MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 1476 December, 1994

Uncertainty Propagation in Model-Based Recognition

David W. Jacobs
NEC Research Institute
4 Independence Way
Princeton, NJ 08540
dwj@research.nj.nec.com

T. D. Alter
MIT AI Laboratory
545 Technology Square, Room 751
Cambridge, MA 02139
tda@ai.mit.edu

Abstract

Robust recognition systems require a careful understanding of the effects of error in sensed features. In model-based recognition, matches between model features and sensed image features typically are used to compute a model pose and then project the unmatched model features into the image. The error in the image features results in uncertainty in the projected model features. We first show how error propagates when poses are based on three pairs of model and image points. In particular, we show how to simply and efficiently compute the distributed region in the image where an unmatched model point might appear, for both Gaussian and bounded error in the detection of image points, and for both scaled-orthographic and perspective projection models. This result applies to objects that are fully three-dimensional, where past results considered only two-dimensional objects. The result is based on an approximation that accurately linearizes the relationship between matched image points and unmatched, projected model points. Secondly, based on the linear approximation, we show how we can utilize *linear programming* to compute the propagated error region for any number of initial matches. Finally, we use these results to extend, from two-dimensional to three-dimensional objects, robust implementations of *alignment*, *interpretation-tree search*, and *transformation clustering*.

1 Introduction

Given a correspondence between a set of image features and model features, a general problem in recognition is to evaluate the correspondence and improve it if necessary. For instance, for object recognition the model may be a sparse set of 3D points and line segments. For aerial images, the model may be a terrain elevation map that includes the world locations of a small set of landmarks. In some applications, a user may supply the initial correspondence, leaving the computer to estimate and refine the model pose (position and orientation). In other cases, the computer must find the initial correspondence as well; this may be done through a combination of grouping, indexing, and raw search. Important computations involved in the recognition include (1) deciding whether the given correspondence provides an accurate alignment, (2) determining which image features could correspond to each unmatched model feature, and (3) choosing a new match to extend the correspondence. These computations are intertwined with the issue of error propagation, that is, the issue of how error in a set of matched image features propagates to uncertainty in the predicted image locations of the remaining model features. We call these predicted image locations the *uncertainty regions* of the model features, and we derive either bounds on these regions or probability distributions on them, depending on our model of error.

There are several reasons why it is useful to carefully understand the propagation of uncertainty, as opposed to assuming some small, simple uncertainty region and using it in all cases. First, as we will show, uncertainty regions can vary quite a bit in size, and may be quite large for the predicted model features, resulting in many candidate image features for each prediction. In particular, grouping techniques commonly give image features that are close together on an object (e.g., [11, 8, 25, 31, 27, 29]), and we will see that this easily can lead to large uncertainty regions. Even when the matched features are far apart in the image, the uncertainty regions of the unmatched points may still be large, due to the depth of the 3D model. Second, both when the image features are nearby and when they are far apart, there are situations in which the pose of the model is unstable, and the uncertainty regions assume surprising shapes. By understanding the propagation of uncertainty, then, we can determine exactly where to look for features, and we can evaluate the stability of the pose produced by the initial correspondence.

1.1 Summary of Results

Given a set of matched image and model points, we determine an unmatched model point's uncertainty region. We consider this problem for the case in which correspondences are based on point features. We handle both scaled-orthographic and perspective projection models. We also consider two different models of error. First, we consider image points detected with errors that have known, independent Gaussian distributions. Second, we consider a bounded error model, in which we suppose that the error distributions are unknown. In this case we make only the weak assumption that the magnitude of the error vectors can be bounded by some maximum number of pixels ϵ. Given no other information, Gaussians may be the preferred error distribution, since image features are displaced by a sum of error vectors, incurred over a series of processes such as digitization, smoothing, and edge detection. A bounded error model may be useful, however, when errors contain a consistent bias that results in distributions that are significantly skewed from normal. In the first case, we show how Gaussian error in matched image points propagates to an uncertainty region with a Gaussian distribution for an unmatched point. In the second case, we show how bounded error in image points propagates to a bounded uncertainty region describing the possible location of an additional model point.

