Fitting Parameterized Three-Dimensional Models to Images

David G. Lowe
Computer Science Department
University of British Columbia
Vancouver, B.C., Canada V6T 1Z4
Email: lowe@cs.ubc.ca

Abstract

Model-based recognition and motion tracking depends upon the ability to solve for projection and model parameters that will best fit a 3-D model to matching 2-D image features. This paper extends current methods of parameter solving to handle objects with arbitrary curved surfaces and with any number of internal parameters representing articulations, variable dimensions, or surface deformations. Numerical stabilization methods are developed that take account of inherent inaccuracies in the image measurements and allow useful solutions to be determined even when there are fewer matches than unknown parameters. The Levenberg-Marquardt method is used to always ensure convergence of the solution. These techniques allow model-based vision to be used for a much wider class of problems than was possible with previous methods. Their application is demonstrated for tracking the motion of curved, parameterized objects.

This paper has been published in *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **13,** 5 (May 1991), pp. 441–450.

1 Introduction

Model-based vision allows prior knowledge of the shape and appearance of specific objects to be used during the process of visual interpretation. Reliable identifications can be made by identifying consistent partial matches between the models and features extracted from the image, thereby allowing the system to make inferences about the scene that go beyond what is explicitly available from the image. By providing this link between perception and prior knowledge of the components of the scene, model-based recognition is an essential component of most potential applications of vision.

One important component of model-based vision is the ability to solve for the values of all viewpoint and model parameters that will best fit a model to some matching image features. This is important because it allows some tentative initial matches to constrain the locations of other features of the model, and thereby generate new matches that can be used to verify or reject the initial interpretation. The reliability of this process and the final interpretation can be greatly improved by taking account of all available quantitative information to constrain the unknown parameters during the matching process. In addition, parameter determination is necessary for identifying object sub-categories, for interpreting images of articulated or flexible objects, and for robotic interaction with the objects.

In most cases, it is possible to solve for all unknown parameters for a 3-D model from matches to a single 2-D image. However, in some circumstances—such as when both the size and distance of the model is unknown—the accuracy of parameter determination can be substantially improved by simultaneously fitting the model to images taken from more than one viewpoint. The methods presented here can be used in either situation.

The locations of projected model features in an image are a non-linear function of the viewpoint and model parameters. Therefore, the solution is based on Newton's method of linearization and iteration to perform a least-squares minimization. This is augmented by a stabilization method that incorporates a prior model of the range of uncertainty in each parameter and estimates of the standard deviation of each image measurement. This allows useful approximate solutions to be obtained for problems that would otherwise be underdetermined or ill-conditioned. In addition, the Levenberg-Marquardt method is used to always force convergence of the solution to a local minimum. These techniques have all been implemented and tested as part of a system for model-based motion tracking, and they have been found to be reliable and efficient.

2 Previous approaches

Attempts to solve for viewpoint and model parameters date back to the work of Roberts [30]. Although his solution methods were specialized to certain classes of objects, such as rectangular blocks, Roberts clearly understood the value of quantitative parameter determination for making vision robust against missing and noisy data. Unfortunately, there were few attempts to build upon this work for many years following its initial publication.

In 1980, the author [19] presented a general technique for solving for viewpoint and model parameters using Newton's method for nonlinear least-squares minimization. Since that time the method has been used successfully in a number of applications, and it also provides the starting point for the work presented in this paper. The application of the method to robust model-based recognition has been described by Lowe [20, 21, 22], McIvor [26], and Worrall, Baker & Sullivan [34]. Bray [2] has applied the method to model-based motion tracking of rigid objects. Ishii *et al.* [14] describe the application of this work to the problem of tracking the orientation and location of a robot hand from a single view of LED targets mounted on the wrist. Their paper provides a detailed analysis that shows good accuracy and stability. Goldberg & Lowe [8] describe the application and testing of a number of more advanced numerical methods for this problem.

In recent years, there has been a considerable increase in the number of publications on parameter solving for model-based vision, with most of the work aimed at solving for viewpoint parameters of rigid objects. Liu *et al.* [18] and Kumar [15] have examined alternative iterative approaches to solving for the viewpoint parameters by separating the solution for rotations from those for translations. However, Kumar shows that this approach leads to much worse parameter estimates in the presence of noisy data. Therefore, he adopts a similar simultaneous minimization as is used in the work above. A quite different approach based on the use of elimination methods to provide the initial problem formulation has been proposed by Ponce and Kriegman [29]. This also uses Newton's method for the final parameter determination based on least-squares minimization.

Haralick *et al.* [11] have experimented with robust methods such as iterative reweighting in order to allow for outliers caused by incorrect matches. However, their results show that even one outlier among 20 correct matches leads to a large increase in expected error following reweighting. The alternative that is used in this paper is to provide a higher-level search process that considers other sets of matches when the first set fails to result in an accurate fit of the model.

2.1 The problem of multiple solutions

Much work has been published on characterizing the minimum amount of data needed to solve for the six viewpoint parameters (assuming a rigid object) and on solving for each of the multiple solutions that can occur when only this minimum data is available. Fischler and Bolles [6] show that up to four solutions will be present for the problem of matching 3 model points to 3 image points, and they give a procedure for identifying each of these solutions. A solution for the corresponding 4-point problem, which can also have multiple solutions under some circumstances, is given by Horaud *et al.* [12]. Huttenlocher and Ullman [13] show that the 3-point problem has a simple solution for orthographic projection, which is a sufficiently close approximation to perspective projection for some applications. They use the term "alignment" to refer to the solution for viewpoint parameters during the model fitting process. In the most valuable technique for many practical applications, Dhome *et al.* [4] give a method for determining all solutions to the problem of matching 3 model lines to 3 image lines. They show that this is particularly useful for generating starting positions for the iterative techniques used in this paper when there are multiple solutions.

This work on determining all possible exact solutions will no doubt be important for some specific vision applications, but it is probably not the best approach for practical parameter determination in general model-based vision. One problem with these methods is that they do not address the issue of ill-conditioning. Even if a problem has only one analytic solution, it will often be sufficiently ill-conditioned in practice to have a substantial number and range of solutions. Secondly, all these methods deal with specific properties of the six viewpoint parameters, and there is little likelihood that they can be extended to deal with an arbitrary number of internal model parameters. Finally, these methods fail to address the problem of what to do when the solution is underconstrained. The stabilization methods described in this paper allow an approximate solution to be obtained even when a problem is underconstrained, as will often be the case when models contain many parameters.

Possibly the most convincing reason for believing that it is not necessary to determine all possible solutions is the fact that human vision apparently also fails to do so. The well-known Necker cube illusion illustrates that human vision easily falls into a local minimum in the determination of viewpoint parameters, and seems unable to consider multiple solutions at one time. Rock [31], pp. 22ff summarizes the way in which human perception seems to always adopt one particular perception at any time even in the face of completely indeterminate continuous variables. The perception can suddenly change to a new stable position in the face of new information, which may come internally from other components of the visual system (attention) as well as from the external stimulus. This behavior is consistent with a stabilized minimization approach for determining the parameter values, in which the process can be initiated from new starting points as new information becomes available. The extremely good performance of human vision in most recognition problems, in spite of its potential for getting stuck in false local minima, indicates that local minima may not be a major problem when determining model parameters.

