# Automatic Task Planning for Robot Vision

**Bill Triggs & Christian Laugier**
LIFIA, INRIA Rhône-Alpes,
46, avenue Félix Viallet, 38031 Grenoble, France.
*Bill.Triggs@imag.fr, Christian.Laugier@imag.fr*

**Abstract**

This paper describes an automated planner for visual inspection and monitoring tasks using a robot mounted CCD camera. It is capable of finding heuristically near-optimal viewing positions for a series of visual tasks, and of ordering them and planning robot paths between them to produce a near minimal cost overall task plan for the sequence. It has been optimized for intervention-style tasks with relatively large, cluttered workspaces, where it is difficult for a human operator to take account of all the constraints involved. The guiding principle of the work has been to aim for reliability by using quantitative, physically-based measures of quality wherever possible, and by using global search to ensure that possible solutions are not overlooked. We discuss a novel global function optimization technique based on decision theory and subjective models of function behaviour, that allows global viewpoint searches to run in the time usually required for local ones.

**Keywords:** Robot Task Planning, Machine Vision, Image Prediction, Global Optimization.

## 1 Introduction

This paper describes an automated task planner designed for visual inspection and monitoring using a robot mounted CCD camera. The system is capable of finding heuristically near-optimal viewing positions for a series of visual tasks, and of ordering the tasks and planning robot paths between them to produce a near minimal cost overall task plan for the sequence.

The design has been optimized for intervention-style applications such as the maintenance and decommissioning of nuclear power plants and deep sea oil rigs. The environments and tasks in these applications are often extremely complex. They can usually be modelled only imperfectly at a coarse scale. Reliability is a key issue, so active sensing is required to reduce uncertainty and ensure robustness. In practice the sensor needs to be mobile (robot-mounted) to avoid occlusion and direct attention towards the points where it is most needed, but this brings up the problem of sensor management. When the complexities of robot kinematics, environmental collision avoidance and visual occlusion are combined with task constraints and more traditional visual indices such as field of view, resolution and focus, it is seldom easy for a human operator to guess where the sensor should best be placed to execute its task effectively, so automatic placement planning is required.

To get some idea of the kind of task the planner was designed for, consider fig. 1. The problem is loosely based on a routine task in nuclear power plant maintenance. The area around the base of the columns must be inspected for damage or detritus. Here we have planned viewpoints only for the difficult-to-see regions *behind* the columns. The sequence of views selected by the system is shown in the lower panel. Note that the robot is at the limit of its reach for several of the views.

Our system is currently in its second generation, being a significantly extended rewrite of the single viewpoint planner described in [21]. It can be configured for a variety of tasks. For example the first version of the planner was integrated into a visually guided grasping system developed under the European Esprit collaboration SECOND [4, 11], where it was used to choose viewpoints for workpiece verification and pose correction and for visual servoing of the grasp approach.

The central activity of the system is the selection of viewing positions and the associated camera robot poses on the basis of task, camera, robot and environmental preferences and constraints. In practice the constraints are often very restrictive and it is necessary to reach a compromise solution. To support this we have adopted an approach based on the global optimization of a heuristically weighted sum of quality metrics, over a search region defined by the true hard constraints such as kinematic reachability. Each metric measures a single distinct aspect of viewpoint quality with respect to the relevant constraints. As far as possible, the as-

To appear in *Int. Symposium Robotics Research*, Munich, Oct. 1995. This work was supported by the European Community through Esprit programs HCM and SECOND.

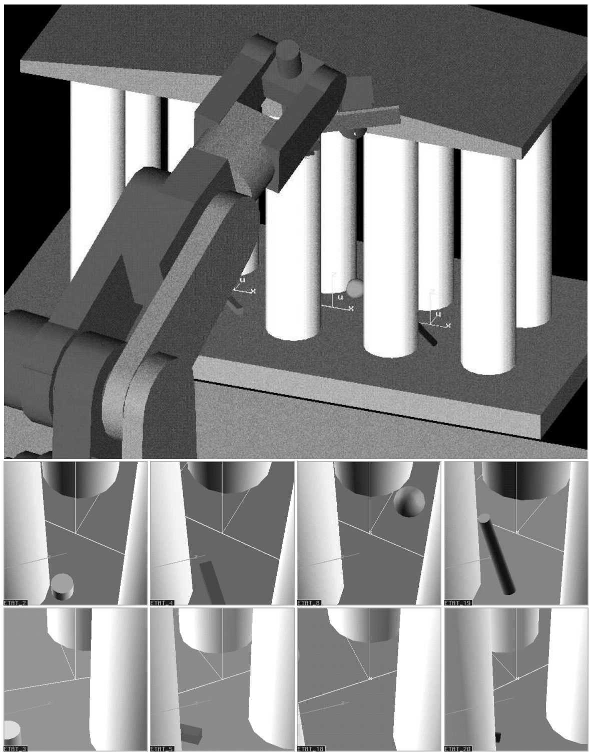

*Figure 1*: A routine inspection task. The areas behind the columns must be inspected for damage or detritus. The sequence of views selected by the system is shown in the lower panel, reading anticlockwise from top right.

sessments are based on objective physical and optical quantities. For image-based metrics, we predict the expected image and perform direct measurements on it.

At present the planner is entirely model-based. There is no feedback from the images taken, although ultimately this would be desirable. As input, it requires a camera calibration and a list of specifications of visual tasks to be executed. These include the location and geometry of the object or region to be viewed and constraints on the viewing direction, field of view, resolution and focus required to complete the task.

The system has three principal components, which are relatively loosely coupled in the present implementation: the viewpoint assessor, the viewpoint search routine, and the tour planner. Each is discussed more fully below, but in brief they work as follows:

- The viewpoint assessor estimates the quality of a single prospective camera placement for a given visual task, as above.

- For each task, the search routine calculates a space of potentially feasible camera placements and searches it to find a globally quasi-optimal viewpoint, using the assessor to evaluate the trial viewpoints it generates. To speed this step we have developed a novel and highly efficient global search technique based on decision theory and subjective models of likely function behaviour. This will be described in detail.

- Once the search routine has found viable viewpoints (camera robot poses) for each of the visual tasks, the tour planner is called. This takes the set of planned viewpoints and converts it into a complete task plan by choosing an efficient ordering for the tasks and planning manoeuvres to move the camera robot safely through the sequence of viewing positions.

