the constraint surface" to find the solution, it has the freedom to move through some "more linear" embedding space, temporarily violating the constraints in return for a more direct path to the goal. (It is important to realize that since minimal parametrizations are often local and highly nonlinear, the "algebraic" distance they have to "crawl" might be large or infinite, while the "linear" distance remains small).

For example photogrammetrists often use 'free' (Euclidean frame) bundle adjustment [1] because this has proved more reliable and faster to converge in practice. Projective reconstruction takes this one step further, gaining simplicity by licensing even projective deformations. The point is that nonlinear fitting is hard enough already, without adding complicated supplementary constraints just to fix a gauge that we do not really care about in the first place.

Allowing unconstrained gauge freedoms does also have disadvantages: (*i*) there are more parameters to deal with; (*ii*) the freedom leads to rank deficient matrices, so it is essential to use stable numerical methods — 'orthogonal' methods like QR, SVD, or pseudo-inverses [20], or some form of regularization like Levenberg-Marquardt — to ensure that the resulting 'randomly chosen' coordinate frames are "reasonable" (nonsingular, well centred on the data, *etc.*). The net result is that although 'free' methods are often more reliable and may converge in fewer iterations than minimal ones, each iteration is slower so the overall speed difference is unclear.

In summary, for precise results it is essential to use an accurate statistical error model and to take any existing constraints into account. Real fitting problems often have various types of **redundancy**: gauge freedoms, constrained parametrizations, and redundant constraints. Minimal parametrizations are always possible in theory, but often inadvisable in practice owing to their complexity and/or poor numerical conditioning.

For serious applications it is seldom enough to estimate just the model parameters $\hat{\mathbf{u}}$. For one thing error estimates (*e.g.* **covariances**) are often needed. In theory these can be obtained from the Fisher information (error surface curvature) at the estimated solution using standard 'covariance propagation' techniques [13]. However some sophistication is needed. Large problems often have a sparse information but a dense covariance, which it may be impractical to store or manipulate. Also, with redundant parametrizations, directions of gauge freedom have infinite covariance and zero information, while constraint-violating directions have zero covariance and infinite information. Some care is needed to work numerically with such singular matrices.

Secondly, **optimal consistent estimates** $\hat{\mathbf{x}}_i$ of the 'true underlying data points' are often useful, *i.e.* ones that are statistically optimal among all those exactly compatible with the estimated model and constraints. For example for the epipolar constraint, optimal consistent estimates are maximum likelihood point pairs where each point lies exactly on its partners estimated epipolar line. These are 'pre-triangulated' in the sense that their matching-constraint-violating uncertainty has already been optimally resolved: the corresponding 3D point can be reconstructed exactly without recourse to further 'triangulation' [16]. Similarly, in 3D modelling, it is often convenient to 'clamp' reconstructed points to the estimated model surface, for example to allow the accumulation of surface detail or as an aid to further geometric processing. We will see below that such 'optimal projection onto the constraints' is actually the key to the whole fitting process: all we have to do is make it explicit.

Finally, for real applications it is essential to provide some form of **robustness** to outliers. Gross outliers are endemic to the visual correspondence problem. The most effective remedy seems to be some form of **consensus search**, identifying “good” matches by their mutual compatibility with a single underlying (set of) model(s). Several approaches have been developed, notably parameter-space voting (Hough transform) and random sampling, but consensus remains a difficult global combinatorial optimization problem, especially in high-dimensions.

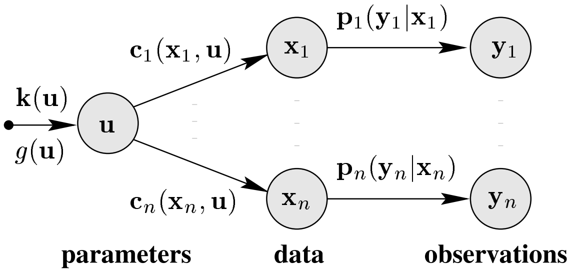
A more subtle but perhaps even more ubiquitous problem is *near* outliers. These are caused by: (*i*) features that have been correctly matched but poorly localized (owing to lighting variations, shadowing, partial occlusion, large changes in viewpoint, oversmoothing, vagueness of the underlying feature, *etc.*); and (*ii*) mismatches that happen to nearly satisfy the constraints (*e.g.* in cluttered scenes, or when there is repetitive texture). By definition near outliers are not too far from consistency so they cause no gross errors, but in precise work they can significantly degrade the accuracy of the final parameter estimate, especially if acceptance thresholds have been set generously in an attempt to catch as many matches as possible.

Handling near outliers is difficult as the outlier decision requires a fairly accurate estimate of the inlier error distribution. M-estimators potentially give the most accurate results, but estimating the required scale parameter σ is problematic. Rank based estimators like least median squares have limited scale invariance (they depend on the acceptance thresholds for the initial set of *possible* inliers, which must reduce outliers to $<50\%$ *without* deleting a significant proportion of true examples) but they are significantly more variable and they do not so directly yield parameter covariances or consistent point estimates.

NB: There seems to be some confusion over robust error measures in the literature, with several authors (*e.g.* [35, 29]) preferring least median squares over M-estimators on the grounds that it “has a higher breakdown point”. For parameter value (not covariance) estimation with roughly known covariance scale, this is incorrect. What *is* true is that the *search method* of random sampling has a (much!) higher breakdown point than the *search method* of gradient descent starting from a least squares solution — but either of those search methods could be applied to either estimator. In fact M-estimators are probably more robust in practice provided their scale is set correctly, as they make an explicit decision about what is plausible data rather than blindly “accepting 50%”.

3 Problem Formulation

Our goal is to fit an implicit parametric model to noisy geometric measurements. Statisticians call this **weighted orthogonal regression** [4, 3]. We will formalize it with the following abstract model, which is sufficiently general to cover most practical vision problems while still allowing efficient implementation. This is essentially just the standard orthogonal regression model, also used (with adaptations) by *e.g.* [27, 19, 20], but it pays to make the logical status of the various elements clear at the outset.



We make a series of uncertain **observations** $\mathbf{y}_i$, $i = 1, \ldots, n$. Each observation relates to some specific underlying geometric entity $\mathbf{x}_i$, which (for want of a better name) we call the **data**. The data are fixed but unknown ‘ideal’ entities, logically distinct from the observations but in some sense the sources of them. In curve or surface fitting, they are the unknown exact positions of the ‘ideal’ underlying points which fall exactly on the true underlying curve or surface, while the observations are noisy measurements of these. In 3D modelling, the data might be ideal points and lines lying exactly on various facets of the true (but ideal) underlying 3D object. For epipolar or trifocal geometry they are pairs or triplets containing the exact image positions of the true underlying 3D points and lines. The data and observations might be in different spaces and contain a mixture of different types (points, lines, conics, *etc.*), but we will call them all ‘points’ for convenience.

Whether the observations are direct measurements of the unknown data points or (for example) measured image projections of 3D points $\mathbf{x}_i$, we suppose that they are linked to the data by known probabilistic uncertainty models $\mathbf{p}_i(\mathbf{y}_i|\mathbf{x}_i)$. For fitting, we convert these into measures of statistical deviation $e_i(\mathbf{x}_i)$, *e.g.* posterior log likelihood $e_i(\mathbf{x}_i|\mathbf{y}_i, \text{priors}) = -\log \mathbf{p}(\mathbf{y}_i|\mathbf{x}_i) - \log \mathbf{p}(\mathbf{x}_i|\text{priors})$. The observations and prior information on $\mathbf{x}_i$ enter the fitting process *only* through these penalty functions e_i, but we will not explicitly display them among e_i’s arguments as they are constant during any particular fitting problem.

The data points $\mathbf{x}_i$ are not arbitrary, but are constrained to lie exactly on some sort of ‘ideal’ geometric model — the underlying curve, surface, 3D object model, epipolar geometry, *etc.*. The exact model is unknown, but is supposed to belong to some parametric family governed by parameters

$\mathbf{u}$. For fixed $\mathbf{u}$ and each data point $\mathbf{x}_i$, the model is specified by a vector of *implicit constraints* $\mathbf{c}_i(\mathbf{x}_i, \mathbf{u}) = \mathbf{0}$ which $\mathbf{x}_i$ must satisfy exactly. We will suppose that these are the *only* constraints on the $\mathbf{x}_i$, which are otherwise arbitrary and unknown (modulo any prior information incorporated into the $e_i()$, as above).

The parameters $\mathbf{u}$ may themselves be constrained by consistency relations $\mathbf{k}(\mathbf{u}) = \mathbf{0}$. For example, the components of the fundamental matrix satisfy the constraint $\det(\mathbf{F}) = 0$. We will also allow a bias or error penalty $g(\mathbf{u})$ on $\mathbf{u}$, perhaps derived from a prior $\mathbf{p}(\mathbf{u}|\text{priors})$ summarizing independent previous measurements of $\mathbf{u}$. We assume that g does not depend on the current data $\mathbf{x}_i$, which are coupled to $\mathbf{u}$ *only* through the constraints $\mathbf{c}_i$, not through either g or e_i.

Our goal is to find a statistically optimal estimate $\hat{\mathbf{u}}$ of the parameters $\mathbf{u}^{\text{true}}$ describing the true underlying model. We are mainly interested in point (mode based) estimators like least squares (LS), maximum likelihood (ML), or maximum a posteriori (MAP), so for the moment we assume that the optimality criterion is the minimization of the combined error metric $g(\mathbf{u}) + \sum_i e_i(\mathbf{x}_i)$. As discussed above, it may also be useful to find optimal consistent (with $\hat{\mathbf{u}}$) estimates $\hat{\mathbf{x}}_i$ of the true data $\mathbf{x}_i^{\text{true}}$. But whether these are needed or not, we will see that recovering them is actually the essence of fitting and must be done implicitly even if it is not made explicit.

We will assume that the constraint functions $\mathbf{k}(\mathbf{u})$ and $\mathbf{c}_i(\mathbf{x}_i, \mathbf{u})$ and the error functions $g(\mathbf{u})$ and $e_i(\mathbf{x}_i)$ are locally smooth and bounded near $\mathbf{u}^{\text{true}}$ and $\mathbf{x}_i^{\text{true}}$, and that we have initial estimates sufficiently good to allow a local optimization method to converge to the globally optimal solution given the observations. Often the data $\mathbf{x}_i$ can be initialized from the observations $\mathbf{y}_i$, while initialization of $\mathbf{u}$ is significantly harder. Some sort of consensus search may be necessary (*e.g.* Hough voting or random sampling), but if so we suppose that this has already been done.