It is worth noting that the parameter solving problem is simplified when accurate 3-D image data is available (as from a scanning laser rangefinder), since this avoids some of the non-linearities resulting from projection. Examples of solutions to this problem are given by Faugeras & Hebert [5] and Grimson & Lozano-Pérez [10]. However, in this paper we restrict our attention to fitting 3-D models to 2-D image features.

3 Object and scene modeling

Most research in model-based vision has been based on models of simple polyhedral 3-D objects. While they are simple to work with, they are clearly inadequate for representing many real-world objects. Some research has been based on models built from certain classes of volumetric primitives, most notably generalized cylinders [1, 3] and superquadrics [27]. While these are attractive because of their ability to capture common symmetries and represent certain shapes with few parameters, they are ill-suited for modeling many natural objects that do not exhibit the set of regularities incorporated into the primitives.

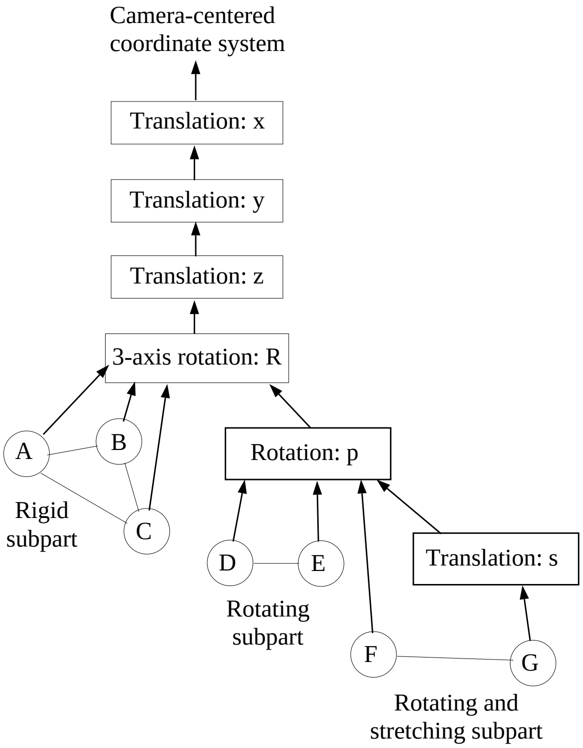

Figure 1: Model points (in circles) are represented as leaves in a tree of rotation or translation transformations. The position and partial derivatives of each point in camera-centered coordinates is determined by the transformations along the path back to the root.

The field that has most thoroughly examined the problem of representing the visual appearance of arbitrary objects is computer graphics. The lessons from developments in that field are quite clear: complex analytic representations have given way to simple local approximations as the most cost-effective solution. The most common forms of local approximation now used for model representation prior to rendering are polygonal surface patches, with the appropriate interpolation in the various parameters of interest for display. Since an arbitrary function can be approximated to any desired degree of accuracy by using enough simple local approximations, the only important issue at this level of representation is one of efficiency. Experience in computer graphics has tended to show that the increased number of approximating patches required for simple linear approximations is more than compensated for by the speed with which they can be manipulated. Of course, more complex splines and volumetric primitives may still be used for model input or other higher-level reasoning.

As with computer graphics, vision is based upon the art of approximation. Of course, it is important to approximate the appropriate measurements, as otherwise an approximation in one quantity may introduce unwanted errors in its derivatives or other functions that depend upon it. In model-based vision, we are concerned with correctly approximating those functions that will be matched with image measurements. In the case of edge-based matching, this will include the projected locations, tangents, curvatures, and discontinuities of edges. If shading or surface properties were being matched, then surface curvatures must also be approximated. We have developed a modeling system that allows these quantities to be modeled as a function of viewpoint and internal model parameters to any desired degree of accuracy and used for efficient parameter solving.

Although model-based vision can learn much from computer graphics, the modeling requirements also have important differences. In model-based matching to 2-D images, the models are matched to derived image features rather than being used to generate dense surface descriptions. For example, it is important to be able to directly calculate the positions of occluding contours, which is not possible in many modeling systems developed for computer graphics. Since the models are projected and manipulated in the inner-loop of the matching process, it is important that all possible sources of efficiency particular to the vision domain be exploited. In addition, certain quantities that do not occur in graphics applications, such as derivatives with respect to model parameters, must be efficiently represented and computed. For all these reasons, it is necessary to develop a modeling system aimed at vision rather than adopting existing systems developed for graphics.

A simple modeling language has been developed that can be used to describe arbitrary models and their internal parameters for use in model-based vision. The language is used to define and name 3-D points, edges and surface patches, as well as model parameters defining rotations and translations. Each model point is a leaf in a tree of coordinate transformations that represent any combination of previous rotations and translations specified by different parameters (see Figure 1). The same mechanism is used to handle both viewpoint parameters and internal model parameters, so the root of this tree represents the camera coordinate frame.

When an internal model is built from a description in this language, a dense pointer network is constructed that links each edge element to its adjoining surface patches and endpoints. A caching mechanism is used so that the visibility of each surface polygon and the projection of each point is calculated only once, unlike in most graphics modeling systems. Because each point on a surface or line may move independently by being attached to different frames, it is possible to specify arbitrary flexible motions of models. For example, the line connecting the points F and G in Figure 1 can stretch under the influence of parameter s and rotate in conjunction with another model subpart under the influence of parameter p (each translation frame specifies a translation direction and each rotation specifies a rotation axis).

Edges are labeled according to whether they lie on a smooth surface or form a discontinuity. By caching a record of the surface normal for the patch on each side of each edge, the visibility and locations of the occluding boundaries and surface discontinuities can be generated very efficiently even when there are large numbers of underlying polygons. An example of a model with occluding boundaries is shown in Figure 2. Later, we will describe how the model representation enables the efficient computation of partial derivatives of image features with respect to each parameter.

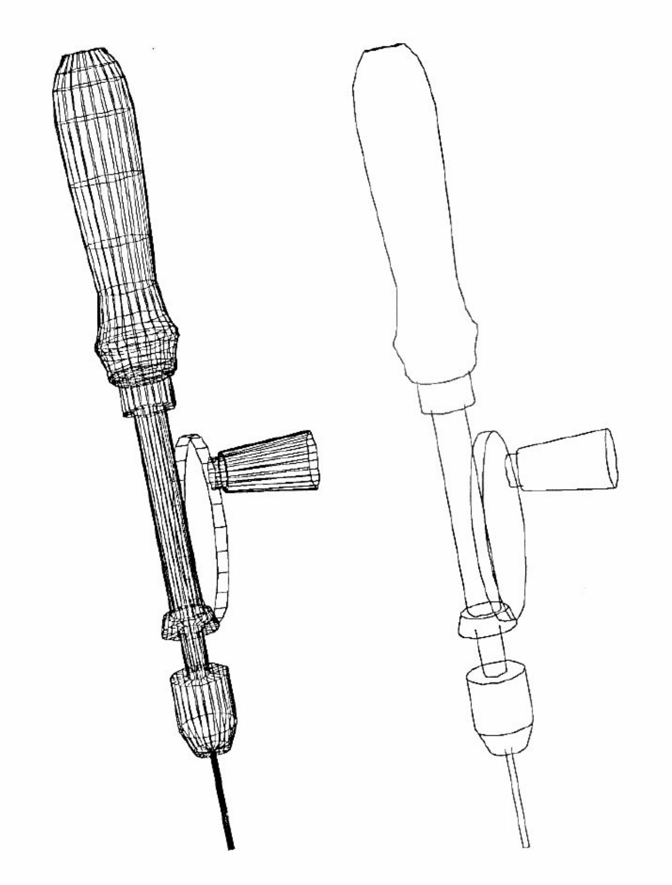

Figure 2: An example of a model with curved surfaces and an internal parameter specifying rotation of the handle. The underlying approximating patches are shown on the left, and the generated contours for matching are shown on the right.