# 2 Previous Work

There have been relatively few previous studies of camera placement planning, mostly concentrated on visual and task constraints. Cowan [6, 7] developed geometric constraint based models of resolution, focus, aperture and occlusion, and also considered simultaneous camera and light source placement. Yi, Haralick and Shapiro [22] studied the camera/light source problem too. But perhaps the most complete previous system is that of Tarabanis and Tsai [18, 19], who also took a constraint-based approach and developed a fairly sophisticated CAD based model of workpiece self-occlusion for their MVP (Machine Vision Planner) system.

We feel that the above work has several major limitations that make it unsuitable for our application:

- *Workspace constraints* on the camera-carrying robot are not considered. This is unrealistic because in practice these are often the most restrictive constraints in the whole problem, especially for fixed-base robots in large workspace intervention-style applications. As far as we are aware, the only previous study to consider workspace constraints is that of Al-Chami [1, 2], which can be viewed as a precursor of the present work.

- *Image-based quantities* are avoided. For example, occlusion is evaluated by back-projecting the ‘occlusion shadows’ of obstacles as seen from the task and excluding every shadowed viewpoint. This sort of ‘backwards’ approach does not generalize easily, *e.g.* to quantitative measures of occlusion. Ultimately, the most satisfactory way to judge expected image quality is to actually predict the image and make direct measurements on it.

- The approach is *constraint based* rather than *decision based*. The emphasis is on passively defining regions of viewpoints with ‘nominally satisfactory’ focus, resolution, occlusion and so forth, rather than on actively searching for a single ‘optimal’ solution. But real applications need to make *decisions*: a single viewpoint must be chosen, and in practice the choice must often be a compromise between several conflicting constraints. A heuristic combination of continuous quality metrics is needed for this: binary classifications do not provide sufficient discrimination.

- *Search issues* are largely ignored. The viewpoint assessment function often turns out to be highly nonconvex, with several distinct feasible regions where occlusion or kinematic constraints cut the workspace in two. The use of a global optimization technique is essential for reliability. Otherwise it is impossible to guarantee that a tentative solution is the *global* extremum and not just a relatively poor local one.

In summary, the approach we advocate is based on the global optimization of a heuristically weighted viewpoint evaluation function, that takes robot workspace constraints into account as well as the more traditional optical and task constraints. As far as possible, the evaluations should be based on objective physical measurements, with image prediction and measurement being used for image-derived quantities.

A program such as this must draw on a rather diverse range of sources. Apart from physical optics (focus, resolution, lighting…), computational geometry (task & environment modelling) and computer graphics (image prediction), we will mention just two: work on general decision-theoretic methods of sensor management (for example by Durrant-Whyte and co-workers, who take a Bayesian decision-theoretic stance [5, 13]) and work on quantitative task-oriented measures of visual quality (for example the control-theoretic framework for visual servoing described by Espiau, Chaumette and Rives [8]).

# 3 Viewpoint Assessment

The heart of the camera placement planner is the viewpoint assessment routine. This takes a partially specified hypothetical camera placement, completes the specification, and then runs a series of independent tests on the placement and its associated robot pose. Each test considers a single distinct aspect of the placement and returns a scalar assessment of its prospective quality from that point of view. The final overall assessment is a heuristically weighted sum of the individual test results. When possible, the individual assessments are based on objective, physically-based quality metrics, although in practice a significant heuristic component is unavoidable.

As an aid to weighting, test values are normalized to be "$\chi^2$-like" in the sense that zero indicates an ideal situation and values much greater than one a marginal one. Hard constraint violations are encoded as essentially infinite test results, but where possible we also include a term proportional to the degree of constraint violation to guide search methods towards the feasible region.

For speed, the tests are arranged roughly in order of computational cost and selectivity, and candidates whose accrued assessment becomes unacceptably large are discarded immediately. This is particularly important in the initial phases of search, when many inaccessible or otherwise unacceptable candidates are tried before the more promising regions of search space are discovered. The current sequence of tests is as follows:

1) Robot kinematics comes first as many other tests depend on it and can not be run if it fails. The assessor supports search in task, end-effector or joint space and provides forward and inverse kinematics to fill in the missing joint and pose information[1]. For inverse kinematics, all of the possible solutions are assessed and the best is chosen. Most of the tests do not depend on the kinematics solution in any case. Note that our 3R spherically-wristed SCEMI robots have relatively simple closed-form kinematics.

2) The camera position is assessed with respect to the environment. This is currently just a quick check against a list of rough workspace bounds to eliminate placements with the camera under the work surface or behind a wall, without the cost of full interference detection (which comes later).

3) The camera position is assessed with respect to the task. A task is characterized by a frame defining a task origin and coordinate system, a polyhedral volume that must be visible, and a set of viewing position and angle bounds and preferences. This test assesses the viewing angle and position of the camera. Simple types of task self-occlusion constraint can be encoded, but a more specialized assessor would be required for tasks with complicated self-occlusion geometry (*c.f.* [18]).

4) The task is assessed with respect to the camera. This is by far the most complex test and the only one to use camera parameters. The image that would be seen by the camera is predicted and assessed. To save the effort of writing a 3D hidden surface graphics library, the current prediction mechanism is heuristic rather than exact. We only project approximate bounding hulls of objects[2], and we use heuristic depth sorting rather than full visible surface prediction when evaluating occlusion and clutter.

4.1) After a quick test to verify that the camera is pointed in roughly the right direction, the task region is projected and its silhouette is clipped against the image borders. A penalty is charged for the fractional area that is clipped.

4.2)

The closest and farthest points of the task region

[1] In fact, this flexibility is not currently needed as all of the existing search routines use inverse task space kinematics.

[2] Most of the objects in our environment models are assembled from convex primitives in any case.

are determined and used for focus and resolution assessment. Lack of focus is penalized according to the area of the blurred image of a point[3], and lack of resolution according to the squared resolution deficit $\left(\frac{\text{desired resolution}}{\text{actual resolution}} - 1\right)^2$.

4.3) For each obstacle in the environment that may occlude or clutter the task, its image is predicted and the degree of its impingement on the task image is assessed. Obstacles lying in front of the task cause occlusion, penalized according to the percentage of the task image area that is occluded. Obstacles lying behind or close to the task image cause clutter. Although it does not actually obstruct the view of the task, clutter also needs to be penalized because it makes visual routines such as segmentation much more difficult. Underlying clutter is penalized according to the underlayed percentage of the task area, and nearby clutter according to the inverse image distance between it and the task region image.

5) The mobility of the camera robot in the chosen pose is assessed, and poses very close to kinematic singularities or joint limits are penalized. Without this softening of the constraints the optimal viewpoint would very often be hard against a kinematic or joint limit. This would make life difficult for local viewpoint search techniques, for our local path planner, and not least for real robot controllers.