To simplify notation and provide insight into the nature of constrained fitting, it is useful to gather the components of the all data points into a single big **joint data vector** $\mathbf{x} \equiv (\mathbf{x}_1, \ldots, \mathbf{x}_n)$. Similarly, the constraints $\mathbf{c}_i$ are assembled into a big **joint constraint vector** $\mathbf{c}(\mathbf{x}, \mathbf{u}) \equiv (\mathbf{c}_1(\mathbf{x}_1, \mathbf{u}), \ldots, \mathbf{c}_n(\mathbf{x}_n, \mathbf{u}))$ and the error functions are summed into a **joint error function** $e(\mathbf{x}) \equiv \sum_i e_i(\mathbf{x}_i|\mathbf{y}_i, \text{priors})$.

For fixed $\mathbf{u}$, the **feasible subspace** $\mathcal{S}_{\mathbf{u}}$ is the set of all $\mathbf{x}$ consistent with the constraints $\mathbf{c}(\mathbf{x}, \mathbf{u}) = \mathbf{0}$. This is just the Cartesian product $\mathcal{S}^1_{\mathbf{u}} \times \ldots \times \mathcal{S}^n_{\mathbf{u}}$, where $\mathcal{S}^i_{\mathbf{u}}$ is the subspace of $\mathbf{x}_i$ compatible with $\mathbf{c}_i(\mathbf{x}_i, \mathbf{u}) = \mathbf{0}$. As $\mathbf{u}$ varies (subject to $\mathbf{k}(\mathbf{u}) = \mathbf{0}$), the subspaces $\mathcal{S}_{\mathbf{u}}$ sweep out the **total feasible subspace** $\mathcal{S} \equiv \bigcup_{\mathbf{u}} \mathcal{S}_{\mathbf{u}}$ (see fig. 1). It is this space of all potentially feasible joint data points that we have to search to find the optimal estimate $\hat{\mathbf{x}}$ and its associated $\hat{\mathbf{u}}$.

It is important to realize that even if there are many $\mathbf{u}$

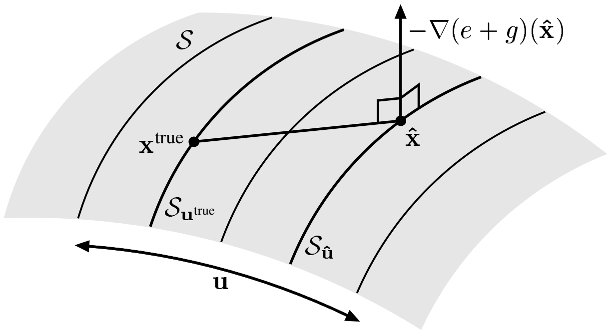


Figure 1: Geometric fitting amounts to minimizing the error over the total feasible subspace $\mathcal{S}$ in joint data space, and reading off the corresponding $\mathbf{u}$.

values consistent with any given $\mathbf{x}_i$, $\mathcal{S}_{\mathbf{u}}$ and $\mathcal{S}$ are usually *proper* subspaces of the joint data space. Generically, $\mathcal{S}$ locally has codimension $\dim(\mathbf{c}) - \dim(\mathbf{u}) + \dim(\mathbf{k})$ and is regularly foliated by *disjoint* subspaces $\mathcal{S}_{\mathbf{u}}$ of codimension $\dim(\mathbf{c})$. This occurs quite naturally: $\mathcal{S}$ is proper whenever there is too much data to interpolate exactly given this many parameters, and the $\mathcal{S}_{\mathbf{u}}$ are generically disjoint whenever each parameter is generically identifiable given the data (*i.e.* locally in $(\mathbf{x}, \mathbf{u})$, for each perturbation $\mathbf{u} \to \mathbf{u} + \boldsymbol{\delta}\mathbf{u}$ there is at least one constraint $\mathbf{c}_i$ for which that perturbation makes a difference). However, note that if there is a gauge freedom in $\mathbf{u}$, gauge-equivalent $\mathcal{S}_{\mathbf{u}}$ will coincide exactly and $\mathcal{S}$ will have fewer dimensions. (In this case, *any* gauge-equivalent $\mathbf{u}$ is equally good as a solution).

A key insight is that the parameters $\mathbf{u}$ — the subdivision of $\mathcal{S}$ into $\bigcup \mathcal{S}_{\mathbf{u}}$ — are in a sense *irrelevant* to the fitting problem. The error metric e is defined in terms of $\mathbf{x}$ alone. The $\mathbf{u}$-penalty $g(\mathbf{u})$ (if any) can in theory be re-expressed as an implicit function of $\mathbf{x}$, which happens to be constant on each $\mathcal{S}_{\mathbf{u}}$. So maximizing the fit just amounts to minimizing the combined error $(e + g)(\mathbf{x})$ over the *total* feasible subspace $\mathcal{S}$ (fig. 1). Once this has been done, the corresponding $\mathbf{u}$ can simply be read off. The optimality condition is simply that $\nabla_{\mathbf{x}}(e+g)$ be orthogonal to $\mathcal{S}$, both in the $\mathcal{S}_{\mathbf{u}}$ directions (so that the estimates $\hat{\mathbf{x}}_i$ are locally optimal *given* $\mathbf{u}$) and transversally in the $\frac{\partial}{\partial \mathbf{u}}$ directions (so that the estimated $\hat{\mathbf{u}}$ or $\mathcal{S}_{\hat{\mathbf{u}}}$ is locally optimal).

Note that if the $\mathbf{y}_i$ are direct measurements of the $\mathbf{x}_i^{\text{true}}$ with Gaussian errors and the prior $g(\mathbf{u})$ is negligible, the error gradient at any point $\mathbf{x}$ is Mahalanobis conjugate the direction of the joint observation $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$: $-\mathbf{e}_{\mathbf{x}} = \boldsymbol{\Sigma}_{\mathbf{x}}^{-1}(\mathbf{y} - \mathbf{x})$. Hence, parametric fitting is equivalent to Mahalanobis-orthogonal projection onto $\mathcal{S}$: $\hat{\mathbf{x}}$ lies at the foot of the Mahalanobis normal to $\mathbf{y}$ in $\mathcal{S}$.

In summary, we can reformulate constrained geometric fitting as the following abstract optimization problem:

Abstract Geometric Fitting Problem
Given:
– an unknown data vector $\mathbf{x}$
– an unknown parameter vector $\mathbf{u}$
– error functions $e(\mathbf{x})$ and $g(\mathbf{u})$
– vectors of constraint functions $\mathbf{c}(\mathbf{x},\mathbf{u})$ and $\mathbf{k}(\mathbf{u})$
Find $(\hat{\mathbf{x}},\hat{\mathbf{u}})$ minimizing $e(\mathbf{x})+g(\mathbf{u})$ subject to the constraints $\mathbf{c}(\mathbf{x},\mathbf{u})=\mathbf{0}$ and $\mathbf{k}(\mathbf{u})=\mathbf{0}$.

Of course, this is just an instance of the still more abstract problem of minimizing a function of the combined parameters $(\mathbf{x},\mathbf{u})$ subject to the combined constraints $(\mathbf{c},\mathbf{k})$, but to exploit the special structure of the problem we will work with the above expanded form.

4 A Fitting Algorithm

The above framework implicitly underlies most current statistical approaches to constrained fitting, but it is usually hidden by "reducing" the problem to one that explicitly involves only the parameters $\mathbf{u}$. We will consider reduction below. In this section we derive an efficient algorithm that directly solves the "full" fitting problem as it is given above.

Advantages of the full approach include: (*i*) simplicity and directness; (*ii*) the fact that it exactly solves the underlying estimation problem without linearization assumptions; and (*iii*) the fact that it automatically gives optimal, exactly consistent estimates $\hat{\mathbf{x}}_i$ of the underlying data points $\mathbf{x}_i^{\text{true}}$. On the other hand, for simple problems the reduced methods are potentially a little faster, especially if they are hard-coded and $\hat{\mathbf{x}}_i$ estimates or covariances are not needed. However for complex problems with multiple constraints $\mathbf{c}_i$, the most practical way to implement the reduced methods is as the first pass of the full one, so the speed difference turns out to be quite small and the full method is recommended.

Our approach is based on the classical constrained optimization technique of **sequential quadratic programming (SQP)** [11, 8]. This is a Newton-like method that requires second derivatives and provides quadratic asymptotic convergence for smooth non-linear cost functions under smooth non-linear constraints. The basic idea is to iterate, at each step approximating the cost and the constraints by their second order Taylor series and exactly solving the resulting subproblem. Suppose we want to extremize a scalar cost function $f(\mathbf{x})$ subject to constraints $\mathbf{c}(\mathbf{x})=\mathbf{0}$, over some variables $\mathbf{x}$. Lagrange multipliers $\mathbf{z}$ give an implicit solution[1]:

$$f_{\mathbf{x}}+\mathbf{z}^\top\mathbf{c}_{\mathbf{x}}=\mathbf{0} \qquad \text{with} \qquad \mathbf{c}(\mathbf{x})=\mathbf{0}$$

Take second order Taylor approximations based at the current estimate $\mathbf{x}$, and look for an update $(\boldsymbol{\delta}\mathbf{x},\boldsymbol{\delta}\mathbf{z})$ that satisfies the Lagrange equations to first order. The result is:

$$\begin{pmatrix} f_{\mathbf{xx}}+\mathbf{z}^\top\mathbf{c}_{\mathbf{xx}} & \mathbf{c}_{\mathbf{x}}^\top \\ \mathbf{c}_{\mathbf{x}} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\boldsymbol{\delta}\mathbf{x} \\ \mathbf{z}+\boldsymbol{\delta}\mathbf{z}\end{pmatrix} = -\begin{pmatrix} f_{\mathbf{x}} \\ \mathbf{c}\end{pmatrix}$$

Solve, update $(\mathbf{x},\mathbf{z})$, re-estimate derivatives, and iterate to convergence.

The coefficient matrix is typically (but not invariably) nonsingular provided $f_{\mathbf{xx}}$ is positive and $\mathbf{c}_{\mathbf{x}}$ has full row rank (*i.e.* the constraints are locally linearly independent). However it is never positive definite owing to the $\mathbf{0}$ in the lower right corner, so care is needed when selecting numerical methods. If $\mathbf{c}_{\mathbf{x}}$ is rank deficient there may be no locally feasible solution ($\mathbf{c}\neq\mathbf{0}$). But if there is, a least-squares (pseudo-inverse) update is adequate for the above iteration.