4 Solving for viewpoint and model parameters

Projection from 3-D to 2-D is a non-linear operation. Fortunately, however, it is a smooth and well-behaved transformation. Rotation in depth prior to projection transforms the projected points as a function of the cosine of the rotation angle. Translation towards or away from the camera introduces perspective distortion as a function of the inverse of the distance. Translation parallel to the image plane is almost entirely linear. Translations and rotations associated with internal model parameters have effects that are identical to the viewpoint parameters, but applied to only a subset of the model points. All of these transformations are smooth and well behaved.

Therefore, this problem is a promising candidate for the application of Newton's method, which is based on assuming that the function is locally linear. While this does require starting with an appropriate initial choice for the unknown parameters and faces the risk of converging to a false local minimum, we will see below that stabilization methods can be used to make this method highly effective in practice.

4.1 Newton's method and least-squares minimization

Rather than solving directly for the vector of non-linear parameters, $\mathbf{p}$, Newton's method computes a vector of corrections, $\mathbf{x}$, to be subtracted from the current estimate for $\mathbf{p}$ on each iteration. If $\mathbf{p}^{(i)}$ is the parameter vector for iteration i, then,

$$\mathbf{p}^{(i+1)} = \mathbf{p}^{(i)} - \mathbf{x}.$$

Given a vector of error measurements, $\mathbf{e}$, between components of the model and the image, we would like to solve for an $\mathbf{x}$ that would eliminate this error. Based on the assumption of local linearity, the affect of each parameter correction, x_i, on an error measurement will be x_i multiplied by the partial derivative of the error with respect to that parameter. Therefore, we would like to solve for $\mathbf{x}$ in the following matrix equation:

$$\mathbf{Jx} = \mathbf{e}$$

where **J** is the Jacobian matrix:

$$J_{ij} = \frac{\partial e_i}{\partial x_j}.$$

Each row of this matrix equation states that one measured error, e_i, should be equal to the sum of all the changes in that error resulting from the parameter corrections. If all these constraints can be simultaneously satisfied and the problem is locally linear, then the error will be reduced to zero after subtracting the corrections.

If there are more error measurements than parameters, this system of equations may be overdetermined (in fact, this will always be the case given the stabilization methods presented below). Therefore, we will find an $\mathbf{x}$ that minimizes the 2-norm of the residual rather than solves for it exactly:

$$\min \|\mathbf{Jx} - \mathbf{e}\|^2.$$

Since $\|\mathbf{Jx} - \mathbf{e}\|^2 = (\mathbf{Jx} - \mathbf{e})^{\mathbf{T}}(\mathbf{Jx} - \mathbf{e})$, it can be shown that this minimization has the same solution as the normal equations,

$$\mathbf{J^T J x} = \mathbf{J^T e}. \tag{1}$$

where $\mathbf{J^T}$ is the transpose of $\mathbf{J}$. This minimization is making the assumption that the original non-linear function is locally linear over the range of typical errors, which is true to a high degree of approximation for the projection function with typical errors in image measurements.

Therefore, on each iteration of Newton's method, we can simply multiply out $\mathbf{J^T J}$ and $\mathbf{J^T e}$ in the normal equations (1) and solve for $\mathbf{x}$ using any standard method for solving a system of linear equations. Many numerical texts criticize this use of the normal equations as potentially unstable, and instead recommend the use of Householder orthogonal transformations or singular value decomposition. However, a close study of the trade-offs indicates that in fact the normal equations provide the best solution method for this problem. The solution using the normal equations requires only half as many operations as the Householder algorithm (and an even smaller fraction with respect to SVD), but requires a precision of twice the word-length of the Householder algorithm in order to solve problems that are equally ill-conditioned [9, 16]. Given the stabilization methods described below, the normal equations are never sufficiently ill-conditioned to require more than single-precision floating point arithmetic, and therefore are more efficient in practice than any of the competing methods. Even if higher precision were required, the trade-offs for single versus double precision computation on modern hardware would likely favor the normal equations.

4.2 Efficient computation of partial derivatives

One of the most expensive aspects of implementing this solution method is calculating the Jacobian matrix of partial derivatives. Therefore, we have developed methods for using precomputation and shared data structures to reduce these costs. In addition, a special technique is used to handle derivatives with respect to full 3-D rotations in order to eliminate singularities and increase the rate of convergence.

As described earlier in the section on model representation, all model points are leaves in a tree of “frame” data structures. Each frame represents a rotation or translation with respect to its parent. Therefore, by tracing back to the root of the tree from each model point, it is possible to identify the set of variable transformations that influence that point. Each frame data structure also contains precomputed results for the current view that can be used by all points which depend on that frame in order to compute their partial derivatives with respect to that frame’s parameters. As there are usually many points influenced by each frame, any precomputation of results for the frame is far more efficient than computing them for each point.

It is possible that the same parameter will appear in more than one frame along a path through the tree (e.g., the last 2 joints of a human finger do not move independently, but depend on a single parameter of tendon contraction). This case is easily handled by simply summing all of the partial derivatives for a particular parameter.

Each type of frame transformation requires different precomputed results, so these are described individually as follows.

Translation. Each variable translation frame contains a 3-D vector giving the directional derivative in camera-centered coordinates with respect to that frame’s variable. As all points depending on that frame will have this same directional derivative, no further computation is required.

Rotation about one axis. Each variable rotation frame contains the 3-D angular velocity vector and the origin of rotation for the current viewpoint. The directional derivative of each point that depends on the frame is computed by taking the cross product of the angular velocity vector with the vector from the origin of rotation to the point.

Rotation about three axes. If we compose three rotations about individual axes in order to compute an arbitrary 3-D rotation, singularities can easily result where the sequential composition of the three rotations fail to specify independent directions of rotation. Therefore, we represent full three-degree-of-freedom rotations with a 3 by 3 rotation matrix, and compute corrections about each of the coordinate axes to be composed with this rotation. This also has the benefit that the derivatives can be computed in an extremely efficient form. For example, the directional derivative of a point with respect to an incremental rotation about the x-axis is the vector $(0, -z, y)$, where z and y refer to the coordinates of the vector from the origin of rotation to the point.