First we compute the uncertainty regions for sets of three matched points. We derive a simple linear expression that approximates the relationship between the matched and unmatched points. This relationship allows us to show that, for bounded error, the uncertainty region for a fourth point is circular, and to derive analytic expressions for the center and radius of the circle. For Gaussian error, this relationship implies that the propagated distribution of uncertainty is also Gaussian, and provides analytic expressions for the center and standard deviation. We perform experiments to verify that these expressions are accurate for the amount of error that is of interest in most recognition applications.

We also take advantage of the linear relationship by introducing a new algorithm that allows us to determine the uncertainty region for any number of matched points. To do this we approximate our bounded error regions with convex polygons, and then show that we can use linear programming to derive a convex polygon that describes the uncertainty region of the unmatched model point. We experiment with both synthetic images and a real image to observe the accuracy of the uncertainty regions that we compute, and to determine the extent to which they shrink as we match more points.

Finally, we show how to extend previous work for linear projection models to the cases of scaled-orthographic and perspective projections. Using the linear approximation we show that we can use Baird's [6] algorithm to tell whether a set of matches between image and model points are geometrically consistent, and that we can apply Cass' [12] and Breuel's [10] algorithms to find, in polynomial time, the model pose that aligns the maximum number of model and image features to within error bounds. We also extend Jacobs' [28] and Sarachik and Grimson's [39] planar alignment algorithms to 3D objects.

1.2 Projection Models

For reference, we review the models of projection that we refer to in this paper. For perspective projection, we can write the corresponding image position (x, y) of a 3D model point $(\overline{x}, \overline{y}, \overline{z})$ in terms of a 3D, rigid rotation matrix $\mathbf{R}$, a 3D translation vector $\vec{u}$, and a camera focal length f. Letting r_{ij} be the elements of $\mathbf{R}$, we have

$$x = f\frac{r_{11}\overline{x} + r_{12}\overline{y} + r_{13}\overline{z} + u_x}{r_{31}\overline{x} + r_{32}\overline{y} + r_{33}\overline{z} + u_z}, \qquad y = f\frac{r_{21}\overline{x} + r_{22}\overline{y} + r_{23}\overline{z} + u_y}{r_{31}\overline{x} + r_{32}\overline{y} + r_{33}\overline{z} + u_z}, \tag{1}$$

where the rows of $\mathbf{R}$ are orthonormal, and where we assume the origin is at the center of projection. When the focal length f is known, there are six degrees of freedom, and consequently three corresponding model and image points are "minimal" to determine the transformation. Given three corresponding points, there exist up to four solutions for the model pose [17].

This paper extensively considers scaled-orthographic (also known as weak-perspective) projection, in which a 3D object is scaled down and projected orthographically into the image. This projection model is appropriate when the camera is far from the objects being viewed with respect to their sizes. In this case, the image position of $(\overline{x}, \overline{y}, \overline{z})$ can be written in terms of the first two rows of a scaled, 3D rotation matrix, $\mathbf{S} = s\mathbf{R}$, and of a scaled, 3D translation vector, $\vec{b}$. Letting s_{ij} be the elements of $\mathbf{S}$, we have

$$x = s_{11}\overline{x} + s_{12}\overline{y} + s_{13}\overline{z} + b_x, \qquad y = s_{21}\overline{x} + s_{22}\overline{y} + s_{23}\overline{z} + b_y, \tag{2}$$

where $\| (s_{11}, s_{12}, s_{13}) \| = \| (s_{21}, s_{22}, s_{23}) \|$ and $(s_{11}, s_{12}, s_{13}) \cdot (s_{21}, s_{22}, s_{23}) = 0$. There are six degrees of freedom in the scaled-orthographic model-to-image transformation, and consequently three corresponding points are minimal to determine the transformation. Given three corresponding points, the transformation always exists if the model points are not collinear and it generally has two solutions [27, 2]; in particular, the scale factor and translation are always unique, and the rigid rotation matrix is unique up to a reflection of the rotated model about a plane parallel to the image [2].