6) Full robot-environment interference detection comes last owing to its very high cost (at least 80–90% of the run time of the assessor). Our current system does not allow us to evaluate the degree of penetration so we simply return a yes/no answer.

In practice the assessor performs well. Most importantly, the viewpoints it selects seem to correspond very well with the ones a human might choose, although the absolute values of its assessments are probably not an infallible index to "absolute view quality" as judged by a human. The basic approach of assessing view quality by image prediction followed by quantitative predicted-image-based quality metrics seems to be effective, and has no doubt contributed to the robustness of the evaluations: despite its many parameters the current assessor has never needed any significant parameter tuning.

However, several aspects of the implementation could be improved. A more precise image prediction mechanism would allow much more refined assessments of probable image quality, as well as providing support for important visual routines such as tracking, model-image matching and so forth. The model of task geometry should be extended to consider more complex types of task self-occlusion. It would also be useful to include a simple model of lighting (although exact light levels are notoriously hard to predict).

# 4 Viewpoint Search

Given the viewpoint assessment function, we need to optimize it to find the best available viewpoint for the task. Several features of the assessment function combine to make it difficult to optimize:

- It is relatively expensive owing to the large amount of geometric computation required for kinematics, image prediction, and particularly collision detection. The search method must make the most of each function evaluation, even if this involves a substantial amount of subsidiary computation.
- It is highly nonlinear with sudden 'cliffs' where hard constraints switch on and relatively shallow local minima lying hard up against constraint boundaries. The search method must be robust as this type of behaviour will almost certainly cause problems for any method that makes strong smoothness assumptions.
- There are often several distinct feasible regions because occlusion and accessibility constraints frequently cut the workspace in two. For reliability, the search method must be as global as possible: local viewpoint searches *do* in practice get caught in local minima.
- The search dimension is potentially quite high (*e.g.*, six for a search over the full space of 3D camera poses or robot configurations). This is sufficiently high to preclude simple ground covering methods such as grid search. Any search method needs to be well focused, and if a global method is used the search dimension probably needs to be reduced.

On balance we feel that a well-directed global search over a reduced-dimension space is most appropriate. In fact, even for local methods we have preferred to restrict the search to the 3D space of accessible upright camera poses directed towards the task centre, in order to guarantee upright, centred images. Global searches of this 3D space turn out to be quite feasible, and as far as we can tell relatively few good viewpoints are lost by restricting attention to it.

A further potential advantage of global methods is that if there are several minima of similar quality, they can output all of them and leave the choice to the user. Such backtracking might substantially improve the robustness of a system based on the viewpoint planner.

In our implementation we have hedged our bets by providing a choice of three different optimization routines, which can be used singly or in combination:

[3] The diameter of this 'defocus disk' for a point at depth $d$ is $\frac{a}{p}\left|\frac{d^{-1}-D^{-1}}{f^{-1}-D^{-1}}\right|$ pixels in the geometric optics limit, where $a$ is the camera aperture, $p$ is the pixel size, $f$ is the focal length, and $D$ is the in-focus depth.

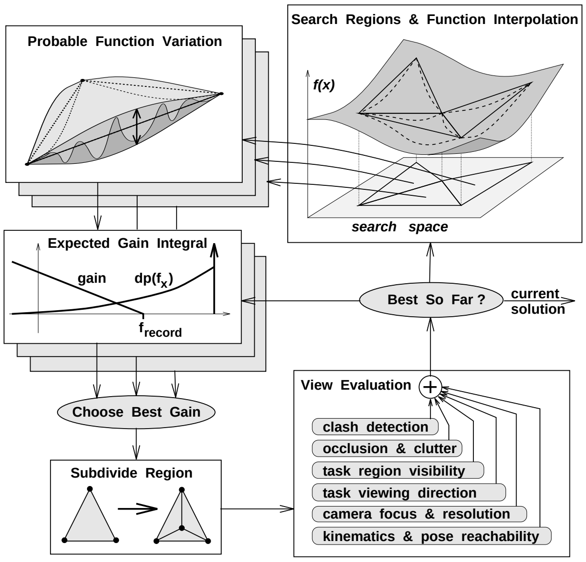


*Figure 2*: The maximum expected gain search loop.

(*i*) maximum expected gain (a quasi-optimal global region-based search technique); (*ii*) simulated annealing; and (*iii*) Powell's method (a classical smooth function minimization technique). We will concentrate on the global method as we feel that it has the best overall performance, with brief notes on the others for comparison.

## 4.1 Maximum Expected Gain Search

To meet the above requirements we have developed a new approach to global function optimization called **maximum expected gain** search. This is a region-based recursive decomposition technique that uses decision theory and subjective-probabilistic models of function behaviour to choose a quasi-optimal location for each function sample. Its key advantage is that it allows knowledge or expectations about likely function behaviour to be encoded in a form directly adapted to search control, so that search effort can be focused very precisely on the regions where it is likely to prove most profitable. The basic search loop is illustrated in fig. 2. More details will appear in [20].

### 4.1.1 General Approach

To understand the method, imagine that you're trying to minimize a function $f(\mathbf{x})$ over some domain, and that you've already evaluated it at several points $\mathbf{x}_1, \ldots, \mathbf{x}_n$. What's the best place to try next? — In general it's impossible to be sure and one has to guess. The only information relevant to this guessing is the values one thinks the function is *likely* to have at different points. These are constrained by general prior information about function behaviour such as continuity, smoothness, and boundedness, and (via continuity) by the known function values at existing nearby sample points.

Taking a Bayesian approach, the most direct way to encode such information is as a **subjective probability distribution** for the function value $f_\mathbf{x}$ at $\mathbf{x}$ given the samples $f_{\mathbf{x}_1}, \ldots, f_{\mathbf{x}_n}$ and the prior information:

$$\mathrm{dp}(f_\mathbf{x}) \;=\; \mathrm{dp}(f_\mathbf{x} | \, \mathbf{x}, f_{\mathbf{x}_1}, \ldots, f_{\mathbf{x}_n}, \text{prior-information})$$

This is a powerful technique because *any* information relevant to optimization can be encoded. Bounds on the function and its derivatives limit the support of the distributions, while softer estimates of typical behaviour control their shape. As a general rule, near an existing sample point $\mathbf{x}_i$ the subjective probabilities will be sharply peaked near $f_{\mathbf{x}_i}$, while further away they will grow increasingly diffuse as the function value becomes more and more uncertain. The amount of spread will depend principally on the distance moved and the expected smoothness of the function.