Once the directional derivatives of each model point have been computed, it is simply a matter of projecting these into image coordinates (u, v). Perspective projection of a model point (x, y, z) in camera-centered coordinates to produce an image point (u, v) is given as follows:

$$u = \frac{-fx}{z} \quad \text{and} \quad v = \frac{-afy}{z}$$

where f is a constant proportional to the focal length of the camera lens. We include another constant, a, specifying the width-to-height aspect ratio of each pixel in the original image, as most current video standards have non-square aspect ratios. Taking the partial derivative of each of the above functions with respect to a parameter p, we get

$$\frac{\partial u}{\partial p} = \frac{-f}{z}\left(\frac{\partial x}{\partial p} - \frac{x}{z}\frac{\partial z}{\partial p}\right)$$

and

$$\frac{\partial v}{\partial p} = \frac{-af}{z}\left(\frac{\partial y}{\partial p} - \frac{y}{z}\frac{\partial z}{\partial p}\right)$$

Here the partial derivatives of x, y and z with respect to p are simply the components of the directional derivatives calculated earlier.

4.3 Measuring perpendicular errors for curves

The methods above would be sufficient if we had matches between points on the model and points in the image. However, in most cases the matches will actually be between projected contours of the model and partial edges in the image. Since the precise position of the endpoints of image edges are unknown (and may be displaced due to occlusion), it is necessary to minimize only the perpendicular distance from points on an image edge to the projected model curve.

It might be thought that self-occluding edges of curved surfaces would require special treatment, as the actual model edge that forms such an occluding contour will shift with changes in viewpoint. However, the surface normal at such an occluding point is exactly perpendicular to the viewing direction, and therefore the instantaneous motion of the contour projected into the image is zero as nearby points on the surface replace it. For larger rotations, the error introduced by non-linearity is easily handled through the same iterations that compensate for other non-linearities.

In order to measure the perpendicular distance from an image point to a projected 2-D model line, it is useful to express the projected model line in the following form:

$$u \sin\theta - v \cos\theta = d$$

where θ is the orientation of the line with respect to the u-axis and d is the signed perpendicular distance of the line from the origin. If we substitute an image point (u', v') into the left side of this equation and calculate a new d', then the signed perpendicular distance of this point from the line is $d' - d$. The partial derivative of this perpendicular error measure is just a linear combination of the partial derivatives of u and v:

$$\frac{\partial d}{\partial p} = \sin\theta \, \frac{\partial u}{\partial p} - \cos\theta \, \frac{\partial v}{\partial p}$$

In practice, we calculate $\sin\theta$ and $\cos\theta$ from 2 points, (u_1, v_1) and (u_2, v_2), on the line. Let L be the length of the line between these points:

$$L = \sqrt{(u_2 - u_1)^2 + (v_2 - v_1)^2}$$

then

$$\cos\theta = \frac{u_2 - u_1}{L} \quad \text{and} \quad \sin\theta = \frac{v_2 - v_1}{L}$$

The perpendicular error is measured between selected points on the image curve and the perpendicular projection of this point onto the closest segment of the projected model curve. This determination of the closest matching point is updated on each iteration of convergence.

4.4 Determining a starting position for convergence

Worrall, Baker & Sullivan [34] have studied the range of convergence for the author's earlier version of this algorithm using Monte Carlo techniques. They found that the algorithm would converge to the correct solution in virtually every case for rotation errors of less than 90 degrees (translation errors have almost no effect). The number of iterations rises with increasing errors up to an average of about 6 iterations at 90 degrees. With the stabilization methods described in the next section, convergence is significantly improved over even these levels.

Therefore, the accuracy requirements for determining the initial starting position are quite minimal. For the motion tracking problem which serves as our initial focus, we simply use the parameter estimates from the previous frame added to a velocity estimate for each parameter obtained from the previous 2 frames. For a general recognition problem, properties of the image matches that are being fitted can be used to determine initial parameter estimates. For rotation in depth, each match can vote for a mean direction from which it is visible (very few model features are visible from all viewpoints) and these direction vectors can be averaged. For rotation in the image plane, we can project the model from the estimated rotation in depth and take the average image rotation between projected model edges and the matching image edges. Estimates for translation can be made by matching the centers of gravity and standard deviations from the centers of gravity for the projected model features and image features. See [21] for an example of calculating initial estimates for a recognition problem.

If there are only about as many matches as are needed to solve for the degrees of freedom, then it is possible that there is more than one local minimum. This problem can be overcome by starting with some extra matches (the solution adopted in the author's applications), by attempting to converge from several starting positions, or by using an analytic method applied to subsets of the matches (as in Dhome *et al.* [4]) to compute a candidate set of starting positions. Yet another approach is to construct an inverse lookup table that maps feature measurements into approximate viewpoint parameter estimates. Such an approach has been used by Thompson and Mundy [32] for vertices and by Goad [7] for a wide range of arbitrary model features.

5 Stabilizing the solution

As long as there are significantly more constraints on the solution than unknowns, Newton's method as described above will usually converge in a stable manner from a wide range of starting positions. However, in both recognition and motion tracking problems, it is often desirable to begin with only a few of the most reliable matches available and to use these to narrow the range of viewpoints for later matches. Even when there are more matches than free parameters, it is often the case that some of the matches are parallel or have other relationships which lead to an ill-conditioned solution. These problems are further exacerbated by having models with many internal parameters.

5.1 Specifying a prior model

All of these problems can be solved by introducing prior constraints on the desired solution that specify the default to be used in the absence of further data. In many situations, the default solution will simply be to solve for zero corrections to the current parameter estimates. However, for certain motion tracking problems, it is possible to predict specific final parameter estimates by extrapolating from velocity and acceleration measurements, which in turn imply non-zero preferences for parameter values in later iterations of non-linear convergence.

Any of these prior constraints on the solution can be incorporated by simply adding rows to the linear system stating the value that we wish to assign each parameter:

$$\begin{bmatrix} \mathbf{J} \\ \mathbf{I} \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{e} \\ \mathbf{d} \end{bmatrix} \tag{2}$$

The identity matrix **I** adds one row for specifying the value of each parameter, and d_i specifies the desired default value for parameter i.

The obvious problem here is that there is no specification of the trade-offs between meeting the constraints from the data versus those of the prior model. The appropriate solution is to weight each row of the matrix equation so that each element of the right-hand side has the same standard deviation. Therefore, as we minimize the error vector, each constraint will contribute in proportion to the number of standard deviations from its expected value.

We will normalize each row of the system to unit standard deviation. If the image measurements are in pixels, then leaving these with a standard deviation of 1 is already a good first estimate for the error in measuring the position of image features. In our matching algorithm, we also take account of potential ambiguities in the match to increase the standard deviation (i.e., reduce the weighting) for matches that exhibit more than one nearby alternative, so that uncertainties in the correct match for nearby alternatives translate into the appropriate uncertainties in position.

The more important normalization is to weight the prior model according to the standard deviations in the prior estimates for each parameter. This is relatively straightforward in the case of motion tracking, where limits on the acceleration of each parameter from frame to frame can be expressed as a standard deviation. However, in the case of model-based recognition from any viewpoint, it may seem that the range of expected values is infinite. In fact, each parameter is limited during convergence because we are assumed to be starting from some initial approximation to the viewpoint. Therefore, the rotation parameters will have a standard deviation of at most $\pi/2$, and the translations will be limited to maintaining the position of the object within the image frame. Internal model parameters will have standard deviations corresponding to a large fraction of their valid range of movement. These deviations may be large in comparison to those arising from the image measurements, but they still play a substantial role in stabilizing the solution for ill-conditioned problems. In fact the standard deviations can be made several times smaller without an adverse effect on the degree to which the final solution fits the data measurements, because the non-linear iterative solution can reset the starting point of the prior model to the results of each previous iteration.