For 3D linear projection, we remove the two non-linear constraints on the rotation parameters in the scaled-orthographic projection model. This transformation is equivalent to applying a scaled-orthographic transformation to the model, and then applying a scaled-orthographic transformation to the resulting image; in total, this is like taking a picture of a photograph [29]. There are eight degrees of freedom in linear projection, and four corresponding points are minimal to determine the transformation. Given a minimal set of matches, this is the only transformation of the three in which the unmatched model points can be written *linearly* in terms of the matched image points. In particular, let the four image and model points be (x_i, y_i) and $(\overline{x}_i, \overline{y}_i, \overline{z}_i)$, respectively, for $i = 1, 2, 3, 4$. Then we can obtain the first row of the transformation by solving

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \overline{x}_1 & \overline{y}_1 & \overline{z}_1 & 1 \\ \overline{x}_2 & \overline{y}_2 & \overline{z}_2 & 1 \\ \overline{x}_3 & \overline{y}_3 & \overline{z}_3 & 1 \\ \overline{x}_4 & \overline{y}_4 & \overline{z}_4 & 1 \end{bmatrix} \begin{bmatrix} s_{11} \\ s_{12} \\ s_{13} \\ b_x \end{bmatrix}. \tag{3}$$

A similar equation holds for the second row of the transformation. These equations give linear expressions for the transformation parameters in terms of the image point coordinates. Since multiplying a matrix by a vector is a linear operation, applying the computed transformation to any unmatched model point gives a linear expression for the model point's image position in terms of the matched image points.

1.3 Background

Due to the value of top-down knowledge in model-based vision, it is common to generate hypotheses about an object's pose based on a small amount of information, and then to look for evidence to confirm or reject the hypotheses. In the *alignment* approach, a small number of image features are matched to model features to determine the object's pose. This pose is used to project additional model features into the image, which are matched to nearby image features for verification (e.g., Roberts [37], Clark et al. [13], Fischler and Bolles [17], Lowe [34], Ayache and Faugeras [5], Huttenlocher and Ullman [27]). In *interpretation-tree search*, additional matches between model and image features are then used to look for more matches, backtracking if enough valid matches cannot be found (e.g., Bolles and Cain [8], Goad [18], Grimson and Lozano-Pérez [23], Horaud [25]). To obtain the object's pose, some approaches use minimal sets of matches between model and image features (e.g., Clark et al. [13], Fischler and Bolles [17], Ayache and Faugeras [5], Horaud [25], Huttenlocher and Ullman [27]). Other approaches use indexing to match more than the minimal number before looking for confirming features (e.g., Rothwell et al. [38], Thompson and Mundy [43], Lamdan et al. [32], Jacobs [29]).

Most recognition systems take an ad-hoc approach to the problem of accounting for the effects of sensing error on the projected positions of unmatched model features. Some systems match projected model features to image features if they are separated by a distance that is less than some threshold (e.g., Clark et al. [13], Fischler and Bolles [17], Brooks [11], Bolles and Cain [8], Huttenlocher and Ullman [27]). Other systems rank the unmatched image features using heuristics involving distance and orientation, and then pick the feature with highest rank (e.g., Ayache and Faugeras [5], Lowe [34]). Many questions remain concerning the performance of these systems. For example, although we know the minimal number of features needed to generate a model pose, we do not know how accurate the pose must be to allow us to identify the object. In addition, some authors stress the importance of using a minimal set of features [17, 27], while others contend that this will not produce a sufficiently accurate pose for recognition [34]. It is in general difficult to characterize the conditions under which these systems will succeed or fail, or to evaluate the relative effectiveness of the different strategies for recognition, or to understand the extent to which each approach makes the best possible use of the information available. A careful understanding of the effects of sensing error is a prerequisite to doing all of these.

1.3.1 Two-dimensional objects

Recently, there has been considerable effort aimed at better understanding the effects of error on the matching process. Some of this work attempts to design algorithms that are guaranteed to perform well in the presence of error (e.g., Baird [6], Cass [12], Breuel [10]), but most relevant to this paper is work that also examines the propagation of error in recognition systems.

Huttenlocher [26] examined the effects of bounded error on the alignment approach to recognition. This analysis considered planar objects viewed from arbitrary 3D positions, assuming scaled-orthographic projection. Pose was determined by matching three model and image points. For some situations, Huttenlocher placed approximate bounds on the uncertainty regions.

Subsequently, Jacobs [28] showed that the true uncertainty regions are discs, and gave analytic expressions for their centers and radii. These regions are circular because in this case the projection model is linear in such a way that error in any of the three matched image points causes error in a projected model point that is identical but scaled by a constant factor. This constant factor depends on the model structure, but not on the viewpoint. Consequently, the sizes of the uncertainty regions are independent of how far apart in the image are the three matched points, which means the uncertainty is independent of the pose of the model. Jacobs' result was used by Grimson et al. [22] to analyze the false-positive sensitivity of planar alignment.