At the risk of a reduced rate of asymptotic convergence, the matrix on the left can be modified without changing the solution. (As always with Newton-like methods, it is the vanishing of the right hand side that determines the solution). The Lagrange term $\mathbf{z}^\top\mathbf{c}_{\mathbf{xx}}$ corrects for the curvature of the constraint surfaces, and can often be dropped if this is expected to be much smaller than the curvature $f_{\mathbf{xx}}$ of the cost isosurfaces (in our case, the point error ellipsoids). Our implementation below does this by default, as it significantly reduces the cost per iteration. A Levenberg-Marquardt-like regularizer $\lambda\mathbf{I}$ can also be added, "small" terms dropped, *etc.*. Modifying the lower right hand $\mathbf{0}$ requires care: when factorized, this block naturally becomes *negative* definite, so any added Levenberg-Marquardt-like term must also be negative (otherwise it will *de*stabilize the system). In any case, the constraints will only be satisfied to first order at the next step if the $\mathbf{c}_{\mathbf{x}}$ and $\mathbf{0}$ blocks are preserved.

In a practical implementation of SQP, it is also necessary to monitor the above step prediction and reduce it if necessary, to guard against convergence to non-minimal stationary points and uncontrolled steps outside the domain of validity of the Taylor approximation. This is recommended (but perhaps not absolutely indispensable) even for the relatively benign (by numerical analysts standards) problems considered in this paper. Inequality constraints can also be incorporated, although this significantly complicates the algorithm [8, 11].

'Full' geometric fitting method: Now we apply SQP to our abstract geometric fitting problem. See the inset above

[1] We will use subscripts to denote derivatives, *e.g.* $\mathbf{c}_{\mathbf{x}}$ is the $\dim(\mathbf{c})\times\dim(\mathbf{x})$ constraint Jacobean $[\partial\mathbf{c}_i/\partial\mathbf{x}_j]$ and $f_{\mathbf{xx}}$ is the Hessian $\left[\partial^2 f/\partial\mathbf{x}_i\partial\mathbf{x}_j\right]$.

$$\boldsymbol{\Lambda} \equiv \mathsf{LDL}^\top = \begin{pmatrix} \boldsymbol{\Lambda}_\mathbf{x} & \mathbf{c}_\mathbf{x}^\top & & \\ \mathbf{c}_\mathbf{x} & \mathbf{0} & \mathbf{c}_\mathbf{u} & \\ & \mathbf{c}_\mathbf{u}^\top & \boldsymbol{\Lambda}'_\mathbf{u} & \mathbf{k}_\mathbf{u}^\top \\ & & \mathbf{k}_\mathbf{u} & \mathbf{0} \end{pmatrix} \qquad \boldsymbol{\Sigma} \equiv \boldsymbol{\Lambda}^{-1} = \mathsf{L}^{-\top}\mathsf{D}^{-1}\mathsf{L}^{-1} = \begin{pmatrix} \boldsymbol{\Sigma}_\mathbf{x} & & \text{symmetric} & \\ \boldsymbol{\Lambda}_\mathbf{c}\mathbf{T}_\mathbf{cx} & -\boldsymbol{\Lambda}_\mathbf{c} & & \\ -\boldsymbol{\Sigma}_\mathbf{u}\mathbf{T}_\mathbf{ux} & \boldsymbol{\Sigma}_\mathbf{u}\mathbf{T}_\mathbf{uc} & \boldsymbol{\Sigma}_\mathbf{u} & \\ -\boldsymbol{\Lambda}_\mathbf{k}\mathbf{T}_\mathbf{kx} & \boldsymbol{\Lambda}_\mathbf{k}\mathbf{T}_\mathbf{kc} & \boldsymbol{\Lambda}_\mathbf{k}\mathbf{T}_\mathbf{ku} & -\boldsymbol{\Lambda}_\mathbf{k} \end{pmatrix}$$

$$\mathsf{D} \equiv \begin{pmatrix} \boldsymbol{\Lambda}_\mathbf{x} & & & \\ & -\boldsymbol{\Sigma}_\mathbf{c} & & \\ & & \boldsymbol{\Lambda}_\mathbf{u} & \\ & & & -\boldsymbol{\Sigma}_\mathbf{k} \end{pmatrix} \qquad \mathsf{L} \equiv \begin{pmatrix} \mathbf{I}_\mathbf{x} & & & \\ \mathbf{T}_\mathbf{cx} & \mathbf{I}_\mathbf{c} & & \\ & -\mathbf{T}_\mathbf{uc} & \mathbf{I}_\mathbf{u} & \\ & & \mathbf{T}_\mathbf{ku} & \mathbf{I}_\mathbf{k} \end{pmatrix} \qquad \mathsf{L}^{-1} = \begin{pmatrix} \mathbf{I}_\mathbf{x} & & & \\ -\mathbf{T}_\mathbf{cx} & \mathbf{I}_\mathbf{c} & & \\ -\mathbf{T}_\mathbf{ux} & \mathbf{T}_\mathbf{uc} & \mathbf{I}_\mathbf{u} & \\ \mathbf{T}_\mathbf{kx} & -\mathbf{T}_\mathbf{kc} & -\mathbf{T}_\mathbf{ku} & \mathbf{I}_\mathbf{k} \end{pmatrix}$$

$$\begin{array}{llll} \boldsymbol{\Lambda}_\mathbf{x} \equiv \mathbf{e}_\mathbf{xx} + \mathbf{z}_\mathbf{c}^\top \mathbf{c}_\mathbf{xx} & \boldsymbol{\Lambda}_\mathbf{k} \equiv \boldsymbol{\Sigma}_\mathbf{k}^{-1} \equiv (\mathbf{k}_\mathbf{u}\boldsymbol{\Lambda}_\mathbf{u}^{-1}\mathbf{k}_\mathbf{u}^\top)^{-1} & \mathbf{T}_\mathbf{cx} \equiv \mathbf{c}_\mathbf{x}\boldsymbol{\Lambda}_\mathbf{x}^{-1} & \mathbf{T}_\mathbf{ux} \equiv \mathbf{T}_\mathbf{uc}\mathbf{T}_\mathbf{cx} \\ \boldsymbol{\Sigma}_\mathbf{c} \equiv \mathbf{c}_\mathbf{x}\boldsymbol{\Lambda}_\mathbf{x}^{-1}\mathbf{c}_\mathbf{x}^\top & \boldsymbol{\Sigma}_\mathbf{u} \equiv \boldsymbol{\Lambda}_\mathbf{u}^{-1} - \boldsymbol{\Lambda}_\mathbf{u}^{-1}\mathbf{k}_\mathbf{u}^\top\boldsymbol{\Lambda}_\mathbf{k}\mathbf{k}_\mathbf{u}\boldsymbol{\Lambda}_\mathbf{u}^{-1} & \mathbf{T}_\mathbf{uc} \equiv \mathbf{c}_\mathbf{u}^\top\boldsymbol{\Sigma}_\mathbf{c}^{-1} & \mathbf{T}_\mathbf{kc} \equiv \mathbf{T}_\mathbf{ku}\mathbf{T}_\mathbf{uc} \\ \boldsymbol{\Lambda}'_\mathbf{u} \equiv \mathbf{g}_\mathbf{uu} + \mathbf{z}_\mathbf{c}^\top\mathbf{c}_\mathbf{uu} + \mathbf{z}_\mathbf{k}^\top\mathbf{k}_\mathbf{uu} & \boldsymbol{\Lambda}_\mathbf{c} \equiv \boldsymbol{\Sigma}_\mathbf{c}^{-1} - \boldsymbol{\Sigma}_\mathbf{c}^{-1}\mathbf{c}_\mathbf{u}\boldsymbol{\Sigma}_\mathbf{u}\mathbf{c}_\mathbf{u}^\top\boldsymbol{\Sigma}_\mathbf{c}^{-1} & \mathbf{T}_\mathbf{ku} \equiv \mathbf{k}_\mathbf{u}\boldsymbol{\Lambda}_\mathbf{u}^{-1} & \mathbf{T}_\mathbf{kx} \equiv \mathbf{T}_\mathbf{ku}\mathbf{T}_\mathbf{uc}\mathbf{T}_\mathbf{cx} \\ \boldsymbol{\Lambda}_\mathbf{u} \equiv \boldsymbol{\Lambda}'_\mathbf{u} + \mathbf{c}_\mathbf{u}^\top\boldsymbol{\Sigma}_\mathbf{c}^{-1}\mathbf{c}_\mathbf{u} & \boldsymbol{\Sigma}_\mathbf{x} \equiv \boldsymbol{\Lambda}_\mathbf{x}^{-1} - \boldsymbol{\Lambda}_\mathbf{x}^{-1}\mathbf{c}_\mathbf{x}^\top\boldsymbol{\Lambda}_\mathbf{c}\mathbf{c}_\mathbf{x}\boldsymbol{\Lambda}_\mathbf{x}^{-1} & & \end{array}$$

for notation. All quantities are evaluated at the current $(\mathbf{x}, \mathbf{u})$. Subscripts $\mathbf{x}$, $\mathbf{u}$ on functions e, g, $\mathbf{c}$ and $\mathbf{k}$ indicate derivatives. Otherwise they are just labels. The SQP update equations can be written as a symmetric block tridiagonal linear system

$$\boldsymbol{\Lambda} \begin{pmatrix} \delta\mathbf{x} \\ \mathbf{z}_\mathbf{c} + \delta\mathbf{z}_\mathbf{c} \\ \delta\mathbf{u} \\ \mathbf{z}_\mathbf{k} + \delta\mathbf{z}_\mathbf{k} \end{pmatrix} = - \begin{pmatrix} \mathbf{e}_\mathbf{x} \\ \mathbf{c} \\ \mathbf{g}_\mathbf{u} \\ \mathbf{k} \end{pmatrix} \tag{1}$$

The second and fourth equations say that the updated constraint violations $\mathbf{c}(\mathbf{x} + \delta\mathbf{x}, \mathbf{u} + \delta\mathbf{u})$ and $\mathbf{k}(\mathbf{u} + \delta\mathbf{u})$ vanish to first order. The first and third say that the updated data and parameter errors have been minimized to second order (*i.e.* the updated error gradients are perpendicular to the constraint surfaces $\mathbf{e}_\mathbf{x}(\mathbf{x} + \delta\mathbf{x}) + \mathbf{z}_\mathbf{c}^\top\mathbf{c}_\mathbf{x} \approx \mathbf{0}$, $\mathbf{g}_\mathbf{u}(\mathbf{u} + \delta\mathbf{u}) + \mathbf{z}_\mathbf{c}^\top\mathbf{c}_\mathbf{u} + \mathbf{z}_\mathbf{k}^\top\mathbf{k}_\mathbf{u} \approx \mathbf{0}$).