5.2 Efficient computation of stabilization

The prior estimates of the parameter values will be weighted by a diagonal matrix $\mathbf{W}$ in which each weight is inversely proportional to the standard deviation, σ_i, for parameter i:

$$W_{ii} = \frac{1}{\sigma_i}$$

This matrix is used to scale each row of the prior model in the lower part of equation (2). We assume that the constraints based on image measurements in the upper part of the equation are already scaled to have unit standard deviation.

$$\begin{bmatrix} \mathbf{J} \\ \mathbf{W} \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{e} \\ \mathbf{Wd} \end{bmatrix}$$

We will minimize this system by solving the corresponding normal equations:

$$\begin{bmatrix} \mathbf{J^T} \; \mathbf{W^T} \end{bmatrix} \begin{bmatrix} \mathbf{J} \\ \mathbf{W} \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{J^T} \; \mathbf{W^T} \end{bmatrix} \begin{bmatrix} \mathbf{e} \\ \mathbf{Wd} \end{bmatrix}$$

Which multiplies out to

$$\left(\mathbf{J^T J} + \mathbf{W^T W} \right) \mathbf{x} = \mathbf{J^T e} + \mathbf{W^T W d}$$

Since $\mathbf{W}$ is a diagonal matrix, $\mathbf{W^TW}$ is also diagonal but with each element on the diagonal squared. This means that the computational cost of the stabilization is trivial, as we can first form $\mathbf{J^TJ}$ and then simply add small constants to the diagonal that are the inverse of the square of the standard deviation of each parameter. If $\mathbf{d}$ is non-zero, then we add the same constants multiplied by $\mathbf{d}$ to the right hand side. If there are fewer rows in the original system than parameters, we can simply add enough zero rows to form a square system and add the constants to the diagonals to stabilize it.

5.3 Forcing convergence

Even after incorporating this stabilization based on a prior model, it is possible that the system will fail to converge to a minimum due to the fact that this is a linear approximation of a non-linear system. We can force convergence by adding a scalar parameter λ that can be used to increase the weight of stabilization whenever divergence occurs. The new form of this system is

$$\begin{bmatrix} \mathbf{J} \\ \lambda\mathbf{W} \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{e} \\ \lambda\mathbf{Wd} \end{bmatrix}$$

This system minimizes

$$||\mathbf{Jx} - \mathbf{e}||^{\mathbf{2}} + \lambda^{\mathbf{2}}||\mathbf{W}(\mathbf{x} - \mathbf{d})||^{\mathbf{2}}$$

Many people in the vision community will recognize this as an example of regularization using a Tikhonov [33] stabilizing functional, as has been applied to many areas of low-level vision (Poggio *et al.* [28]). In this case, the parameter λ controls the trade-off between approximating the new data, $||\mathbf{Jx} - \mathbf{e}||^{\mathbf{2}}$, and minimizing the distance of the solution from its original starting position, $\mathbf{d}$, prior to non-linear iteration, $||\mathbf{W}(\mathbf{x} - \mathbf{d})||^{\mathbf{2}}$.

The use of this parameter λ to force iterative convergence for a non-linear system was first studied by Levenberg [17] and later reduced to a specific numerical procedure by Marquardt [24]. They realized that as the parameter λ is increased, the solution would increasingly correspond to pure gradient descent with smaller and smaller step sizes, along with its properties of guaranteed (but slow) convergence. For decreasing λ, the problem instead moves over to Newton's method, with its fast quadratic convergence near the solution but the possibility of divergence when starting too far away. Therefore, Marquardt suggested the simple solution of monitoring the residual of each solution and increasing λ by factors of 10 until the residual decreased; otherwise, λ is decreased by a factor of 10 on each iteration. This does not guarantee any particular rate of convergence and can, of course, converge to a local rather than global minimum. However, it has proved highly effective in practice and is one of the most widely used methods for non-linear least-squares.

Marquardt did not assume any prior knowledge of the weighting matrix $\mathbf{W}$, but instead estimated each of its elements from the euclidean norm of the corresponding column of $\mathbf{J^TJ}$. In our case, the availablity of $\mathbf{W}$ allows the algorithm to perform much better when a column of $\mathbf{J^TJ}$ is near zero. It also gives the stabilization a much more predictable behavior. Increasing the value of λ will essentially freeze the parameters having the lowest standard deviations and

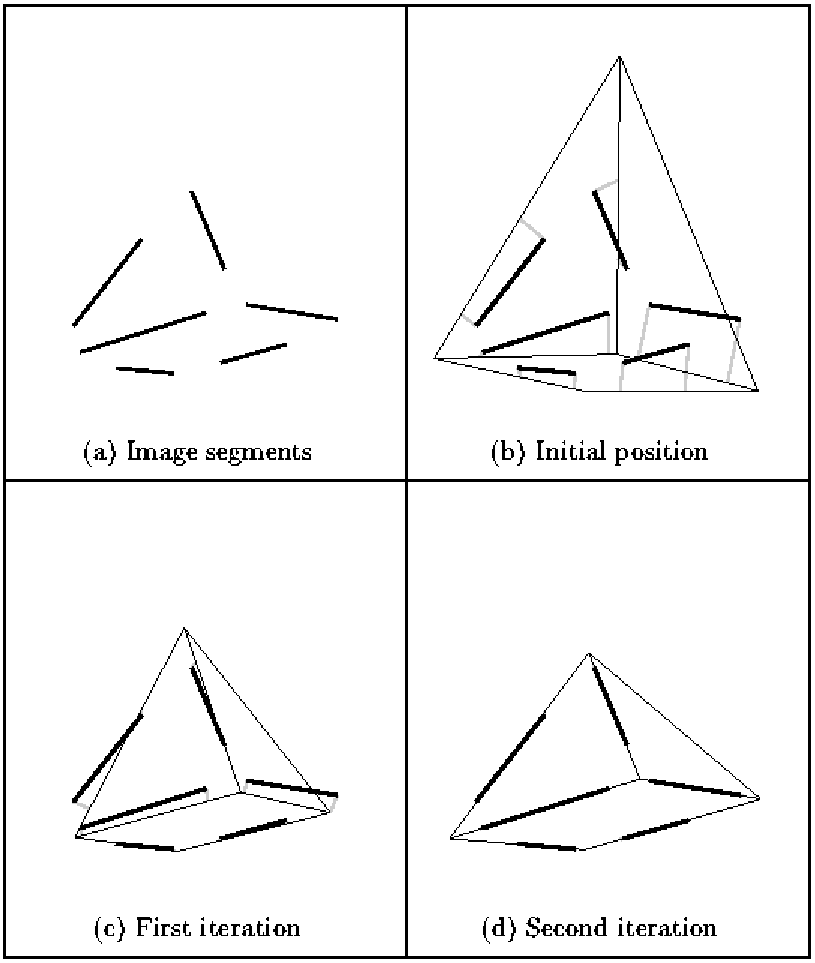

Figure 3: Two iterations of convergence are shown for determining 3-D viewpoint and the height of the pyramid from partial matches to 2-D image segments. Perpendicular errors being minimized are displayed as gray bars between model and image edges.

therefore solve first for those with higher standard deviations. For our problem, this implies that convergence for difficult problems will proceed by solving first for translations and then proceeding on subsequent iterations to solve for rotations and finally short-range internal model parameters.