A number of researchers have also considered the effect of Gaussian error on alignment methods. As mentioned above, for planar objects, each predicted model point can be written as a linear combination of the matched image points. Therefore, Gaussian error in the image points leads to Gaussian uncertainty in every predicted point (e.g., [42]). Sarachik and Grimson [39] used this observation to propose a new method of performing and evaluating alignment approaches to recognition. Beveridge et al. [7] and Wells [47] use a Bayesian approach to derive estimates of the likelihoods of particular model poses.

Error propagation has also been studied in the context of Geometric Hashing approaches to recognition. Costa et al. [15] considered the distribution of uncertainty regions in terms of the affine invariant parameters that describe the image points. Rigoutsos and Hummel [35, 36] also considered this issue for Gaussian and uniform error. Both Costa et al. and Rigoutsos and

Hummel then considered the implications of these results for recognition schemes. Lamdan and Wolfson [33] considered the related problem of determining when three image points provide an unstable basis for Geometric Hashing. Grimson and Huttenlocher [20] considered the effects of bounded error on Geometric Hashing, and provided loose bounds on this effect. Jacobs [28] determined exactly how bounded error effects Geometric Hashing indices. Grimson et al. [22] then further developed this result and used it to analyze the performance of Geometric Hashing algorithms. Sarachik and Grimson's [39] results also apply to Geometric Hashing.

1.3.2 Three-dimensional objects

Error propagation is more complex in recognition systems that deal with fully three-dimensional objects. Bolles et al. [9] studied how error propagates from the parameters of a model-to-image transformation to the predicted model points. Bolles et al. assumed that the errors in the parameters were independent and normally-distributed and that estimates of the distributions would be available. Unlike other previous work, Bolles et al. dealt with perspective projection, which made the relationship between the error vectors in the transformation parameters and the predicted points non-linear. In fact, their analysis is the most similar to our own, because they took a (first-order) approximation that linearizes the error-vector relationship. As a result they obtained Gaussian uncertainty distributions. The main difference with our work, in addition to our treatment of bounded error, is that we will let the error be in the matched image points, instead of assuming we know the distributions for all of the transformation parameters. Furthermore, we will derive direct expressions for the predicted points in terms of the matched points, so that we do not explicitly go through a rigid transformation. We believe this direct approach is more intuitive geometrically, and, in some cases it allows us to obtain a more accurate approximation.

Recently, Grimson et al. [21] presented a formal analysis of error propagation starting from the matched image points, for three-dimensional objects. They considered scaled-orthographic projection and bounded, circular error. Starting from three matched points, they provided a numerical method of bounding the uncertainty in the transformation parameters. Then they used the bounds on the parameters to obtain complicated, loose bounds on the uncertainty regions of the predicted points. Via these bounds, they analyzed the false-positive sensitivity of 3D-from-2D alignment and transformation clustering, in the domain of point features. The numerical technique is less practical, however, for use at run-time in a recognition system.

Using the same projection and error models as Grimson et al. [21], Alter and Grimson [4] presented experiments that show that the true uncertainty regions tend to be circular to a good approximation, and presented a numeric method for more accurately bounding the uncertainty regions. This technique was used to study again the false-positive sensitivity of 3D-from-2D alignment, except also using line features for verification. [4] demonstrated that using points for generating hypotheses and lines for verification could lead to robust recognition. As before, the numerical error-propagation technique is less practical for a real-time system. Furthermore, the two weak-perspective solutions lead to two distinct uncertainty regions, which is not true when the model is planar. The technique in [4] sometimes performed poorly when the two regions overlapped, because it had difficulty distinguishing them.

Also for 3D objects, Weinshall and Basri [46] provided analytic bounds on the amount of error in a least-squares solution that is used to match four model and image points. This is useful because, currently, the least-squares solution itself can be found only through iterative methods.