The coefficient matrix $\boldsymbol{\Lambda}$ has a block tridiagonal recursive decomposition [12] $\mathsf{LDL}^\top$, where L is a blocked lower band unit matrix and D is block diagonal. The inverse $\boldsymbol{\Sigma} = \boldsymbol{\Lambda}^{-1}$ is dense, but linear algebra can be performed efficiently using the sparse decomposed forms. For example linear equations can be solved by forward propagation (effectively multiplying by L^{-1}, accumulating components downwards), followed by back propagation (multiplying by $\mathsf{L}^{-\top}$, accumulating components upwards) [12]. Note that $\boldsymbol{\Lambda}_\mathbf{x}$ and $\boldsymbol{\Lambda}_\mathbf{u}$ are usually positive (semi)definite (*e.g.* if $\mathbf{e}_\mathbf{xx}$ and $\mathbf{g}_\mathbf{uu} + \mathbf{c}_\mathbf{u}^\top\boldsymbol{\Sigma}_\mathbf{c}\mathbf{c}_\mathbf{u}$ dominate the curvature terms $\mathbf{c}_\mathbf{xx}$, $\mathbf{c}_\mathbf{uu}$, $\mathbf{k}_\mathbf{uu}$), while the diagonal blocks $-\boldsymbol{\Sigma}_\mathbf{c}$ and $-\boldsymbol{\Sigma}_\mathbf{k}$ corresponding to the constraints $\mathbf{c}$ and $\mathbf{k}$ are typically *negative* (semi)definite.

5 Implementation Details

The code is packaged as a generic search driver which calls two user routines to evaluate e_i, $\boldsymbol{\Lambda}_{\mathbf{x}i}$, $\mathbf{c}_i$, $\mathbf{c}_{\mathbf{x}i}$ and $\mathbf{c}_{\mathbf{u}i}$ for each point, and g, $\mathbf{g}_\mathbf{u}$, $\mathbf{g}_\mathbf{uu}$, $\mathbf{k}$, and $\mathbf{k}_\mathbf{u}$. Only the user routines are problem specific. We allow for a few special cases like diagonal or constant, pre-decomposed $\boldsymbol{\Lambda}_{\mathbf{x}i}$, since this can significantly increase the overall speed. The rest of this section gives details of the technical implementation of the driver, and can be skipped by non-technicians.

Splitting $\mathbf{x}$ and $\mathbf{c}$ into individual data points $\mathbf{x}_i$ and constraints $\mathbf{c}_i$ and rearranging in terms of $(\mathbf{x}_i, \mathbf{c}_i)$ pairs, the coefficient matrix $\boldsymbol{\Lambda}$ takes the form

$$\begin{pmatrix} \boxed{\begin{matrix} \boldsymbol{\Lambda}_{\mathbf{x}1} & \mathbf{c}_{\mathbf{x}1}{}^\top \\ \mathbf{c}_{\mathbf{x}1} & \mathbf{0} \end{matrix}} & & & \boxed{\begin{matrix} \mathbf{0} \\ \mathbf{c}_{\mathbf{u}1} \end{matrix}} \\ & \ddots & & \vdots \\ & & \boxed{\begin{matrix} \boldsymbol{\Lambda}_{\mathbf{x}n} & \mathbf{c}_{\mathbf{x}n}{}^\top \\ \mathbf{c}_{\mathbf{x}n} & \mathbf{0} \end{matrix}} & \boxed{\begin{matrix} \mathbf{0} \\ \mathbf{c}_{\mathbf{u}n} \end{matrix}} \\ \boxed{\begin{matrix} \mathbf{0} & \mathbf{c}_{\mathbf{u}1}{}^\top \end{matrix}} & \cdots & \boxed{\begin{matrix} \mathbf{0} & \mathbf{c}_{\mathbf{u}n}{}^\top \end{matrix}} & \boxed{\begin{matrix} \boldsymbol{\Lambda}'_\mathbf{u} & \mathbf{k}_\mathbf{u}^\top \\ \mathbf{k}_\mathbf{u} & \mathbf{0} \end{matrix}} \end{pmatrix}$$

Solving the SQP update equations requires evaluation and decomposition of this matrix, forward substitution of the right hand side, then back substitution to find the update. Whatever the decomposition method, decomposition and forward substitution treat the diagonal block for each pair $(\mathbf{x}_i, \mathbf{c}_i)$ separately, followed by an accumulation over the $\mathbf{c}_{\mathbf{u}i}$ terms to give the $\mathbf{u}$ ones and treatment of the $(\mathbf{u}, \mathbf{k})$ block. Back substitution propagates the $\mathbf{u}$ solution back up into each individual $(\mathbf{x}_i, \mathbf{c}_i)$ block via the $\mathbf{c}_{\mathbf{u}i}$. The entire computation is $\mathcal{O}\left(n \cdot \left((|\mathbf{x}| + |\mathbf{c}|)^3 + |\mathbf{c}||\mathbf{u}|\right) + (|\mathbf{u}| + |\mathbf{k}|)^3\right)$ ($|\cdot| =$

$\dim(\cdot)$) — linear in the number n of data points. Only the nonzero blocks (including the diagonal zeros) need be stored, so the memory requirement is $\mathcal{O}(n)$ too.

There are several types of matrix decomposition that could be used, depending on how well conditioned the problem is. Above we gave the block $\mathsf{LDL}^\top$ decomposition which is theoretically convenient but not necessarily the best for applications. The current implementation uses a modified form of Choleski decomposition [12, 11]. This is very fast and has proved effective for all the examples considered in this paper, but it does not handle rank deficiency very gracefully. For example, it fails if the $\mathbf{c}_i$ are linearly dependent $\text{rank}(\mathbf{c}_{\mathbf{x}i}) < \dim(\mathbf{c})$. This happens, *e.g.*, in trifocal tensor estimation if more than the minimum 3 of the 9 components of the three image point matching constraint are taken (see below). The problem can be avoided by choosing a suitable subset or projection of the constraints, but it would be useful to have a stabler fall-back method for more difficult problems. Future work will investigate this: sparse QR and Householder reduction to tridiagonal form are likely to be the stablest (and slowest) possibilities, while the Aasen or Partlett-Reid factorizations [12] might prove useful intermediate solutions. However all of these methods are 2, 4 or more times slower than Choleski decomposition, which has proved adequate for all of the problems considered here.

The current implementation ignores the Lagrange ($\mathbf{z}$) terms in the $\mathbf{\Lambda_x}$ and $\mathbf{\Lambda_u}$ blocks, as we expect the point uncertainty to be small on the scale of the constraint (fitted surface) curvature for most realistic problems. (This does not change the solution, only the convergence rate). With this simplification and non-negative $\mathbf{e_{xx}}$ and $\mathbf{g_{uu}}$, the $\mathbf{\Lambda}$ matrices are all at least positive semi-definite and the implementation assumes this. Also, with many error models the $\mathbf{\Lambda_x}$ blocks are constant (and even diagonal) and can be decomposed in advance for speed.

Our Choleski implementation is slightly nonstandard in that the input matrix is not positive definite. This means that its decomposition is complex: the decomposed $\mathbf{c}$ and $\mathbf{k}$ columns are pure imaginary. However all the factors of i cancel out in the end, so it is only a matter of bookkeeping to keep track of where they should be and what signs they produce. To increase the stability slightly in near rank-deficient cases, we also use a version of Gill & Murray's modified Choleski factorization [11]. This edits (increases the modulus of) diagonal elements if they would be too small. For example, in our trivalent tensor routine, $\mathbf{k}(\mathbf{u})$ includes all 10 cubic constraints on the tensor, even though only 8 of these are linearly independent, but the stabilization is sufficient to handle this without problems.

Finally, note that the $\mathsf{LDL}^\top$ or Choleski decomposition of non-positive matrices (as here) is usually considered to be ill-conditioned and hence ill-advised [12, 11]. This is true in general, but the special structure of the problem helps us here. Provided there are no rank deficiencies, the diagonal $\mathbf{0}$'s give rise to diagonal blocks that are either strictly positive or strictly negative definite. This is enough to stabilize the problem (barring rank degeneracies!). However if any sort of regularizer (*e.g.* Levenberg-Marquardt or modified Choleski) is used, it is important regularize the negative blocks negatively, otherwise it *de*stabilizes the decomposition.

6 Covariance Estimation

After convergence, the matrix $\mathbf{\Sigma}$ gives a linearized estimate of the joint covariance of the estimated solution $(\hat{\mathbf{x}}, \hat{\mathbf{z}}_{\mathbf{c}}, \hat{\mathbf{u}}, \hat{\mathbf{z}}_{\mathbf{k}})$. In particular, the joint covariance of $(\hat{\mathbf{x}}, \hat{\mathbf{u}})$ is given by the submatrix

$$\hat{\mathbf{\Sigma}}_{\mathbf{xu}} \equiv \left\langle \begin{pmatrix} \boldsymbol{\delta}\mathbf{x}\boldsymbol{\delta}\mathbf{x}^\top & \boldsymbol{\delta}\mathbf{x}\boldsymbol{\delta}\mathbf{u}^\top \\ \boldsymbol{\delta}\mathbf{u}\boldsymbol{\delta}\mathbf{x}^\top & \boldsymbol{\delta}\mathbf{u}\boldsymbol{\delta}\mathbf{u}^\top \end{pmatrix} \right\rangle \approx \begin{pmatrix} \mathbf{\Sigma_x} & -\mathbf{T}_{\mathbf{ux}}^\top \mathbf{\Sigma_u} \\ -\mathbf{\Sigma_u T_{ux}} & \mathbf{\Sigma_u} \end{pmatrix}$$

This is dense as the $\hat{\mathbf{x}}_i$ are coupled to each other via $\hat{\mathbf{u}}$. It is also singular, $(\mathbf{c_x}\ \mathbf{c_u})\hat{\mathbf{\Sigma}}_{\mathbf{xu}} = \mathbf{0}$, because the constraints are exactly enforced to first order for all feasible $(\boldsymbol{\delta}\mathbf{x}, \boldsymbol{\delta}\mathbf{u})$: $\boldsymbol{\delta}\mathbf{c} = \mathbf{c_x}\boldsymbol{\delta}\mathbf{x} + \mathbf{c_u}\boldsymbol{\delta}\mathbf{u} = \mathbf{0}$. Similarly, $\mathbf{k_u \Sigma_u} = \mathbf{0}$. But $\hat{\mathbf{x}}$ alone has a covariance *across* the constraints: $\mathbf{c_x \Sigma_x} \neq \mathbf{0}$. This is the projection of the constraint-satisfying joint $(\mathbf{x}, \mathbf{u})$ variability. However note that the constraint violation and the $\mathbf{x}_i$-$\mathbf{x}_j$ couplings in $\mathbf{\Sigma_x}$ are small, typically $\mathcal{O}(1/n)$.