6 Results of implementation

All of the methods for object modeling and parameter solving described above have been implemented in about 4000 lines of C code. A very simple example of model fitting is shown in Figure 3. The model is a pyramid with an internal parameter allowing for variable height. The model was projected from one particular set of parameter values, and random intervals of some

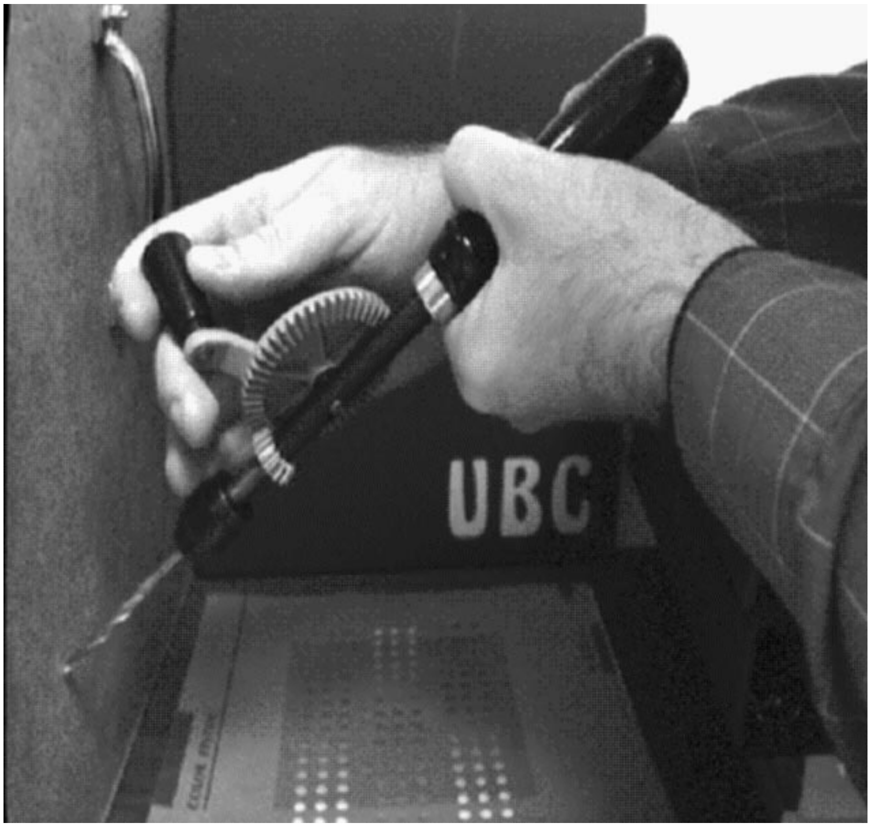

Figure 4: An image from a motion sequence of a person using a hand drill.

of the projected segments were chosen for matching. The model parameters were changed to produce the starting parameter estimates shown in Figure 3(b). In this figure, the perpendicular errors being minimized are displayed as gray bars between the projected model segments and the matching image segments. Figures 3(c) and 3(d) show the output following the first two iterations of the stabilized algorithm presented above. This fast rate of convergence within a couple of iterations is typical over a wide range of initial parameter values (up to at least 60 degree errors in rotation parameters). See Worrall, Baker & Sullivan [34] for a systematic exploration of convergence over a wide range of errors, even prior to the addition of the stabilization and Levenberg-Marquardt methods. In fact, divergence is relatively rare, so it is uncommon for the Levenberg-Marquardt method to take effect; however, its computational cost is also low, so it is probably of practical value.

6.1 Application to motion tracking

One initial application of these methods has been to the problem of model-based motion tracking. A Datacube image processor was used to implement Marr-Hildreth [25] edge detection in real time on 512 by 485 pixel images. The image containing these edge points is transferred to

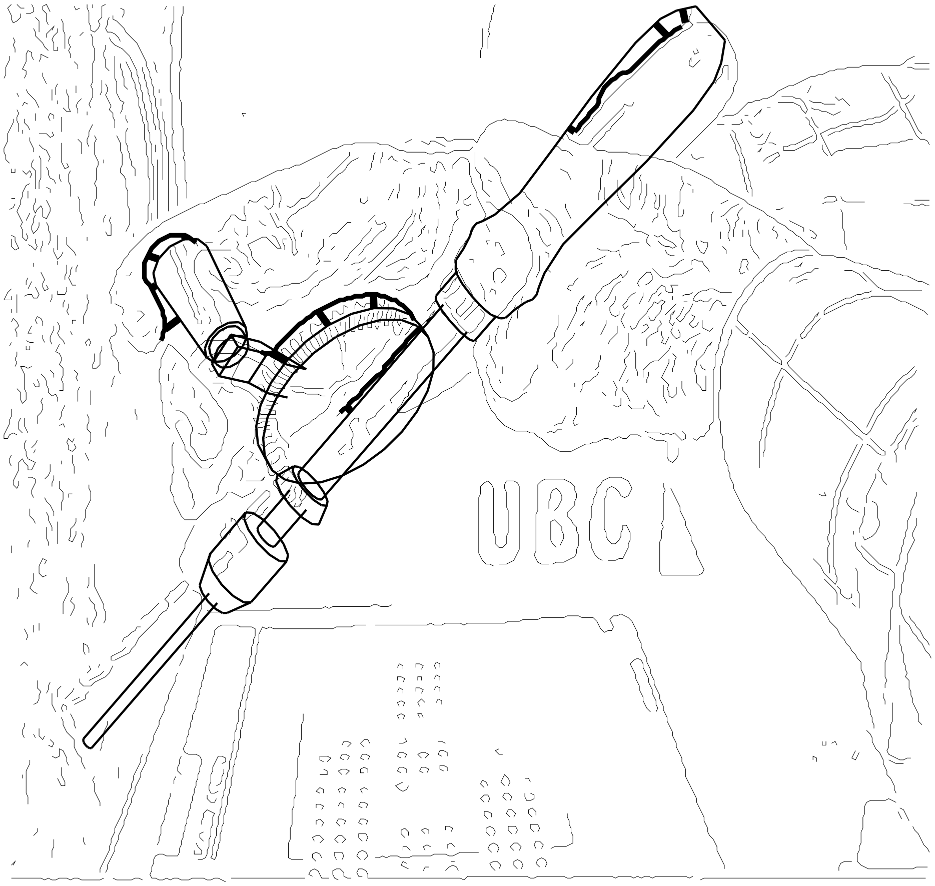

Figure 5: Edges extracted from the image of Figure 4 using the Canny edge detector. Superimposed on these edges are the model from its previous estimated viewpoint, nearby matching edges, and perpendicular errors to be minimized.

a Sun 3/260, where the edges are linked into lists on the basis of local connectivity. A fairly simple matching technique is used to identify the image edges that are closest to the current projected contours of a 3-D model. The few best initial matches are used to perform one iteration of the viewpoint solution, then further matches are generated from the new viewpoint estimate. Up to 5 iterations of this procedure are performed, with a gradually narrowing range of image locations which are searched for potential matches (this helps to eliminate any false outlier matches). For simple models with straight edges, all of these steps can be performed in less than 1 second, resulting in a system that can perform robust but rather slow real-time motion tracking. We have run this system for thousands of frames at a time by holding an object in front of the video camera and slowly moving it. Correctness of the motion tracking can be easily judged in real time by watching a wire-frame model superimposed on the image from the current set of parameter estimates. We are currently exploring the use of parallel architectures that could greatly speed the operation of this system so that it performs at video rates for complex object models.