Finally, there has been a great deal of work on finding a pose that minimizes error, when enough image and model features have been matched to overdetermine the pose. Some of this work analyzes the effect that errors in image features have on the accuracy of the resulting pose, including Kumar and Hanson [30] and Hel-Or and Werman [24]. The work of Hel-Or and Werman is particularly relevant to us, because they also consider how error propagates through the pose to the projections of unmatched feature points. Assuming Gaussian error, they use an extended Kalman filter to find the minimal error pose resulting from a match between any number of image and model points. The Kalman filter then allows them to compute a Mahalanobis distance that indicates the likelihood that error can account for the apparent deviation between a projected model point and a potentially matching image point.

In summary, there are simple analytic solutions for how error propagates from three matched image points, when the objects are two-dimensional and undergo scaled-orthographic projection.

This is true both when the image-point error is bounded by circles and when it is normally distributed. In the case of circular error, every propagated uncertainty region is a circle, whose size is independent of the camera viewpoint.

For three-dimensional objects, it appears empirically that circular error again propagates to circular uncertainty regions. Nevertheless, there is no analytic solution, which would be preferred for building an efficient system. As well, current numerical solutions either significantly overestimate the uncertainty regions or can break down when the two weak-perspective regions overlap. Further, it is not known whether the uncertainty regions are exactly or approximately circles, or whether the sizes of the regions depend on the viewpoint. If the regions are circles only approximately, one would like to know which configurations of the model and image points cause the regions to deviate from circularity. Although much progress has been made in understanding the effects of propagated error, there are significant problems that are not yet understood.

Finally, there have been a number of sensitivity analyses that determine the susceptibility of recognition systems to false-positive errors. Most of these analyses are restricted to two-dimensional objects, because this is where error propagation is most readily understood. Nonetheless, there do exist sensitivity analyses for three-dimensional objects, which use numerical techniques to get a handle on the propagated error.

2 Fourth-Point Uncertainty Region

In this section, we address the following problem: Given exactly three matching point pairs, $(\vec{i}_0, \vec{m}_0)$, $(\vec{i}_1, \vec{m}_1)$, and $(\vec{i}_2, \vec{m}_2)$, where the locations of $\vec{i}_0$, $\vec{i}_1$, and $\vec{i}_2$ contain small amounts of error, what is the error in the computed image position of a fourth model point, $\vec{m}_3$? This section presents an analytic solution to this problem, which is based on a first-order approximation, and results in a linear relationship between the errors in the matched (basis) image points and the error in an unmatched, projected model point. Here we consider weak-perspective projection, and in a later section we extend the results to perspective.

For weak-perspective projection, we show that the linear relationship takes a simple form that can be used to predict the uncertainty region for an unmatched model point when there is bounded error or Gaussian error in the image points. When we allow the three basis image points to be perturbed within bounded error regions, the resulting uncertainty region is also bounded. When we allow for Gaussian error in the image points, the uncertainty region is a probability distribution. Previously these uncertainty regions were known analytically only for planar objects [28, 39]. Our results have no such restriction, and they reduce to the known solutions when the model is planar. Furthermore, when the error in the image points is bounded by circles, the region takes the form of a circle centered at the "nominal point," which is the point that $\vec{m}_3$ projects onto when there is no error in the basis points. This result agrees with the experimental observations of Alter and Grimson [4].

2.1 The Basic Geometry

We begin by examining the propagated uncertainty when there is error in exactly one of three matched image points, $\vec{i}_2$. To do this, we introduce a particular representation for any third model point that allows us to see how a change in the location of the third image point affects the projected location of any unmatched model point. In this representation, we let the origin of the image coordinate system be at $\vec{i}_0$, the z direction be orthogonal to the image plane, and the x axis point in the same direction as $\vec{i}_{01}$, where $\vec{i}_{01} = \vec{i}_1 - \vec{i}_0$.

Furthermore, we use the following representation of 3D model points in terms of the three basis model points (see Fig. 1): Originally, the model points lie in some model coordinate system. For any model point $\vec{m}_i$, $i \geq 0$, let r_i be the length of the perpendicular from $\vec{m}_i$ to the infinite line containing $\vec{m}_0$ and $\vec{m}_1$, let d_i be the distance from $\vec{m}_0$ to the intersection of the perpendicular and the line, and let τ_i be the rotation off the basis plane (the plane containing the three basis points).