If $\mathbf{u}$ is frozen at the estimate $\hat{\mathbf{u}}$, the $\hat{\mathbf{x}}_i$ variations are decoupled and the $\mathbf{x}$ covariance is described not by $\mathbf{\Sigma_x}$ but by the sparse matrix $\mathbf{\Sigma_x}|_{\hat{\mathbf{u}}} \equiv \mathbf{\Lambda}_{\mathbf{x}}^{-1} - \mathbf{\Lambda}_{\mathbf{x}}^{-1}\mathbf{c}_{\mathbf{x}}^\top \mathbf{\Sigma}_{\mathbf{c}}^{-1}\mathbf{c_x}\mathbf{\Lambda}_{\mathbf{x}}^{-1}$ (*i.e.* $\mathbf{\Sigma_x}$ with $\mathbf{\Lambda_c}$ replaced by $\mathbf{\Sigma}_{\mathbf{c}}^{-1}$, since $\mathbf{\Sigma_u} \to \mathbf{0}$). This *does* satisfy $\mathbf{c_x \Sigma_x}|_{\hat{\mathbf{u}}} = \mathbf{0}$, and represents the projection of the unconstrained $\mathbf{x}$ covariance $\mathbf{\Lambda}_{\mathbf{x}}^{-1}$ onto $\mathcal{S}_{\hat{\mathbf{u}}}$, whereas $\mathbf{\Sigma_x}$ represents the (less singular) projection onto $\mathcal{S}$.

As always with constrained fitting, the covariances are rank deficient because variations that violate the active constraints are forbidden. The corresponding information matrices do not exist, although pseudo-inverses can sometimes be substituted. If necessary, the results can be transformed into any convenient minimal (unconstrained) local parametrization by the usual covariance propagation formula $\mathbf{\Sigma_x} \to \mathbf{J \Sigma_x J}^\top$, where $\mathbf{J}$ is the (here rectangular) Jacobean $[\partial\mathbf{x}'/\partial\mathbf{x}]$. In favourable cases, the results are regular nonsingular covariance matrices, with well-defined inverses. However for most applications it is usually much more convenient not to calculate the covariances or informations at all, but rather to store them in some factorized form that allows easier, more efficient numerical manipulation. (Choleski decomposition is used in our current implementation).

In contrast, if the parameters have a gauge freedom it is the information matrices that are rank deficient and the covariances that do not exist. Since gauge variations have no effect at all on the fit, they have zero information and infinite covariance. Some problems (*e.g.* estimating projection matrices by autocalibration) combine both constraints and a gauge freedom, in which case an orthogonal numerical approach like QR, SVD or pseudo-inverse seems essential.

7 Reduced Fitting Methods

Now we return to the theory and derive approximate, reduced versions of the full method, in which the data vector $\mathbf{x}$ is not explicitly updated, but is represented as an implicit first order correction about a fixed expansion point $\mathbf{x}^0$. In the sense that they approximately minimize a "reduced error metric" very similar to those used by standard first order geometric fitting techniques like gradient-weighted least squares, these methods can be seen as generalizations (and indeed derivations) of the standard approach, to allow constraints $\mathbf{k}(\mathbf{u})$ on the parameters $\mathbf{u}$. However, we stress that all reduced methods are first order approximations. It is the full method evaluating at $(\hat{\mathbf{x}}, \hat{\mathbf{u}})$ that gives the exact optimal solution, not a reduced one evaluating at $(\mathbf{x}^0, \hat{\mathbf{u}})$.

It is convenient to choose the expansion point $\mathbf{x}^0$ to be an exact *unconstrained* minimum of e, so that $\mathbf{e_x}(\mathbf{x}^0) = \mathbf{0}$. For example, if the observations $\mathbf{y}_i$ are direct measurements of the data points $\mathbf{x}_i$ and there is no strong prior on $\mathbf{x}_i$, taking $\mathbf{x}_i^0 \equiv \mathbf{y}_i$ should globally minimize any reasonable error metric $e(\mathbf{x})$. Also, we will ignore the $\mathbf{\Lambda_x}$ constraint curvature penalty $\mathbf{z_c^\top c_{xx}}$ throughout this section, as we will not bother to estimate $\mathbf{z_c}$.

The most direct reduced approach is given by propagating the $\mathbf{x}$ and $\mathbf{c}$ blocks of (1) into $\mathbf{u}$ using a partial block tridiagonal forward substitution (*c.f.* row 3 of L^{-1}), to give the following reduced linear system

$$\begin{pmatrix} \mathbf{\Lambda_u} & \mathbf{k_u^\top} \\ \mathbf{k_u} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\delta}\mathbf{u} \\ \mathbf{z_k} + \boldsymbol{\delta}\mathbf{z_k} \end{pmatrix} = - \begin{pmatrix} \mathbf{g_u^\top} + \mathbf{c_u^\top \Sigma_c^{-1} \tilde{c}} \\ \mathbf{k} \end{pmatrix} \qquad (2)$$

where $\tilde{\mathbf{c}} \equiv \mathbf{c} - \mathbf{c_x \Lambda_x^{-1} e_x}$. This can be solved for $\boldsymbol{\delta}\mathbf{u}$ and $\mathbf{z_k} + \boldsymbol{\delta}\mathbf{z_k}$. In the full method, the results are then back propagated into $\mathbf{c}$ and $\mathbf{x}$ to find $\mathbf{z_c}$ and

$$\begin{aligned} \boldsymbol{\delta}\mathbf{x} &= -\mathbf{\Lambda_x^{-1}} \left(\mathbf{e_x} + \mathbf{c_x^\top \Lambda_c} (\tilde{\mathbf{c}} + \mathbf{c_u} \boldsymbol{\delta}\mathbf{u}) \right) \\ &= -\mathbf{\Lambda_x^{-1} c_x^\top \Lambda_c} (\mathbf{c} + \mathbf{c_u} \boldsymbol{\delta}\mathbf{u}) \qquad \text{if } \mathbf{e_x} = \mathbf{0} \end{aligned} \qquad (3)$$

The evaluation point is the current $(\mathbf{x}, \mathbf{u})$ and the full solution gives an improved $(\mathbf{x} + \boldsymbol{\delta}\mathbf{x}, \mathbf{u} + \boldsymbol{\delta}\mathbf{u})$ which satisfies the constraints to first order (2^{nd} if the Lagrange terms are included) and minimizes the error to second order. By design, the solution $(\mathbf{u} + \boldsymbol{\delta}\mathbf{u}, \mathbf{z_k} + \boldsymbol{\delta}\mathbf{z_k})$ is *independent* of the evaluation point $(\mathbf{x}, \mathbf{u})$, to first order in $(\boldsymbol{\delta}\mathbf{x}, \boldsymbol{\delta}\mathbf{u})$. The updated, minimized value of the error function is

$$\begin{aligned} e(\mathbf{x} + \boldsymbol{\delta}\mathbf{x}) &= e - \tfrac{1}{2} \mathbf{e_x^\top \Lambda_x^{-1} e_x} + \tfrac{1}{2} (\tilde{\mathbf{c}} + \mathbf{c_u} \boldsymbol{\delta}\mathbf{u})^\top \mathbf{\Lambda_c} (\tilde{\mathbf{c}} + \mathbf{c_u} \boldsymbol{\delta}\mathbf{u}) \\ &= e + \tfrac{1}{2} \mathbf{c^\top \Lambda_c c} \qquad \text{if } \mathbf{e_x} = \mathbf{0} \text{ and } \boldsymbol{\delta}\mathbf{u} = \mathbf{0} \end{aligned}$$

The first two terms give the unconstrained minimum, while the third gives the extra cost for enforcing the constraints. The overall effect of the update is to minimize the **reduced error metric**

$$e_{\text{reduced}}(\mathbf{u}) \equiv g(\mathbf{u}) + \tfrac{1}{2} \mathbf{c^\top \Lambda_c c} \qquad (4)$$

with respect to $\mathbf{u}$, with the convention that $\mathbf{\Lambda_c}$ is held constant in derivatives with respect to $\mathbf{u}$, so that the solution ($\boldsymbol{\delta}\mathbf{u} = \mathbf{0}$) occurs when $\mathbf{c^\top \Lambda_c c_u} + \mathbf{g_u} = \mathbf{0}$. If $g \equiv 0$ we can also simplify $\mathbf{\Lambda_c}$ to $\mathbf{\Sigma_c^{-1}}$ in the above equations[2], since $\mathbf{c^\top \Sigma_c^{-1} c_u} = \mathbf{0}$. In fact, the optimal $\mathbf{x}$ update for *fixed* $\mathbf{u}$ is $\boldsymbol{\delta}\mathbf{x} = -\mathbf{\Lambda_x^{-1} c_x^\top \Sigma_c^{-1} c}$, which gives the same updated error $e(\mathbf{x} + \boldsymbol{\delta}\mathbf{x}) = e + \frac{1}{2} \mathbf{c \Sigma_c^{-1} c}$. So minimizing $\mathbf{c^\top \Lambda_c c}$ or $\mathbf{c^\top \Sigma_c^{-1} c}$ amounts to finding the $\mathbf{u}$ most compatible with $\mathbf{x}$ in the sense that the estimated residual error for the first-order-corrected $\mathbf{x}$ at $\mathbf{u}$ is minimized.

Also, if we allow $\mathbf{\Lambda_c}$ or $\mathbf{\Sigma_c}$ to enter into $\mathbf{u}$ derivatives in (4) above, the net effect is to slightly increase the accuracy of the approximation. For example, using the standard identity $\boldsymbol{\delta}(\mathbf{A}^{-1}) = -\mathbf{A}^{-1} \cdot \boldsymbol{\delta}\mathbf{A} \cdot \mathbf{A}^{-1}$ on $\mathbf{\Sigma_c^{-1}}$ and the $\boldsymbol{\delta}\mathbf{x}$ update (3) we obtain

$$\begin{aligned} \tfrac{1}{2} \nabla_{\mathbf{u}} (\mathbf{c^\top \Sigma_c^{-1} c}) &= \mathbf{c^\top \Sigma_c^{-1}} (\mathbf{c_u} - \mathbf{c_{ux}} \cdot \mathbf{\Lambda_x^{-1} c_x^\top \Sigma_c^{-1} c}) \\ &\approx \mathbf{c^\top \Sigma_c^{-1}} (\mathbf{c_u} + \mathbf{c_{ux}} \cdot \boldsymbol{\delta}\mathbf{x}) \\ &\approx \mathbf{c^\top \Sigma_c^{-1}} \nabla_{\mathbf{u}} \mathbf{c}(\mathbf{x} + \boldsymbol{\delta}\mathbf{x}) \end{aligned} \qquad (5)$$

So the effect of the non-constant denominator $\mathbf{\Sigma_c}$ is to move the evaluation point of $\mathbf{c_u}$ onto the constraint surface to first order. This 'straightens' the constraint direction and partially corrects for the bias we introduced by holding $\mathbf{x}$ fixed.