Of course, such subjective distributions have no existence in objective reality: they simply reflect our ignorance or laziness in refusing to evaluate the function exactly. For search guidance, they are useful only to the extent that they are both (*i*) tractable and (*ii*) accurate estimates of the true function behaviour. Their exact forms are entirely subjective and can only be specified heuristically according to intuition or computational convenience.

Now we can answer our question about where to look next. In general, points that have a high probability of having a good function value will be good places to look. But we can be much more specific. At any given point in the search there will be some *best currently known function value* or **current record** $f_{\text{record}}$. The aim of optimization is to find the best possible record in the fewest possible function evaluations. Evaluations that do not improve the current record are in some sense 'wasted'. This suggests that the best measure of the prospective value of a trial point is the **expected improvement in the current record** anticipated from evaluating the function at that point. We will call this improvement the **gain** and advocate a search régime that greedily selects the point with the **maximum expected gain** for evaluation at each step.

The expected gain is given by an expectation value integral:

$$\langle \text{gain}_\mathbf{x} \rangle \;=\; \int_{-\infty}^{f_{\text{record}}} (f_{\text{record}} - f) \; \mathrm{dp}(f_\mathbf{x} | \, \mathbf{x}, \ldots)$$

Note that the result depends only on the part of the distribution below the current record. As the record improves, the integral gets pushed further and further out into the tail of the distribution. For this reason, we will concentrate mainly on capturing the form of the asymptotic tails of the subjective function value distributions.

The current record and the subjective value distributions both evolve as samples are added, so the implementation will need to keep track of the changes in the expected gains. Although this can represent a significant overhead, in many practical cases it is amply justified by a dramatic increase in search efficiency because the *absolutely most promising point* (within the limits of the current information and the subjective models) is investigated at each step. As the record improves, the expected gain of any given point decreases and relatively unpromising points become less and less likely to contribute: hence the search is well focused. On the other hand, once the most promising regions have been thoroughly investigated, the method turns increasingly towards verifying that apparently less promising but less well investigated regions do not contain large unexpected fluctuations that create islands of good function values: hence the search is complete.

As it stands, the above method is not practical: (*i*) it is hard to tie down the forms of the subjective distributions, which grow increasingly complex as more samples are added; (*ii*) it is difficult to optimize the gain integral globally over the continuum of values $\mathbf{x}$ (after all, this is the sort of problem we were trying to solve in the first place!).

Both of these problems can be tackled by making a major simplifying assumption. We will suppose that: (*i*) at each search step the search domain can be subdivided into a set of non-overlapping **regions of influence**, each 'under the influence of' an associated set of **control points** chosen from among the existing function samples; and (*ii*) the function value distributions in each region can be approximated by ones depending *only* on the region's controlling samples. In other words, for all $\mathbf{x}$ in the region controlled by samples $\mathbf{y}, \ldots, \mathbf{z}$:

$$\mathrm{dp}(f_{\mathbf{x}}) \approx \mathrm{dp}(f_{\mathbf{x}} | \, \mathbf{x}, f_{\mathbf{y}}, \ldots, f_{\mathbf{z}}, \text{prior-information})$$

This should be a reasonable subjective model provided the sets of control points can be chosen to be sufficiently simple to make the model tractable, while still containing the samples that have subjectively significant influences on the function values in each region. Usually, the control points will be (a subset of) the samples nearest to the region.

This assumption simplifies the problem enormously because the forms of the function value distributions can be specified parametrically once and for all for a generic influence region, and then instantiated for particular regions on the fly. Even more significantly, given a sufficiently simple form for the value distributions, the location and gain of the most promising point in the generic region can be evaluated (or more commonly estimated) once and for all as a function of the region parameters and the current record. This reduces the difficult problem of ranking an infinite number of potential sample points to the much simpler one of ranking a finite number of regions according to the *best potential sample point* each contains.

#### 4.1.2 Overview of Implementation

At this point we can sketch the implementation. The central data structure is a priority queue of records describing influence regions. The user must supply a region evaluation routine that determines a (quasi-) optimal sample location in the region and its expected gain with respect to the current record, presumably by using the subjective function value model outlined above.

For efficiency it is essential to postpone sorting and evaluation work for as long as possible, so an optimal, lazy queue implementation is required. We have used a version of Fredman and Sedgewick's elegant pairing heap queue [10]. This is ideal for our application as it is efficient (amortized $\mathcal{O}(\log n)$ updates), extremely simple to implement, and naturally lazy in that the minimum possible amount of sorting is always performed at the last possible moment.

To reduce the evaluation workload, regions are ranked in the queue according to *possibly out-of-date* expected gains. At each search step, regions are repeatedly pulled off the head of the queue, re-evaluated with respect to the current record, and reinserted, until a region with an up-to-date expected gain is found. This region has the best expected gain in the queue even with respect to the *current* record because gains can only decrease as the record improves. The function is evaluated at the chosen location(s) for the selected region and the region decomposition is updated, with deleted regions being deleted from the queue and newly created ones evaluated and inserted. Often, the update will simply be a fixed subdivision of the selected region.

The process continues until a sufficiently good sample has been found or a sufficiently thorough search has been made. The running gain provides a rough but quantitative guide to the improvement further search is likely to yield. Expected gains can also be replaced by 'expected profits' (*i.e.* expected gain less the cost of a further search step), in which case search can continue until the expected profit is zero.

#### 4.1.3 Region Models

To implement the method, forms must be chosen for the influence regions and the subjective distributions.

We have investigated two main types of region decomposition: $2^d$-trees and Delaunay triangulations. The Delaunay model allows slightly better search focusing and is perhaps more refined, but the $2^d$-tree model is less complicated, numerically more robust and has a much lower overhead in geometric computation. Both models are strictly 'local', with function samples at each region vertex acting as the controlling vertices of the region. It would also be interesting to investigate slightly less local influence models such as $k$-nearest-neighbour regions.

In the $2^d$-tree model the search arena is subdivided into a set of non-overlapping hypercubes that form the leaves of a $2^d$-tree (*e.g.* an oct-tree for our 3D search problem)[4]. There is a function sample at every cube vertex. At each search step the most promising region is removed from the queue and bisected along each coordinate to produce $2^d$ child cubes, which replace the parent in the queue. The model does not quite fit the above general paradigm in that several function samples may be created at each step, but the only change needed is to hypothesize a subjective distribution for the *best* value among all the new function samples when estimating the expected gain integrals. The fact that a single subdivision may create as many as $3^d - 2^d$ new vertices (and hence function samples) is a potential source of inefficiency, however in practice the amount of oversampling is usually small: away from the boundary there are never more than $(3/2)^d - 1$ samples per region, and if the subdivision is locally fairly uniform (as is often the case) the true number is usually much closer to 1.