A view of a 3D model is determined by choosing the six parameters of a weak-perspective transformation that will be applied to the model. (There are two parameters for in-plane translation, three for rotation, and one for scale.) In this section we have fixed $\vec{i}_0$ and $\vec{i}_1$. By fixing the locations in the image where two of the model points project, we have determined four of the transformation's parameters. In particular, we initially can rigidly transform and scale the model so that $\vec{m}_0 = \vec{i}_0$, $\vec{m}_1 = \vec{i}_1$, and $\vec{m}_2$ is in the $z = 0$ plane—in so doing, the model

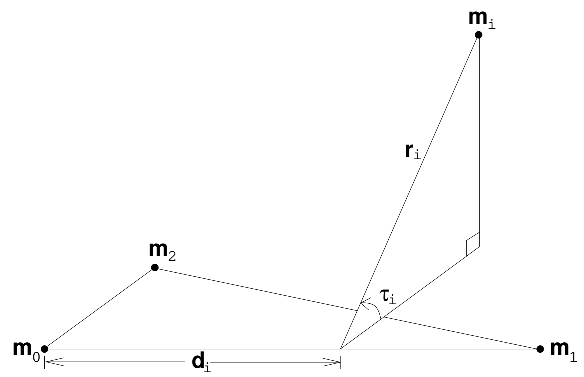

Figure 1: Every model point has unique (d_i, r_i, τ_i) coordinates, unless it is on the line through $\vec{m}_{01}$, where τ_i is unrestricted. Note that $\tau_i = 0$ for points in the basis plane.

is scaled by

$$s_0 = \frac{\| \vec{i}_{01} \|}{\| \vec{m}_{01} \|}\,. \tag{4}$$

In order to keep $\vec{m}_0$ and $\vec{m}_1$ projecting onto $\vec{i}_0$ and $\vec{i}_1$, respectively, there can be no further in-plane translation nor in-plane rotation. As shown in Fig. 2, we still are free to rotate about the y axis as long as $\vec{m}_1$ continues to project onto $\vec{i}_1$, which means that any such rotation about the y axis determines the scale factor. After rotating about the y axis and rescaling, the only remaining degree of freedom is a rotation about the vector $\vec{m}_{01}$.

Next we derive an expression for the image position of $\vec{m}_2$ as a function of the two free parameters. As illustrated in Fig. 2, the model is scaled by s, then rotated by ϕ about the x axis, and then rotated by θ about the y axis (denoted by $\mathbf{R}_{\{\theta,y\}}$). This aligns the projections of the three model points with their corresponding image points. As in Fig. 1, we let $\vec{m}_2$'s coordinates relative to the basis model points be $(d_2, r_2, \tau_2) = (d, r, 0)$—the last element is 0 since $\vec{m}_2$ is in the basis plane. So $\vec{m}_2 = \mathbf{R}_{\{\theta,y\}}(sd, sr\cos\phi, sr\sin\phi)$, where $\phi \in [0, 2\pi)$, which gives

$$\vec{m}_2 \;=\; \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} sd \\ sr\cos\phi \\ sr\sin\phi \end{bmatrix} \;=\; s \begin{bmatrix} d\cos\theta - r\sin\theta\sin\phi \\ r\cos\phi \\ d\sin\theta + r\cos\theta\sin\phi \end{bmatrix} \tag{5}$$

By our choice of image coordinate system, $\theta \in [0, \pi/2)$. Project orthographically:

$$\vec{m}_2^{\pi} = (x, y) = (sd\cos\theta, 0) + sr(-\sin\theta\sin\phi, \cos\phi).$$

As $\vec{m}_1$ rotates around the y axis, the scale factor is constrained to keep $\vec{m}_1$ projecting onto

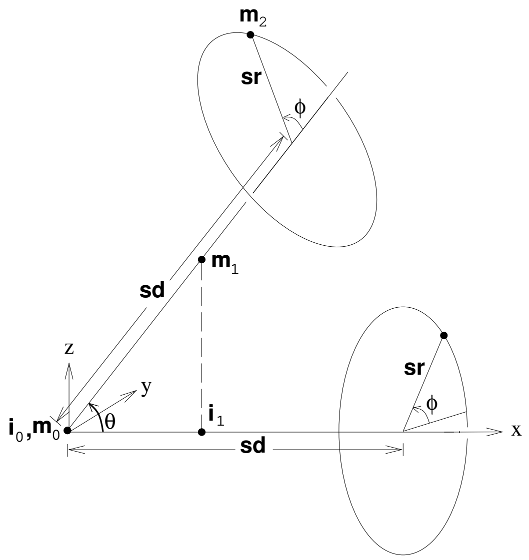

Figure 2: The out-of-plane rotations: a rotation about the y axis and a rotation about the vector $\vec{m}_{01}$.