In summary, we have the following **reduced fitting method**:

Reduced Fitting Method

Given:
– an unknown parameter vector $\mathbf{u}$
– error functions $e(\mathbf{x})$ and $g(\mathbf{u})$
– a data point $\mathbf{x}^0$ minimizing $e(\mathbf{x})$
– vectors of constraint functions $\mathbf{c}(\mathbf{x}, \mathbf{u})$ and $\mathbf{k}(\mathbf{u})$

Find $\hat{\mathbf{u}}$ minimizing $g + \frac{1}{2} \mathbf{c^\top \Lambda_c c}$ at $\mathbf{x} = \mathbf{x}^0$, subject to the constraints $\mathbf{k}(\mathbf{u}) = \mathbf{0}$. If required, update $\mathbf{x}^0$ by $\hat{\mathbf{x}} \approx \mathbf{x}^0 - \mathbf{\Lambda_x^{-1} c_x^\top \Lambda_c c}$. If $g = 0$, $\mathbf{\Lambda_c}$ can be replaced by $\mathbf{\Sigma_c^{-1}}$.

As familiar examples of reduced error metrics, consider the case of 2D implicit curve or 3D implicit surface fitting, for which a single scalar constraint $\mathbf{c}_i$ gives a 'gradient weighted' error measure [27]:

$$e(\mathbf{u}) = \frac{1}{2} \sum_i \frac{|\, \mathbf{c}_i(\mathbf{y}_i, \mathbf{u}) \,|^2}{\mathbf{c}_{\mathbf{x}i}(\mathbf{y}_i, \mathbf{u})^\top \, \mathbf{\Sigma}_{\mathbf{x}i} \, \mathbf{c}_{\mathbf{x}i}(\mathbf{y}_i, \mathbf{u})}$$

[2] Again $\mathbf{\Sigma_c}$ is held constant in derivatives. Similarly, if $g \not\equiv 0$ we can minimize $g(\mathbf{u}) + \frac{1}{2} (\mathbf{c} - \mathbf{c_u \Sigma_u g_u^\top})^\top \mathbf{\Sigma_c^{-1}} (\mathbf{c} - \mathbf{c_u \Sigma_u g_u^\top})$.

where $\mathbf{\Sigma}_{\mathbf{x}i}$ is the covariance of $\mathbf{y}_i$. For the estimation of the fundamental matrix from pairs of corresponding image points $(\mathbf{y}_i, \mathbf{y}'_i)$, the reduced error measure becomes:

$$e(\mathbf{u}) = \frac{1}{2} \sum_i \frac{|\, \mathbf{y}_i^\top \, \mathbf{F} \, \mathbf{y}'_i \,|^2}{\mathbf{y}'^\top_i \mathbf{F}^\top \mathbf{\Sigma}_{\mathbf{x}i} \mathbf{F} \mathbf{y}'_i + \mathbf{y}_i^\top \mathbf{F} \mathbf{\Sigma}_{\mathbf{x}'i} \mathbf{F}^\top \mathbf{y}_i} \qquad (6)$$

Note that this is the only one of the error metrics considered by Zhang [35] that is statistically correct to first order.

For the trifocal tensor, the reduced measure $\mathbf{c}^\top \mathbf{\Sigma}_{\mathbf{c}}^{-1} \mathbf{c}$ is too complicated to write out explicitly. The matrix $\mathbf{\Sigma}_{\mathbf{c}}$ is 3×3 if the constraints are fully projected[3]. The easiest way to evaluate $\mathbf{\Sigma}_{\mathbf{c}}^{-1}$ is to explicitly calculate and multiply $\mathbf{c}_{\mathbf{x}}$ and $\mathbf{\Lambda}_{\mathbf{x}}^{-1}$ and invert numerically. In fact, to implement the reduced method it seems easier to avoid explicitly differentiating $\mathbf{c}^\top \mathbf{\Sigma}_{\mathbf{c}}^{-1} \mathbf{c}$, instead directly using (2), or estimating $\mathbf{c}_{\mathbf{ux}}$ and using (5).

In fact, although the reduced method is simpler and faster than the full one for simple problems with scalar constraints, it has little if any advantage for more complex ones with multiple constraints. If the constraint covariance $\mathbf{\Sigma}_{\mathbf{c}} = \mathbf{c}_{\mathbf{x}}^\top \mathbf{\Lambda}_{\mathbf{x}}^{-1} \mathbf{c}_{\mathbf{x}}$ must be calculated and inverted numerically, the easiest way to implement the reduced method is as the forward propagation phase of the full one. Then — given that the required matrices have already been factorized — the extra cost of the back propagation to calculate the $\mathbf{x}$ updates and implement the full method is minimal.

Of course, the reduced method is also less precise than the full one in that it is based on a first order approximation to $\hat{\mathbf{x}}$. The suboptimality is perhaps small for most practical problems, but (5) and the relation to the true problem makes it clear that the *correct* evaluation point is always the (estimated) underlying point $\hat{\mathbf{x}}_i$, not the observation $\mathbf{y}_i$ or $\mathbf{x}_i^0$. In fact, although the full method necessarily costs more per iteration, it sometimes turns out to be faster than the reduced one because its more accurate evaluations give convergence in fewer iterations.

8 Relation to Other Methods

Our reduced error metrics coincide with those used in weighted orthogonal regression [4, 3], 'gradient weighted least squares' [27], and Kanatani's 'optimal parametric estimation' [20]. This is no accident as the derivations of all of these essentially amount to first order linearization, and there is only one statistically correct answer to this order. On the other hand, the associated numerical optimization techniques differ widely. Our method allows for arbitrary constraints $\mathbf{k}(\mathbf{u})$ on the parameters, whereas most of the other approaches assume that the problem can be reformulated in terms of an unconstrained minimal local parametrization.

'Levenberg-Marquardt' (Gauss-Newton with a diagonal regularizer) is probably the most popular technique. ODRPACK [4, 3] is a publicly available trust-region implementation designed for large orthogonal regression problems. Our current reduced implementations also include L-M regularizers and might be classed as "L-M with constraints", although it should be stressed that this leaves a wide choice of linear algebra methods, and it is largely these that determine the speed and stability of the final method.

Renormalization: Kanatani [19, 20] has developed an orthogonal least squares optimization technique called **renormalization**. This is closely related to the **total least squares** method [17]. It seems to be mainly intended for problems in which the constraints $\mathbf{c}(\mathbf{x}, \mathbf{u})$ are homogeneous linear in $\mathbf{u}$ (or can be made so by embedding $\mathbf{u}$ in a larger space and either ignoring or linearizing the $\mathbf{u}$-constraints that arise). In this case — modulo the dependence of the 'weight matrix' $\mathbf{\Sigma}_{\mathbf{c}}$ on $\mathbf{u}$ — the cost function is a homogeneous quadratic in $\mathbf{u}$ and can be minimized (subject to $\|\mathbf{u}\| = 1$) by an eigenvector routine. This minimum is slightly biased as it ignores the variation of $\mathbf{\Sigma}_{\mathbf{c}}$ with $\mathbf{u}$. To remove the bias, the $\mathbf{x}$ evaluation point must effectively be moved onto the constraint surface, as in (5) above. If we do this the minimum of the reduced error will be exactly zero. In renormalization, the correction is done implicitly (and slightly mysteriously), by calculating the bias of the quadratic matrix and subtracting it. The result is akin to eigenvalue estimation by inverse iteration [12] and converges quite rapidly. (However Kanatani suggests that this be implemented with a full eigenvector routine at each step, which seems rather wasteful). It is not clear to me that renormalization has any significant advantage over direct nonlinear optimization, but it remains an interesting technique.

Bayesian approach: All of the above discussion is within the framework of statistical point estimators (like maximum likelihood or MAP) on the combined data and parameter space $(\mathbf{x}, \mathbf{u})$. Another more Bayesian approach[4] focuses on the posterior *marginal* likelihood of $\mathbf{u}$, *i.e.* the $\mathbf{x}_i$ values are viewed as unwanted 'nuisance parameters' and averaged (integrated) out of the problem. This can seldom be done exactly, but if we assume that the constraints (and hence the foliation of $\mathcal{S}$ by feasible subspaces $\mathcal{S}_{\mathbf{u}}$ — *c.f.* fig. 1) are locally approximately linear and that the conditional posterior error distributions $\mathbf{p}(\mathbf{x}|\mathbf{y}, \mathbf{u})$ in $\mathcal{S}_{\mathbf{u}}$ are well localized and have volume roughly independent of $\mathbf{u}$, the

[3] One could also work with the pseudo-inverse of a redundant set of constraints — even with all 9 components of the trilinear matching constraints — but that would be still more complex.

[4] Sometimes called "empirical Bayes". A "pure Bayesian" would also hedge over the posterior for $\mathbf{u}$, in the final uses to which $\mathbf{u}$ is put. This is seldom feasible, but when it is it can significantly increase overall system robustness. It can be approximated, *e.g.*, by working with confidence intervals over $\mathbf{u}$.

values of the $\mathcal{S}_{\mathbf{u}}$ integrals can be approximated by the posterior probabilities (*i.e.* heights) of their modes. Hence, the ML estimate $\hat{\mathbf{u}}$ still holds, and can be recovered by minimizing the reduced error function $\frac{1}{2}\mathbf{c}^\top \boldsymbol{\Sigma}_{\mathbf{c}}^{-1}\mathbf{c}$. In particular, these assumptions hold for normal distributions, which have constant shape (but not height or position) over *any* decomposition into parallel subspaces[5].

9 Robust Formulation

All of the above discussion has been in a non-robust least-squares-like framework. Now we consider how to robustify the method against possible outliers. Since this is a local approach we assume that reasonable initial estimates of $\mathbf{u}$ and (if necessary) $\mathbf{x}$ have already been found, *e.g.* by a robust consensus search.

One path to robustness would be to replace the error metrics $e_i(\mathbf{x}_i|\mathbf{y}_i)$ with robust M-estimators, perhaps derived from an outlier-polluted observation distribution $\mathbf{p}(\mathbf{y}_i) \sim \mathbf{p}(\mathbf{y}_i|\mathbf{x}_i) + \mathbf{p}(\mathbf{y}_i|\text{outlier})$. This is possible, but not recommended. For one thing, it gives highly non-convex error surfaces which make numerical optimization difficult. In particular, for distant outliers $\mathbf{y}_i$, the connection to the corresponding estimated data point $\mathbf{x}_i$ is 'switched off', so that $\mathbf{x}_i$ is free to wander anywhere on the constraint surface. This leads to very poor numerical conditioning, so the $\mathbf{x}_i$ need to be 'frozen' by some sort of regularization procedure.