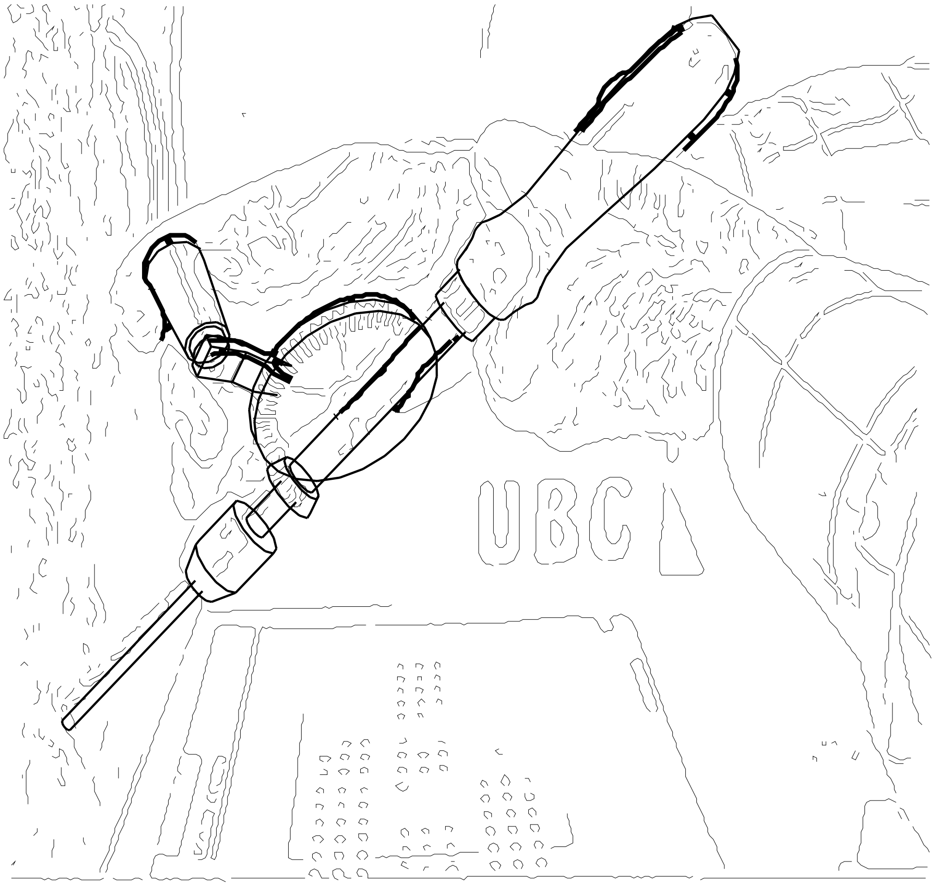

Figure 6: The new model position and handle rotation after one iteration of model fitting. New matches to image edges are shown with heavy lines.

Figures 4–7 show the operation of the system for one frame of motion tracking. However, due to the complexity of the model, this version requires about 6 seconds of processing per frame on a Sun 3/260 and does not operate in real time. Figure 4 shows an image of a hand drill from which edges are extracted with a simplified version of the Canny edge detector. In Figure 5, the model is shown superimposed on these edges from the previous best estimate of its current viewpoint. A simple matching algorithm is used that finds image edges that are close to the projected model curves over the maximum possible length of the edge. These matches are ranked according to their length and average separation, and the best ones are chosen for minimization. The selected matches are shown with heavy lines in Figure 5 along with perpendicular bars marking the errors between model and image curves that are minimized. After one iteration of model fitting, the new model position is shown in Figure 6 along with a new set of image matches generated from this position. Note that the rotation of the handle is a free parameter along with the viewpoint parameters. After this second iteration of convergence, the final results of model fitting are shown superimposed on the original image in Figure 7. Note that due to occlusion and errors in low-level edge detection, this final result is based on

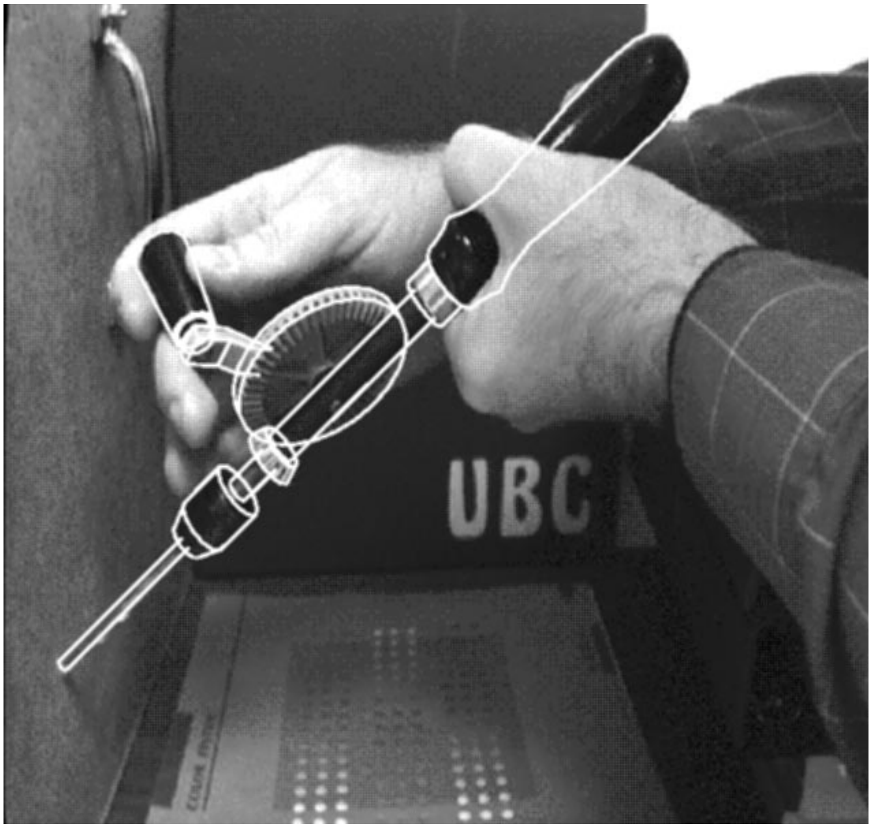

Figure 7: After the second iteration of convergence, the model is shown superimposed on the original image.

only a small subset of the predicted image edges. However, due to the overconstrained nature of the problem, in which far more measurements are available than unknown parameters, the final result can be reliable and accurate.