$\vec{i}_1$. From Fig. 2, this constraint is

$$s \parallel \vec{m}_{01} \parallel \cos\theta = \parallel \vec{i}_{01} \parallel \quad \Longrightarrow \quad s = \frac{\parallel \vec{i}_{01} \parallel}{\parallel \vec{m}_{01} \parallel \cos\theta} = \frac{s_0}{\cos\theta} \tag{6}$$

Consequently,

$$(x, y) = (s_0 d, 0) + s_0 r \left(-\tan\theta \sin\phi,\ \frac{\cos\phi}{\cos\theta} \right), \tag{7}$$

which gives the image position of $\vec{m}_2$ as a function of the "out-of-plane" rotation angles θ and ϕ. Fig. 3 shows graphically the successive rotations of $\vec{m}_2$ by θ and ϕ, followed by the orthographic projection.

By setting (x, y) in Equation 7 to the nominal location of $\vec{i}_2$, we could solve for the nominal values of θ and ϕ. If done, this would provide a solution to the problem of recovering 3D pose from three corresponding points. There are several solutions to this problem already, however, and instead we could apply one of them and then use the solution to compute θ and ϕ (see [3] for a review of the solutions).

2.2 First-Order Approximation

Next we allow for error in one of the basis points, $\vec{i}_2$. The problem is to determine how $\vec{m}_3^\pi$ changes as a function of $\vec{i}_2$, with $\vec{i}_0$ and $\vec{i}_1$ remaining fixed. In Section 2.4, we explain how to

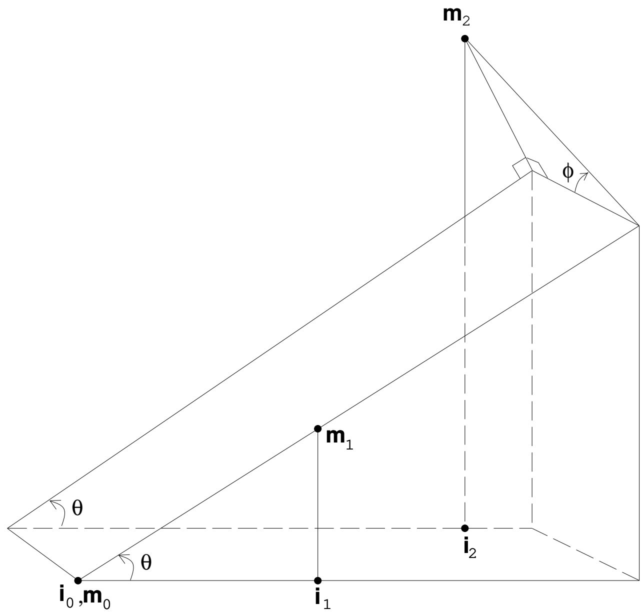

Figure 3: The image position of $\vec{m}_2$ is a function of the out-of-plane rotations θ and ϕ.

extend the result to the case where all three points can move. The out-of-plane rotations θ and ϕ, and the scale, will change as $\vec{i}_2$ moves in the plane. If the changes in θ and ϕ are sufficiently small, then the changes in x and y as a function of θ and ϕ will be given by their first derivatives. From Equation 7,

$$\frac{\partial x}{\partial \theta} = -s_0 r \frac{1}{\cos^2\theta}\sin\phi, \qquad \frac{\partial y}{\partial \theta} = s_0 r \frac{\sin\theta}{\cos^2\theta}\cos\phi \tag{8}$$

$$\frac{\partial x}{\partial \phi} = -s_0 r \frac{\sin\theta}{\cos\theta}\cos\phi, \qquad \frac{\partial y}{\partial \phi} = -s_0 r \frac{1}{\cos\theta}\sin\phi \tag{9}$$

$\overrightarrow{t_\theta} = \left(\frac{\partial x}{\partial \theta}, \frac{\partial y}{\partial \theta}\right)$ and $\overrightarrow{t_\phi} = \left(\frac{\partial x}{\partial \phi}, \frac{\partial y}{\partial \phi}\right)$ are the tangent vectors at the point (x, y) to the image curves that are traced out by changing θ and ϕ.