In simplex-based models like the Delaunay one, the regions form a simplicial complex with a function sample at each vertex. At each step, the most promising region is selected, a single new function sample is created at a chosen location inside it, and the triangulation is updated to include the new vertex. To ensure that the simplices remain fairly uniform (and hence provide good control over the function values in their interior), the update process must have a strong tendency to break up large facets, especially edges. Delaunay triangulations [3] are interesting in this respect because their geometric properties ensure that the simplices are locally as well-shaped as possible. To preserve the Delaunay property, the amortized $\mathcal{O}(\log n)$ incremental Delaunay vertex insertion process must retriangulate the 'conflict neighbourhood' of each newly inserted vertex.

The principal advantages of the Delaunay model over the $2^d$-tree one are:

- Only one new sample is created at each step, so it is potentially very economical in function evaluations.
- The sample can be located anywhere in the simplex, so there is maximal scope for effective sample placement and the method can be used with pre-existing samples, ones created by local search strategies, *etc*.
- The initial search region can be an arbitrary convex polytope rather than simply a hypercube, so less time is spent locating the feasible region during the initial phases of search.

The disadvantages of the Delaunay model are:

- It is geometrically more complex and significantly harder to implement.
- The computational cost of the retriangulation process can easily negate any extra efficiency the technique may have, so it is probably only suitable for fairly expensive evaluation functions.
- Round-off error tends to make retriangulation rather delicate, particularly in high dimensions. In practice, the sample placement heuristic must be chosen quite carefully to avoid numerical problems. It is also essential to make the code robust to vertex insertions on simplex facets (including external ones).

In the current viewpoint planner we have preferred the $2^d$-tree for its simplicity and robustness, but we have experimented extensively with both and find their performance to be comparable for viewpoint assessment. With a more expensive evaluation function the Delaunay model would probably have the edge.

#### 4.1.4 Models of Function Variation

The final component needed for the implementation is a plausible model of probable function behaviour for each region. The precise form this takes is necessarily very subjective, but to be useful for search focusing it must reflect the likely influence of existing nearby function samples, region size and shape, and prior constraints such as positivity or smoothness on function values in the region. It must also be sufficiently tractable to allow the most promising sample point to be estimated, and the expected gain integral to be evaluated.

The models we have used are based on "probabilistic Lipschitz bounds" on function variation. Classical Lipschitz bounds limit the maximum change in a function $f(\cdot)$ between any two points of a region to the form $|f(\mathbf{x}) - f(\mathbf{y})| < K\|\mathbf{x} - \mathbf{y}\|^\nu$, where $K$ and $\nu$ are positive constants and $\|\cdot\|$ is some positive metric. Here, we will view this formula as a heuristic subjective estimate of the *probable* amount of variation in a region of a given size rather than as a strict bound. Specifically, we will suppose that the function variation $\Delta f$ across a region of size $\Delta\mathbf{x}$ has a subjective probability distri-

[4] To reduce the amount of pointer chasing, we do not not actually store the tree structure. Vertices are accessed by hashing their coordinates.

bution of the form

$$\mathrm{dp}(\Delta f) \;\approx\; \rho(\mu\,\Delta f / \|\Delta \mathbf{x}\|^{\nu})$$

where $\mu$ and $\nu$ are constants characteristic of $f(\cdot)$ and $\rho(\cdot)$ is some fixed probability distribution. This is nothing more than a simple heuristic model aimed at capturing the intuition that large variations become less likely as the region gets smaller.

Without further information, the choice of $\rho(\cdot)$ is necessarily rather arbitrary and must be made experimentally. This is unfortunate as the key parameters for search focusing are the exponent $\nu$ and the asymptotic form of the tail of $\rho(\cdot)$, since it is these that govern how quickly small regions with relatively poor function samples will be discounted from the search. For simplicity we have based $\rho(\cdot)$ on an exponential distribution in most of our experiments, although a power law or a Gaussian might have been equally plausible choices.

However none of the above distributions can be used without modification. They all have tails stretching to $-\infty$, whereas we know that our viewpoint assessment function is bounded below by zero. In fact, across particular regions it is often possible to find a strictly positive lower bound for the function. When available, such bound information can be extremely valuable: it puts strong limits on the subjective function value distributions in the region, and hence allows the prospective gain to be estimated much more tightly, and it allows regions whose lower bound is greater than the current record to be rejected outright. In our case, lower bounds turn out to be especially important for rejecting regions that contain no feasible viewpoints, but they are also used to discount feasible regions in relatively poor areas of the search space. Again, the method used to incorporate the bound information into the subjective distributions is essentially arbitrary. At present we simply truncate the distributions at the bound value, although we have also experimented with smoother types of cutoff.

Putting all of the above together, we can outline the models adopted for sample point choice and expected gain evaluation in the viewpoint planner. Instead of applying the Lipschitz model directly to the viewpoint assessment function, we apply it to the *deviation* of the function from a fiducial interpolation passing through the existing samples at the region vertices. In the Delaunay case, this is simply the unique linear interpolant of the function through the simplex vertices, while in the $2^d$-tree case it is the corresponding $d^{th}$-order tensor-product spline[5].

[5] If the cube coordinates are normalized to $[0,1]^d$ and $\phi(x,q)$ is $x$ for $q=1$ and $1-x$ for $q=0$, this is $\sum f(\mathbf{q}) \cdot \prod_{j=1}^{d} \phi(x_j, q_j)$, where the sum is over the $2^d$ cube vertices $q_i = \{0,1\}, i=1,\ldots,d$.

In the Lipschitz model, the expected deviation scale $|\Delta f|$ increases monotonically and isotropically as the point moves away from any existing sample vertex. Hence, the point of the region with the maximum scope for deviation is always the point maximally distant from all the vertices. In the case of the $2^d$-tree model this is the centre of the cube, while in the Delaunay case it is the circumcentre of the simplex (the centre of the unique sphere passing through the vertices) — provided this lies within the simplex.

For finite records, this central point is usually *not* the point with the maximum expected gain: some point further 'downhill' (towards the best sample vertex) is usually better. However in general the best point turns out to be quite hard to estimate. Since the asymptotic search complexity is governed mainly by the asymptotic rate at which small regions far above the current record are discounted, we have concentrated on getting the model right in this limit. And in this limit the central point *is* optimal, because the large deviation required to better the record completely swamps any gradient term in the fiducial interpolation.