More importantly, a robustified $e(\mathbf{x}_i|\mathbf{y}_i)$ may not be what we really wanted in any case. Even gross outliers usually correspond to *something* in the image, and it is at least not completely implausible to hypothesize that whatever that something is, the coupling of data to observation is roughly described by the given error model $e(\mathbf{x}_i|\mathbf{y}_i)$. What is in question about outliers is not so much their error model, as the fact that they belong to the fitted constraint or surface at all. In other words, it is the connection of $\mathbf{x}_i$ to $\mathbf{u}$ that needs to be 'robustified', not that of $\mathbf{x}_i$ to $\mathbf{y}_i$. For example, for epipolar geometry estimation the outliers are *mismatches* — real image events with presumably correct uncertainty models that just happen not to be in correspondence. Even for near outliers resulting from poorly localized image tokens, it is unclear that a weakened data-observation coupling is really more appropriate than a weakening of their influence on the fit. In any case, we will take the view that outliers are potentially 'good' points that just happen not to lie on the constraint surface, and develop a model with robust $\mathbf{x}$-$\mathbf{u}$ coupling.

The default working hypothesis is that apparent outliers are potential inliers, but ones that we are not currently prepared to trust. Although we want to reduce their influence on the fit $\mathbf{u}$, we still want the data estimates $\mathbf{x}_i$ to track the evolving constraint surface, as this is useful information under the null hypothesis that they are actually inliers. Hence, the $\mathbf{x}$-$\mathbf{y}$ and $\mathbf{x}$-$\mathbf{c}$ couplings $e(\mathbf{x}_i|\mathbf{y}_i)$ and $\mathbf{c}_{\mathbf{x}i}$ must be left at full strength to ensure that the $\mathbf{x}_i$ stay both on the surface and under the influence of the observations $\mathbf{y}_i$. In fact, it makes no sense to include a point in the fit unless both terms remain active, since otherwise it will either converge to $\mathbf{y}_i$ or wander uselessly across the surface.

The only remaining candidate for robustification is the $\mathbf{c}$-$\mathbf{u}$ coupling $\mathbf{c}_{\mathbf{u}i}$. We will use a simple probabilistic model, but many alternatives are possible. Given outlier-polluted observation distributions $\mathbf{p}(\mathbf{y}_i) \sim \mathbf{p}(\mathbf{y}_i|\mathbf{x}_i)+\mathbf{p}(\mathbf{y}_i|\text{outlier})$, estimate weighting factors (inlier probabilities) $\rho_i(\mathbf{x}_i, \mathbf{u}) = \mathbf{p}(\text{inlier}|\hat{\mathbf{x}}_i, \mathbf{y}_i) \approx \mathbf{p}(\mathbf{y}_i|\mathbf{x}_i)/\mathbf{p}(\mathbf{y}_i|\text{outlier})$, and replace each $\mathbf{c}_{\mathbf{u}i}$ with $\rho_i \mathbf{c}_{\mathbf{u}i}$. This amounts to using the *expected* coupling strength, since the correct coupling is 1 for inliers and 0 for outliers. It is as if the constraints varied more slowly with the parameters for unreliable points.

For diagnostic purposes and convergence testing we also calculate the weighted fitting error $\sum_i \rho_i e_i$. However we emphasize that the algorithm actually minimizes the *un*weighted error — derivatives of $\rho_i(\mathbf{x}_i)$ are never needed, and there is no non-convexity.

For the reduced method, the weight factors ρ can be estimated from the implicit $\boldsymbol{\delta}\mathbf{x}$ and the reduced error contributions $\mathbf{c}_i \boldsymbol{\Lambda}_{\mathbf{c}i} \mathbf{c}_i \approx \boldsymbol{\delta}\mathbf{x}_i^\top \boldsymbol{\Lambda}_{\mathbf{x}i} \boldsymbol{\delta}\mathbf{x}_i$ (*c.f.* (3)). Again, the ρ's are weighting factors for use on the right hand side of (2), they do not enter into derivatives with respect to $\mathbf{u}$.

10 Matching Tensor Estimation

As examples of the full and reduced methods, we consider the precise estimation of the fundamental matrix, the essential matrix and the trifocal tensor, in each case taking the full nonlinear constraints into account. We work in homogeneous coordinates and assume basic familiarity with projective vision (*e.g.* see [5]).

Fundamental matrix: This case is fairly simple, so we can use the reduced method with the statistically correct error metric (6). The parameter vector $\mathbf{u}$ contains all 9 components of $\mathbf{F}$. The constraint $\det(\mathbf{F}) = 0$ gives us one component of $\mathbf{k}$, but we also include a second constraint $\|\mathbf{F}\|^2 = \sum_{AB'} \mathbf{F}_{AB'}^2 = 2$ to ensure that $\mathbf{F}$ stays well normalized.

Essential matrix: Here the constraints $\mathbf{k}$ are a little more complicated, but we can continue to use the reduced method and the full 9-component parametrization with a normalization constraint. (Another possibility would be a rotation-translation parametrization, $\mathbf{E} \sim [\mathbf{t}]_\times \mathbf{R}$). The Demazure

[5] To see this, work in coordinates aligned with the subspaces. The modes and heights are given by sectioning with the Mahalanobis-orthogonal subspace.

constraint [5]

$$\mathbf{M}(\mathbf{E})\mathbf{E} = \mathbf{0} \quad \text{where} \quad \mathbf{M}(\mathbf{E}) \equiv \frac{1}{2}\,\text{trace}(\mathbf{E}\mathbf{E}^\top) - \mathbf{E}\mathbf{E}^\top$$

has 9 components, but we know that only $8 - 5 = 3$ of these are algebraically independent. Using all 9 constraints would be quite slow as we would have to process (including normalization) a 10×9 rank 4 Jacobean $\mathbf{k_u}$ and a 10×10 symmetric rank 4 covariance matrix $\mathbf{\Sigma_k}$. Instead, we heuristically choose a subset of 3 "strong" constraints and ignore the rest. It is not too important how this is done, so long as the result has a well-conditioned rank 3 Jacobean. For instance, we can project out the 3 constraints $\mathbf{d}^\top\mathbf{M}(\mathbf{E})\mathbf{E}$ along some direction $\mathbf{d}$. An exact essential matrix $\mathbf{E} \sim [\mathbf{t}]_\times\mathbf{R}$ satisfies $\mathbf{M}(\mathbf{E}) \sim \mathbf{t}\mathbf{t}^\top$, so the "strongest" direction is $\mathbf{d} \approx \mathbf{t}$ in the sense that for orthogonal directions $\mathbf{d}^\top\mathbf{t} = 0$ and an exact $\mathbf{E}$, the constraints trivially vanish. In practice, we estimate $\mathbf{d} \approx \mathbf{t}$ heuristically from the columns of the approximately rank 1 matrix $\mathbf{M}(\mathbf{E})$, and project both the Demazure constraints and their Jacobean along this. The Jacobean is lengthy, so we used MAPLE to calculate it.

Trifocal tensor: This case is quite complicated and it is easiest to use the full method. We assume some knowledge of tensors and properties of the trifocal tensor, see [25, 14, 31, 30]. We focus on the point matching case (lines are also possible), and only consider one of the 3 distinct $3 \times 3 \times 3$ trifocal tensors between the 3 images, say $\mathbf{G}_{A_1}{}^{B_2C_3}$. (The indices are labelled by the image they come from, and the summation convention applies, *c.f.* [31]). Contracting $\mathbf{G}$ with any image 1 point $\mathbf{x} = \mathbf{x}^{A_1}$ gives a 3×3 matrix, say $(\mathbf{G} \cdot \mathbf{x}) \equiv \mathbf{G}_{A_1}{}^{B_2C_3}\mathbf{x}^{A_1}$. If $\mathbf{x}'$ and $\mathbf{x}''$ are the image 2 and 3 points corresponding to $\mathbf{x}$, and $\mathbf{e}'$ and $\mathbf{e}''$ are the epipoles of image 1 in image 2 and 3, a *closure identity* [30] shows that (with a suitable choice of relative scales)

$$(\mathbf{G} \cdot \mathbf{x}) \sim \mathbf{x}'e''^\top - e'\mathbf{x}''^\top \tag{7}$$

This has rank at most two, so $\det(\mathbf{G} \cdot \mathbf{x}) = 0$. This holds for any $\mathbf{x}$, so it amounts to an algebraic constraint on $\mathbf{G}$. There are 10 such constraints in all, encapsulated in the 10 independent components of the symmetric tensorial expression $\varepsilon_{A_2B_2C_2}\ \varepsilon_{A_3B_3C_3}\ \mathbf{G}_{A_1}{}^{A_2A_3}\ \mathbf{G}_{B_1}{}^{B_2B_3}\ \mathbf{G}_{C_1}{}^{C_2C_3} = \mathbf{0}$. (The ε's are alternating tensors that effectively take a 3×3 determinant. The result is automatically symmetric in the free indices $A_1B_1C_1$). These constraints are the trifocal analogues of $\det(\mathbf{F}) = 0$. They are not all linearly independent: explicit calculation shows that the rank of their 27×10 Jacobean is generically 8. Another way to write them is just to take $\det(\mathbf{G} \cdot \mathbf{x}_i) = 0$ for any 8 sufficiently general vectors $\mathbf{x}_i$. For basis vectors $\mathbf{x}_i$ (*i.e.* $A_1 = B_1 = C_1$) the constraints are well known, *e.g.* [25].

On the other hand, the 26 (up to scale) components of $\mathbf{G}$ characterize the 3 image projective camera geometry, which has $3(3\times4-1)-(4\times4-1) = 18$ degrees of freedom (*i.e.* three 3×4 projection matrices modulo a 4×4 choice of 3D projective frame). So the components of $\mathbf{G}$ are subject to $26 - 18 = 8$ independent algebraic constraints. Since the above cubic constraints generically have rank 8, they generically span the complete space of constraints on $\mathbf{G}$. (There may be exceptional points at which this is not true).

The trifocal point matching constraints $[\mathbf{x}']_\times(\mathbf{G} \cdot \mathbf{x})[\mathbf{x}'']_\times = \mathbf{0}$ can be derived by multiplying (7) on the left and right by $[\mathbf{x}']_\times$ and $[\mathbf{x}'']_\times$. Only 3 of these $3 \times 3 = 9$ equations are linearly independent for any given point triplet. Their projections along either $\mathbf{x}'$ (on the left) or $\mathbf{x}''$ (on the right) trivially vanish, leaving only the $2 \times 2 = 4$ orthogonal relations. And even if $\mathbf{x}$, $\mathbf{x}'$ and $\mathbf{x}''$ do not correspond, (7) guarantees that the simultaneous projection along e' and e'' also vanishes, so one more degree of freedom is lost. Given these facts, suitably "strong" pairs of projection directions $\{\mathbf{d}', \mathbf{x}' \wedge \mathbf{d}'\}$ and $\{\mathbf{d}'', \mathbf{x}'' \wedge \mathbf{d}''\}$ are easily found by using $[\mathbf{x}']_\times(\mathbf{G} \cdot \mathbf{x}) \sim (\mathbf{x}' \wedge \mathbf{e}')\mathbf{x}''^\top$ to estimate $\mathbf{d}' \sim \mathbf{x}' \wedge \mathbf{e}'$, and symmetrically for $\mathbf{d}''$. The resulting three relations generically span the $3 \cdot 2 - 3 = 3D$ space of algebraic matching constraints on the $3 \cdot 2 = 6$ pixel coordinates of the image points, modulo the 3 d.o.f. of their parent 3D point (*i.e.* the projected 3×6 $\mathbf{c}_{\mathbf{x}i}$ Jacobean generically has rank 3).