7 Conclusions and future directions

This paper has presented general methods for fitting models with arbitrary curved surfaces and any number of internal parameters to matched image features. Considerable attention has been given to issues of robustness and efficiency, and these techniques should serve as a practical basis for model fitting in most applications of model-based vision.

There are a number of directions in which these methods could be further improved. One is in dealing with objects that have very large numbers of variable parameters. Since the complexity of solving a linear system rises as $O(n^3)$ in the number of variables, it would likely be more efficient to partition problems with very large numbers of parameters into smaller subsets. The simultaneous solution method would be used for all parameters with large ranges of uncertainty, but the remaining ones would be solved for on the basis of local independent optimization. This would become particularly important if generic classes of objects are modeled, as was done in the Acronym system [3], in which almost every dimension of the object is variable.

While this paper extends the modeling and parameter solving components of a vision system so that they can work with curved objects, there is still much research to be done regarding low-level curve segmentation and grouping. The author has developed some multi-scale curve smoothing methods [23] that would be suitable for the initial curve description, but much remains to be done at the level of grouping and indexing in order to produce a fully general system for recognition of curved objects. By making use of properties such as curvature, smoothness and higher-level groupings, it should be possible to make major gains in the reliability of matching.

References

[1] Binford, Thomas O., “Visual perception by computer,” *IEEE Systems Science and Cybernetics Conference,* Miami (1971).

[2] Bray, Alistair J., “Tracking objects using image disparities,” *Image and Vision Computing,* **8,** 1 (1990), 4–9.

[3] Brooks, Rodney A., “Symbolic reasoning among 3-D models and 2-D images,” *Artificial Intelligence,* **17** (1981), 285-348.

[4] Dhome, M., M. Richetin, J.T. Lapresté, and G. Rives, “Determination of the attitude of 3-D objects from a single perspective view,” *IEEE PAMI,* **11,** 12 (1989), 1265–78.

[5] Faugeras, O.D., and M. Hebert, “The representation, recognition and locating of 3-D objects,” *The International Journal of Robotics Research,* **5,** 3 (1986), 27–52.

[6] Fischler, Martin A. and Robert C. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM,* **24,** 6 (1981), 381-395.

[7] Goad, Chris, “Special purpose automatic programming for 3D model-based vision,” in *From Pixels to Predicates,* ed. Alex Pentland, (Ablex Publishing Co., 1986), 371-391.

[8] Goldberg, Robert R., and David G. Lowe, “Verification of 3-D parametric models in 2-D image data,” *Proc. of IEEE Workshop on Computer Vision,* Miami (November 1987), pp. 255-257.

[9] Golub, Gene H., and Charles F. Van Loan, *Matrix Computations,* 2nd Edition, The Johns Hopkins University Press, Baltimore (1989).

[10] Grimson, Eric, and Thomás Lozano-Pérez, “Model-based recognition and localization from sparse range or tactile data,” *Int. Journal of Robotics Research,* **3** (1984), 3-35.

[11] Haralick, R.M., H. Joo, C. Lee, X. Zhuang, V.G. Vaidya, and M.B. Kim, “Pose estimation from corresponding point data,” *IEEE Trans. on Systems, Man and Cybernetics,* **19**, 6 (1989), 1426–1446.

[12] Horaud, R., B. Conio, O. Leboulleux, and B. Lacolle, “An analytic solution for the perspective 4-point problem,” *Proc. Conf. Computer Vision and Pattern Recognition*, San Diego (June 1989), 500–507.

[13] Huttenlocher, Daniel P., and Shimon Ullman, “Object recognition using alignment,” *Proc. First Int. Conf. on Computer Vision*, London, England (June 1987), 102–111.

[14] Ishii, M., S. Sakane, M. Kakikura and Y. Mikami, “A 3-D sensor system for teaching robot paths and environments,” *The International Journal of Robotics Research*, **6**, 2 (1987), pp. 45–59.

[15] Kumar, Rakesh, “Determination of camera location and orientation,” *Proc. DARPA Image Understanding Workshop,* Palo Alto, Calif. (1989), 870–879.

[16] Lawson, Charles L., and Richard J. Hanson, *Solving Least Squares Problems,* Prentice-Hall, Englewood Cliffs, NJ (1974).

[17] Levenberg, K., “A method for the solution of certain non-linear problems in least squares,” *Quart. Appl. Math.,* **2** (1944), 164–168.

[18] Liu, Y., T.S. Huang and O.D. Faugeras, “Determination of camera location from 2-D to 3-D line and point correspondences,” *IEEE PAMI,* **12,** 1 (1990), 28–37.

[19] Lowe, David G., “Solving for the parameters of object models from image descriptions,” *Proc. ARPA Image Understanding Workshop* (College Park, MD, April 1980), 121–127.

[20] Lowe, David G., *Perceptual Organization and Visual Recognition* (Boston, Mass: Kluwer Academic Publishers, 1985).

[21] Lowe, David G., “Three-dimensional object recognition from single two-dimensional images,” *Artificial Intelligence,* **31,** 3 (March 1987), 355-395.

[22] Lowe, David G., “The viewpoint consistency constraint,” *International Journal of Computer Vision,* **1,** 1 (1987), 57-72.

[23] Lowe, David G., “Organization of smooth image curves at multiple scales,” *International Journal of Computer Vision* **3**, 2 (June 1989), 119–130.

[24] Marquardt, Donald W., “An algorithm for least-squares estimation of nonlinear parameters,” *Journal. Soc. Indust. Applied Math.,* **11,** 2 (1963), 431–441.

[25] Marr, David, and Ellen Hildreth, “Theory of edge detection,” *Proc. Royal Society of London, B,* **207** (1980), 187-217.

[26] McIvor, Alan M., “An analysis of Lowe’s model-based vision system,” *Proc. Fourth Alvey Vision Conference,* Univ. of Manchester (August 1988), 73–78.

[27] Pentland, Alex P., “Perceptual organization and the representation of natural form,” *Artificial Intelligence,* **28,** 3 (1986), 293-331.

[28] Poggio, Tomaso, Vincent Torre and Christof Koch, "Computational vision and regularization theory," *Nature,* **317,** 6035 (Sept. 1985), 314–319.

[29] Ponce, Jean, and David J. Kriegman, "On recognizing and positioning curved 3D objects from image contours," *DARPA Image Understanding Workshop,* Palo Alto, CA (1989), 461–470.

[30] Roberts, L.G., "Machine perception of three-dimensional solids," in *Optical and Electro-optical Information Processing,* eds. J. Tippet *et al.* (Cambridge, Mass.: MIT Press, 1965), 159-197.

[31] Rock, Irvin, *The Logic of Perception* (Cambridge, Mass.: MIT Press, 1983).

[32] Thompson, D.W., and J.L. Mundy, "Model-based motion analysis: Motion from motion," *Robotics Research: The Fourth International Symposium,* R. Bolles and B. Roth, eds., (Cambridge, Mass.: MIT Press, 1988), 299–309.

[33] Tikhonov, A.N., and V.Y. Arsenin, *Solutions of Ill-posed Problems,* W.H. Winston, Washington, D.C., 1977.

[34] Worrall, A.D., K.D. Baker and G.D. Sullivan, "Model based perspective inversion," *Image and Vision Computing,* **7,** 1 (1989), 17–23.