Hence, for simplicity and (at least) asymptotic efficiency, we always place the next function sample at the centre of a region. The expected gain of the region is then estimated as the expected gain at this point, based on the Lipschitz deviation model with an exponential characteristic distribution truncated at the best lower bound we have for the function on the region.

### 4.1.5 Discussion

In practice the maximum expected gain viewpoint search method works very well. It runs as quickly as the local methods we have implemented and seems to converge reliably to the globally optimal viewpoint. Despite the very heuristic nature of the probabilistic models used to derive it, it seems robust and easy to tune. (For example it performs well with any Lipschitz exponent $\nu$ between about 0.5 and 1.0 and any scale factor $\mu$ within a factor of 2–4 of the optimum). Of course, it *can* be badly tuned, but in such cases it either runs slowly but reliably (like a global grid search), or quickly converges to a (possibly local) minimum like a local search technique. The distinction between global and local search may be muddied, but at least the failure modes are relatively benign.

Perhaps the main disadvantage of the method is its relative complexity. Although the underlying ideas are simple there are quite a few pieces to implement. Another problem is termination: as with other multidimensional search methods it is difficult to know when to stop searching. In theory the expected gains can be used as a guide, but in practice they decay quite slowly and irregularly[6], so that simple thresholding is not a reliable stopping criterion. Currently we just search for a fixed number of samples — hardly an optimal policy. The Lipschitz scaling model for expected function variation could also be improved to allow slightly more control over the search profile, so that (for example) the search could be practically global down to a certain resolution, and then become rapidly local after that.

### 4.2 Local Search Methods

For comparison with the above global method we have implemented two local search techniques: Powell's direction set method and simplex-based simulated annealing, both derived from (but not identical to) routines from Numerical Recipes [16].

Powell's method (as implemented in [16]) is a quasi-quadratically convergent method designed for smooth functions, that does not require function derivatives. It works by repeated line searches along an evolving set of approximately conjugate directions. We had hoped to use it to 'polish' coarse solutions found by the global search, but its performance has proved disappointing in our application: it was not designed for functions with sudden steps and tends to 'stick' on constraint walls rather than sliding along them. By softening the constraints slightly (*e.g.* by penalizing poses close to the robot joint limits) it can be made to converge, but it is neither so reliable nor so fast as the global technique.

By contrast, the simplex-based annealing method makes very few assumptions about function smoothness. It works by applying the simulated annealing acceptance criterion to trials, each of which flips, stretches or shrinks a simplex of trial solutions. Annealing methods are often considered to be quasi-global, but this depends entirely on the annealing schedule. In our case, for realistic schedules, the method is perforce rather local. Certainly, if it is started in a suboptimal feasible region there is no significant chance of its tunnelling through an infeasible barrier to a better feasible one.

In practice the annealing method performs very respectably for the most part, although it *does* sometimes get caught in local minima. Apart from these cases, it can be tuned to give similar results to the global method for similar amounts of computation, so the simplex-based trial generator seems to be making quite effective use of function evaluations (a traditional weak point of annealing methods). On the whole we tend to prefer the global method because of its greater reliability, but the performance and relative simplicity of the annealing method make it an attractive second choice.

[6] For any one region the expected gain decays monotonically, but new regions are created and destroyed all the time.

## 5 Tour Planning

The tour planner is responsible for creating a feasible overall plan for a sequence of visual tasks, starting from an unordered set of viewpoints (camera robot poses) produced by the viewpoint planner. It must select an efficient ordering of the tasks and ensure that collision-free paths can be found to move the camera robot through the sequence of viewing positions. Nothing about it is specific to the viewpoint problem: it simply plans a low-cost tour visiting an arbitrary list of $n$ 'sites' (robot poses).

An underlying path planner is needed to plan segments of the tour. We are currently using a potential field based method [9, 14] that is fast but relatively unreliable. Planning is intrinsically costly and is likely to dominate the run time relative to simpler operations such as graph manipulation in any practical implementation, so we will measure the cost of tour planning by counting planner calls. The minimum possible cost is $\mathcal{O}(n)$ as the tour visits $n$ sites.

The classical tour planning problem is the Travelling Salesman problem: find a minimal cost tour that visits each vertex of a completely connected graph exactly once. This is NP hard as an exact problem, and hence intractable even for moderate $n$. However for many graphs (notably those with costs based on geometric proximity) good approximate solutions can be found reliably in $\mathcal{O}(n^2)$ using simulated annealing [12, 15, 16].

On the other hand, a direct travelling salesman attack on our problem would require a complete matrix of edge (robot path) costs between each pair of sites in the tour-graph. These would have to be produced by $\mathcal{O}(n^2)$ calls to the underlying path planner, which would make the method intolerably slow relative to the $\mathcal{O}(n)$ output. Although it is not possible to avoid $\mathcal{O}(n^2)$ calls in the worst case, we would like to find a method closer to '$\mathcal{O}(n)$ in favourable cases', even if this involves a considerable cost in graph manipulations.

One simple way to achieve this is to construct candidate tours using *estimated* path costs, and then to attempt to verify them using the path planner. If the planner fails an alternative tour has to be planned with improved cost estimates, but 'in favourable cases' the method will hopefully converge within a few cycles.

Another point is that although the tour must visit each site at least once, we do not really want to insist on its visiting each site *exactly* once. In fact, given the unreliable nature of our path planner this would often make the problem insoluble: a site connected to only one other could have no tour passing through it. The tour planner should fail only when it is impossible to connect the site-graph using the local path planner, although in difficult cases it may generate long tours with many detours around missing graph edges.

This suggests that we should base our candidate tours on estimated inter-site *route* costs for travel via any route through the graph, rather than simply on estimated *edge* costs for direct (path planner) paths between sites. Such route costs can be evaluated in $\mathcal{O}(n^2 \log n)$ using Dijkstra's algorithm [17], and updated in the same worst case time (but in practice often much more quickly) using an incremental version of this algorithm. The use of route costs also allows us to plan with respect to a background route map (perhaps built up over previous calls to the planner) and hence to speed up planning and enhance reliability.

Putting all of this together, the complete tour planning algorithm goes as follows:

0) Add the tour sites to the route map graph.

1) For each pair of sites in the graph, estimate the edge (path planner path) cost between them as either: (*i*) the true path cost if a path has been planned ($\infty$ if the planner failed); or (*ii*) an estimate of the probable path cost based on 'straight line distance in C-space' or some similar lower bound on path cost, inflated by some small factor $k$.