All this gives the following implementation. Each point triplet is represented by a data vector $\mathbf{x}_i$ of 6 pixel coordinates, and the trifocal tensor $\mathbf{G}$ is represented by a 27 component vector $\mathbf{u}$. For each triplet, $\mathbf{c}_i$ contains the three projected trilinear constraints. At present $\mathbf{k}$ contains a normalization constraint and all 10 cubic constraints on $\mathbf{G}$, as we have no insight as to how to reliably choose 8 strong projection directions (short of an orthogonal projection of the Jacobean $\mathbf{k_u}$, which would probably take longer than using all 10 constraints). The 11×27 Jacobean is complicated and was evaluated using MAPLE.

The rank deficiency of the $\mathbf{k_u}$ Jacobean seems to cause no significant numerical difficulties, even though the current Choleski-based implementation is potentially rather unstable under rank deficiency. Indeed, deficiency in $\mathbf{c_x}$ (*e.g.* including redundant matching constraints for each point triplet) causes significant instability as it occurs much earlier in the Choleski chain. A slower, more orthogonal numerical method would be needed to handle this, and is currently under investigation. (However we stress that for the current implementation this is not a problem).

11 Experiments

The methods described above are implemented and seem to work well, but at present we only have very preliminary experimental results. More will be given in the final version of this paper.

All of the methods converge quickly (say within 2–5 iterations) if started from a reasonable approximate solution, *e.g.* obtained from random sampling or (barring outliers) a linear method. For the fundamental matrix and trifocal methods the domain of convergence seems to be quite large. Preliminary trials suggest that even from a completely random initialization, the trifocal method converges reliably (but takes about 20 iterations) while the fundamental matrix method occasionally (with probability a few percent) falls into a local minimum. The essential matrix method seems to be fairly sensitive to errors in the assumed calibration. We are not yet sure whether this is a problem with the implementation or a genuine feature of the problem.

Note that 'residual fitting error' (*i.e.* the residual constraint violation of fitted observations against the estimated matching tensor) is *not* a suitable performance measure for the methods presented here. Imposing additional constraints on the tensors can only *increase* this apparent error, even though it makes the final estimates more accurate. To test the performance, we must either validate against ground truth or independent observations, or examine the residual 3D or reprojection errors of reconstructions based on the estimated tensors. Our present results use the latter approach.

We will illustrate the performance on precise data of 133 points obtained from a triplanar calibration grid with ~ 0.02 pixel error. We report the projective 3D and reprojection errors obtained from **closure based** projective reconstruction methods [33], as these directly use the matching tensors for reconstruction and seem to have error roughly proportional to their accuracy. (In contrast, factorization based reconstruction methods [26, 32] are much less sensitive, so are better for reconstruction but worse for the current test). In each case, some of the improvement in the results is due to the use of a more accurate error model, and some to enforcing the nonlinear constraints.

For the fundamental matrix the full and reduced methods gave very similar results, each decreasing the 3D residual by about 50% (0.037/0.070%) and the reprojection one by about 30% (0.018/0.027 pixel) over a well-conditioned linear method [15]. For the trifocal tensor, the full method improved the 3D residual by about 22% (0.031/0.040%) over a linear SVD based '7 point' estimator, but changed the reprojection one by only about 10% (0.018/0.020 pixel). For the essential matrix we used the same projective reconstruction and linear **F** matrix estimator for testing, but assumed a calibration matrix obtained from a conventional calibration method. The nonlinear method improved the linear results by about 25% (0.045/0.060%) for 3D reconstruction and about 8% (0.13/0.14 pixel) for reprojection, but these 3D results are about 18% worse than the uncalibrated ones given above so we suspect that our assumed calibration was not sufficiently accurate.

Run times seem quite reasonable. On the above examples on a Sun 4, our linear **F** matrix method takes about 14ms. Improving the results with either the full or reduced **F** matrix method takes an extra 18ms (but 28ms for an older, less efficient reduced implementation). The essential matrix method takes about 20ms, a linear trifocal one about 120ms and the nonlinear trifocal update about 230ms. (In each case, the nonlinear methods converged in 3 iterations).

12 Discussion & Conclusions

We have introduced a new and general approach to fitting geometric data under constraints. Its main characteristics are that it maintains explicit estimates of the positions of the "true underlying data", and that it is embedded in an efficient numerical framework designed to handle redundancy and constraints, not only between the data and the parameters, but also on the parameters themselves. The method can be applied to many practical vision problems, *e.g.* curve, surface and 3D model fitting, fundamental matrix and trifocal tensor estimation, reconstruction under constraints. In particular, we have discussed its application to the precise estimation of the fundamental matrix, the essential matrix and the trifocal tensor, in each case taking account of all the nonlinear constraints involved. The method allows an interesting robust formulation, which works well in practice and seems somewhat more principled than most existing approaches. We have also shown how our approach 'reduces' to a generalization of several existing ones in a linear approximation.

Future work will improve the numerical implementation, especially with a view to handling redundancies more gracefully, and look much more closely at the experimental performance, including handling of outliers and comparisons with alternative approaches.

References

[1] K. B. Atkinson. *Close Range Photogrammetry and Machine Vision*. Whittles Publishing, Roseleigh House, Latheronwheel, Caithness, Scotland, 1996.

[2] P. Beardsley, P. H. S. Torr, and A. Zisserman. 3d model acquisition from extended image sequences. In *European Conf. Computer Vision*, volume II, pages 683–95, Cambridge, U.K., April 1996.

[3] 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.

[4] P. T. Boggs and J. E. Rodgers. Orthogonal distance regression. *Contemporary Mathematics*, 112:183–94, 1990.

[5] O. Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. MIT Press, 1993.

[6] O. Faugeras, S. Laveau, L. Robert, G. Csurka, and C. Zeller. 3d reconstruction of urban scenes from sequences of images. Technical Report RR-2572, INRIA, Sophia Antipolis, France, June 1995.

[7] A. W. Fitzgibbon. A buyer's guide to conic fitting. In *British Machine Vision Conference*, pages 513–22, Birmingham, 1995.

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

[9] W. Förstner. 10 pros and cons of performance characterization in computer vision. In *Workshop on Performance Characterization of Vision Algorithms*, Cambridge, U.K., 1996.

[10] W. A. Fuller. *Measurement Error Models*. John Wiley, New York, 1987.

[11] P. Gill, W. Murray, and M. Wright. *Practical Optimization*. Academic Press, 1981.

[12] G. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1989.

[13] R. M. Haralick. Propagating covariance in computer vision. In K. Bowyer and N. Ahuja, editors, *Advances in Image Understanding: a Festschrift for Azriel Rosenfeld*. IEEE Computer Society Press, 1996.

[14] R. Hartley. Lines and points in three views – an integrated approach. In *Image Understanding Workshop*, Monterey, California, November 1994.

[15] R. Hartley. In defence of the 8-point algorithm. In E. Grimson, editor, *IEEE Int. Conf. Computer Vision*, pages 1064–70, Cambridge, MA, June 1995.

[16] R. Hartley and P. Sturm. Triangulation. In *ARPA Image Understanding Workshop*, pages 957–66, Monterey, November 1994.

[17] S. V. Huffel and J. Vandewalle. *The Total Least Squares Problem: Computational Aspects and Analysis*. SIAM Press, Philadelphia, PA, 1991.

[18] K. Kanatani. Statistical analysis of geometric computation. *Computer Vision, Graphics & Image Understanding*, 59(3):286–306, 1994.

[19] K. Kanatani. Statistical bias of conic fitting and renormalization. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 16(3):320–6, 1994.

[20] K. Kanatani. *Statistical Optimization for Geometric Computation: Theory and Practice*. Elsevier Science, Amsterdam, 1996.

[21] X. Liu, R. M. Haralick, and K. B. Thornton. Site model construction using geometric constrained optimization. In *DARPA Image Understanding Workshop*, volume I, pages 357–71, Palm Springs, CA, February 1996.

[22] X. Liu, T. Kanungo, and R. M. Haralick. Statistical validation of computer vision software. In *DARPA Image Understanding Workshop*, volume II, pages 1533–40, Palm Springs, CA, February 1996.

[23] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–5, 1981.

[24] A. Shashua. Algebraic functions for recognition. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 17(8):779–89, 1995.

[25] A. Shashua and M. Werman. On the trilinear tensor of three perspective views and its underlying geometry. In *IEEE Int. Conf. Computer Vision*, Boston, MA, June 1995.

[26] 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.

[27] G. Taubin. Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 13(11):1115–38, 1991.

[28] N. Thacker, D. Prendergast, and P. Rockett. B-Fitting: an estimation technique with automatic parameter selection. In *British Machine Vision Conference*, pages 283–92, Edinburgh, 1996.

[29] P. H. S. Torr and D. W. Murray. A review of robust methods to estimate the fundamental matrix. Technical report, Robotics Research Group, Oxford, U.K., 1996. Submitted to *Computer Vision, Graphics & Image Understanding*.

[30] B. Triggs. The geometry of projective reconstruction I: Matching constraints and the joint image. Submitted to *Int. J. Computer Vision*.

[31] B. Triggs. Matching constraints and the joint image. In E. Grimson, editor, *IEEE Int. Conf. Computer Vision*, pages 338–43, Cambridge, MA, June 1995.

[32] B. Triggs. Factorization methods for projective structure and motion. In *IEEE Conf. Computer Vision & Pattern Recognition*, pages 845–51, San Francisco, CA, 1996.

[33] B. Triggs. Linear projective reconstruction from matching tensors. In *British Machine Vision Conference*, pages 665–74, Edinburgh, September 1996.

[34] Z. Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. Technical Report 2676, INRIA, INRIA Sophia-Antipolis, France, 1996. To appear in Image and Vision Computing.

[35] Zhengyou Zhang. Determining the epipolar geometry and its uncertainty: A review. Technical Report 2927, INRIA, INRIA Sophia-Antipolis, France, 1996. To appear in *Int. J. Computer Vision*.