2) Find the estimated minimal route cost between each pair of sites in the graph using the Dijkstra algorithm (or the incremental one if only a few edge costs have changed).

3) Find a trial tour of the sites based on the estimated route costs, using the simulated annealing travelling salesman algorithm.

4) If the tour cost is $\infty$, fail. (The tour contains some edge for which the path planner failed, which indicates that the corresponding state is unreachable).

5) For each edge of the tour for which a plan has not yet been attempted, plan a path. If any plan fails or its cost is higher than the estimate, abandon the trial tour and go back to step 1.

6) Output the current tour and exit.

An important enhancement to step 5 is that if the graph has not yet been connected we only attempt to plan edges leading to currently unconnected sites: tour feasibility (graph connection) is initially a higher priority than tour optimality.

The only heuristic parameter in the method is the cost estimate inflation factor $k$ used in step 1. It trades route optimality for planning effort. In essence, routes (or at least those selected for tours) are replanned until the method *proves* that the ones it has are within a factor $k$ of the shortest possible ones in the completely path-planned graph. With $k = 1$ the behaviour is akin

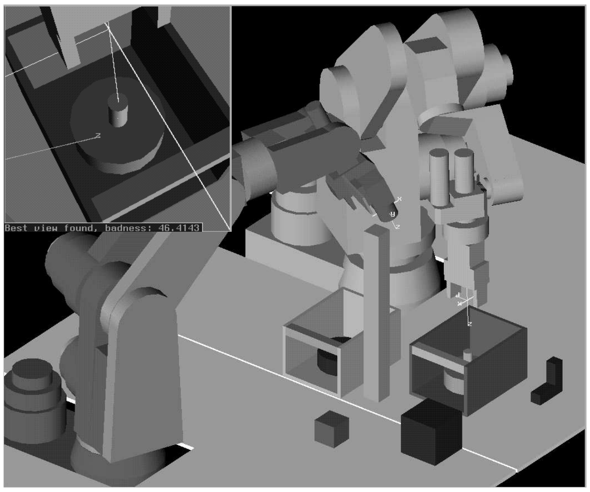


*Figure 3*: A planned camera placement, and (inset) the corresponding predicted image. The task was to get a clear view of the central peg in the right hand box.

to that of $A^*$ and guaranteed shortest routes are found, while with larger $k$ there is a tendency to reuse existing routes, even if some untried ones may be slightly shorter. The final tour is within a factor $k\delta$ of the true minimal one, where $\delta$ is the cost overrun in the travelling salesman solution (difficult to estimate, but often no more than a few percent).

In practice the algorithm works well and usually succeeds or proves failure within a few $n$ planner calls, despite the unreliability of the local path planner. The tours it generates seem subjectively good, although they often have to make detours around edges whose path plans have failed. If there are a few unreachable sites this is usually detected within a few $n$ planner calls, although the worst case run time is certainly $\mathcal{O}(n^2)$, for example when failing exhaustively to connect two disconnected $\mathcal{O}(n)$-site cliques.

It must also be emphasized that the '$\mathcal{O}(n)$ favourable case' run time has been bought at a high worst-case cost in graph operations: at each of the $\mathcal{O}(n^2)$ worst-case iterations there can be an $\mathcal{O}(n^2 \log n)$ route cost update and an $\mathcal{O}(n^2)$ travelling salesman problem. However order of magnitude run time estimates suggest that in our case the trade-off is amply justified out to $n \approx 10^3$, and probably far beyond.

One potential weakness of the current implementation is that there is no feedback from the tour planner to the choice of viewpoints. If a self-consistent but inaccessible viewpoint is chosen, the entire task plan will fail. Feedback would presumably help to reduce the problem, but we have not included any as it was not clear what form it should take.

## 6 Implementation & Experiments

The current version of our system contains about 10,000 lines of fairly modular C code and runs on Silicon Graphics hardware under the robot modeller ACT [14]. A typical application is shown in fig. 1. The problem is loosely based on a routine inspection task in the maintenance of steam generators in nuclear power plants. The area around the bases of the columns must be examined for damage or detritus. The task is formulated as a request to view a series of overlapping disk-shaped areas covering the region. (For clarity we have only included areas hidden behind columns, since these are the hardest to get a clear view of). The current robot can not quite reach the ideal viewpoints for the most difficult areas, so a compromise solution is found. Some areas can be partially seen from either side of their column, so the search space has several local minima. The planned sequence of views is shown in the lower panel. In each case the viewpoint was found by a global search over the entire reachable space of upright camera poses directed towards the centre of the area to be viewed. The tour planner has reordered the views so that left-looking ones precede right-looking ones, as the left-right reorientation is relatively expensive for the robot. On an SGI R4000 machine each viewpoint search takes about 4 seconds. Planning a robot path between viewpoints takes 1–2 seconds, so the total run time is about 5–10 seconds per viewpoint.

Another problem is illustrated in fig. 3. The task was simply to get a clear view of the central peg in the right hand box. Again occlusion cuts the search space in two so the problem has several deep local minima. The final pose is very near the limit of the robot's reach. Fig. 4 shows a grasp approach being visually servoed from a viewpoint chosen by the first version of the planner.

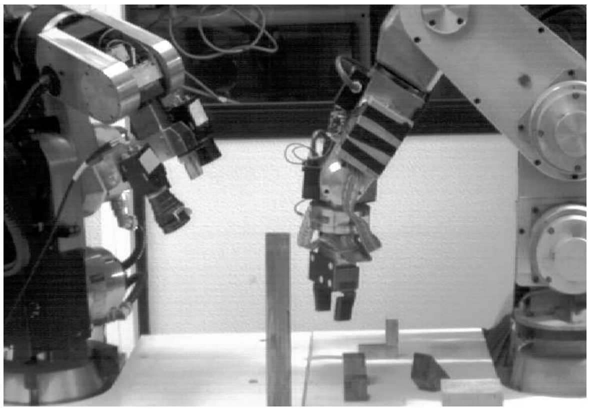

*Figure 4*: Visually servoing a grasp approach from a planned viewpoint.

## 7 Discussion

Overall we are pleased with the viewpoint planner's performance. Despite the large number of parameters involved, the viewpoint assessment and search routines have proved to run reliably with very little tuning. The global search seems robust and surprisingly efficient. Most importantly, the viewpoints selected really do seem to correspond to ones a person might choose as the subjectively "best available camera placements".

The tour planner also performs reasonably well, within the limits of the underlying path planner. For the simple problems we have tried, either a tour is found or one of the sites is shown to be unreachable (according to the path planner), within a few times $n$ (the number of sites) planner calls. However even for 'simple' problems the current local path planner fails fairly often, so that tours frequently need to make long detours around the gaps left by its failures.

As with any large system, there are many minor improvements that could be made, and several major limitations. As it stands, the whole system is rather too static and model based. Real images are never used. Plans are based entirely on internal geometric models that may not be an accurate reflection of the true geometric environment, let alone the true visual one. No attempt is made to model important visual effects such as shadowing and specularities. Any real system is likely to need the ability to compensate for such unmodelled effects by making on-line adjustments to a planned viewpoint, so some means of feeding visual information back into the viewpoint selection process would be desirable.

Similarly, there is currently no feedback from the tour planner to the viewpoint selection process: a viewpoint is chosen once and for all for each task, and then the path planner either succeeds or fails to find paths connecting these viewpoints. The whole process may fail when there was a perfectly acceptable alternative viewpoint for which it would have succeeded.

Another major omission is the ability to deal with time constraints and moving targets. This would be useful for visual servoing and process monitoring applications. It probably lies more in the domain of active vision than visual planning, however there is a real (and difficult) motion planning problem too: generate a task plan that optimizes a complex continuous criterion (sustained view quality), in a workspace encumbered with hard constraints (joint limits, collisions, time constraints…). The difficulties are: (*i*) the size of the search space; (*ii*) the fact that there may be points at which the camera has to lose sight of the task in order to pass an obstacle or reorient itself, so that the search has a mixed continuous/discrete character.

Finally, the current user interface needs improvement. In particular, it would be useful to have a more efficient means of specifying repetitive visual tasks such as seam or surface inspection. At present these must be discretized and programmed by hand.

As far as future work is concerned, our immediate priorities for the system are improving the user interface (especially the task-specification aspects), the integration with visual servoing, and the reliability of the path planner. In the longer term we are interested in incorporating visual feedback into the plans and in planning sustained visual tasks such as visual tracking in cluttered environments.

# 8 Summary

We have described an automated planner for visual inspection and monitoring tasks using a robot mounted CCD camera. It is optimized for intervention-style applications with large potentially cluttered workspaces, and therefore needs to take account of workspace constraints on the camera-carrying robot as well as task constraints and more traditional visual indices such as field of view, focus, resolution and occlusion. It is capable of finding heuristically near-optimal viewing positions for a series of visual tasks, and of ordering them and planning robot paths between them to produce a complete, near-minimal-cost overall task plan for the sequence.

The guiding principle of the design was to aim for reliability by using quantitative, physically-based measures of quality wherever possible, and by using global search to ensure that possible solutions are not overlooked. As a means to this, we have developed a novel global function optimization technique based on decision theory and subjective models of function behaviour, that allows global viewpoint searches to run in the time usually required for local ones.

**References**


[1] O. Al-Chami. *Contribution à l'intégration Robotic-Vision en Manipulation Automatisée: Modélisation de la Tâche, Placement d'une Caméra Mobile et Localisation Fine d'Objet*. PhD thesis, LIFIA, Grenoble, 1994.

[2] O. Al-Chami and C. Laugier. Stratégie perceptive pour positionner une caméra. In *Proc. AFCET Reconnaisance des Formes et Intélligence Artificiel*, volume 1, pages 617–22, 1994.

[3] F. Aurenhammer. Voronoi diagrams — a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, 1991.

[4] C. Bard, C. Bellier, J. Troccaz, C. Laugier, B. Triggs, and G. Vercelli. Achieving dextrous grasping by integrating planning and vision based sensing. *Int. J. Robotics Research*. Accepted, to appear in 1995.

[5] A. Cameron and H. Durrant-Whyte. Optimal sensor placement. *Int. J. Robotics Research*, 9(3), 1990.

[6] C. K. Cowan and A. Bergman. Determining the camera and light source location for a visual task. In *IEEE Int. Conf. Robotics & Automation*, pages 509–14, 1989.

[7] C. K. Cowan and P. D. Kovesi. Automatic sensor placement from vision task requirements. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 10(3):407–16, May 1988.

[8] B. Espiau, P. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Trans. Roboics & Automation*, 8(3):313–26, June 1992.

[9] B. Faverjon and P. Tournassaud. A local based approach for path planning manipulators with a high number of degrees of freedom. In *IEEE Int. Conf. Robotics & Automation*, Raleigh, March 1987.

[10] D. Fredman and R. Sedgewick *et.al.* The pairing heap: A new form of self-adjusting heap. *Algorithmica*, 1:111–29, 1986.

[11] R. Horaud, F. Dornaika, C. Bard, and C. Laugier. Integrating grasp planning and visual servoing for automatic grasping. In *$4^{th}$ Int. Symposium on Experimental Robotics*, Stanford, 1995.

[12] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. *Science*, 220:671–80, 1983.

[13] J. M. Manyika and H. F. Durrant-Whyte. *Data Fusion and Sensor Management*. Ellis Horwood, 1994.

[14] E. Mazer, J. Troccaz, and *et.al.* ACT: A robot programming environment. In *IEEE Int. Conf. Robotics & Automation*, pages 1427–32, Sacramento, California, April 1991.

[15] R. H. J. M. Otten and L. P. P. P. van Ginneken. *The Annealing Algorithm*. Kluwer, Boston, 1989.

[16] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992. $2^{nd}$ edition.

[17] R. Sedgewick. *Algorithms*. Addison-Wesley, Reading, MA, 1988.

[18] K. Tarabanis and R. Y. Tsai. Sensor planning for robotic vision. In O. Khatib, J. Craig, and T. Lozano-Pérez, editors, *Robotics Research 2*. MIT Press, 1992.

[19] K. Tarabanis, R. Y. Tsai, and P. K. Allen. Automated sensor planning for robotic vision tasks. In *IEEE Int. Conf. Robotics & Automation*, pages 76–82, 1991.

[20] B. Triggs. Global optimization using subjective models of function behaviour. Technical report, INRIA Rhône-Alpes, Grenoble, France, 1995. (To appear).

[21] B. Triggs and C. Laugier. Automatic camera placement for robot vision tasks. In *IEEE Int. Conf. Robotics & Automation*, pages 1732–7, Nagoya, 1995.

[22] S. Yi, R. M. Haralick, and L. G. Shapiro. Automatic sensor and light source positioning for machine vision. In *10th Int. Conf. Pattern Recognition*, pages 55–9, June 1990.