$$\| \overrightarrow{t_\theta} \| = \frac{s_0 r}{\cos^2\theta}\sqrt{\sin^2\phi + \sin^2\theta\cos^2\phi} \tag{10}$$

$$\| \overrightarrow{t_\phi} \| = \frac{s_0 r}{\cos\theta}\sqrt{\sin^2\phi + \sin^2\theta\cos^2\phi} \tag{11}$$

For a small change in θ, let α_θ represent the direction in which the image point $\vec{i}_2$ moves, measured counter-clockwise from the x axis, and similarly for ϕ and α_ϕ. Then

$$\alpha_\theta = \tan^{-1}\left(\frac{\partial y}{\partial \theta}, \frac{\partial x}{\partial \theta}\right) = \tan^{-1}(\sin\theta\cos\phi, -\sin\phi) \tag{12}$$

$$\alpha_\phi = \tan^{-1}\left(\frac{\partial y}{\partial \phi}, \frac{\partial x}{\partial \phi}\right) = \tan^{-1}(-\sin\phi, -\sin\theta\cos\phi) \tag{13}$$

The dot product of the arguments to the inverse tangent is 0, and so the tangent vectors $\overrightarrow{t_\theta}$ and $\overrightarrow{t_\phi}$ are perpendicular. We can use the normalized cross product of the arguments to see whether the angle between $\overrightarrow{t_\theta}$ and $\overrightarrow{t_\phi}$ is $\pm 90^\circ$:

$$\sin(\alpha_\phi - \alpha_\theta) = \frac{(-\sin\phi)(-\sin\phi) - (\sin\theta\cos\phi)(-\sin\theta\cos\phi)}{\parallel (-\sin\phi, \sin\theta\cos\phi) \parallel \parallel (-\sin\theta\cos\phi, -\sin\phi) \parallel} = 1$$

Thus $\alpha_\phi = \alpha_\theta + 90^\circ$.

2.3 Relative Error between the Third and Fourth Points

We still are considering the case where only $\vec{i}_2$ moves. This section shows that if the projected third model point moves by a small amount, then any projected fourth model point moves by a constant factor times that small amount, and in an analogous direction. We also show that the two weak-perspective solutions lead to different constant factors.

Equations 10 and 11 give the magnitude of the changes in $\vec{m}_2^\pi$ for small changes in θ and ϕ. The only differences for $\vec{m}_3^\pi$ are in the relative coordinates of the model points (Fig. 1). For $\vec{m}_2$, the coordinates are $(r_2, d_2, \tau_2) = (r, d, 0)$. For $\vec{m}_3$, let the coordinates be $(r_3, d_3, \tau_3) = (r', d', \tau')$. Consequently, in the expressions, we change $r \rightarrow r'$ and $d \rightarrow d'$. Further, θ and s_0 do not change since they are measured with respect to $\vec{m}_0$ and $\vec{m}_1$, and do not depend on whether we are considering $\vec{m}_2$ or $\vec{m}_3$. However, ϕ is the amount of rotation of $\vec{m}_2$ away from the plane of the wedge in Fig. 3. For any other model point, $\vec{m}_3$, the amount of rotation away from the wedge is given by $\phi + \tau'$, since, according to Fig. 1, τ' is the amount of rotation of $\vec{m}_3$ away from $\vec{m}_2$. Hence for $\vec{m}_3$, we change $\phi \rightarrow \phi + \tau'$. All together, we get

$$\parallel \overrightarrow{t'_\theta} \parallel = \frac{s_0 r'}{\cos^2\theta}\sqrt{\sin^2(\phi+\tau') + \sin^2\theta\cos^2(\phi+\tau')} \tag{14}$$

$$\parallel \overrightarrow{t'_\phi} \parallel = \frac{s_0 r'}{\cos\theta}\sqrt{\sin^2(\phi+\tau') + \sin^2\theta\cos^2(\phi+\tau')} \tag{15}$$

Note that $\parallel \overrightarrow{t'_\theta} \parallel / \parallel \overrightarrow{t_\theta} \parallel = \parallel \overrightarrow{t'_\phi} \parallel / \parallel \overrightarrow{t_\phi} \parallel$. Therefore, when either ϕ or θ changes, the ratio of the size of the change in $\vec{m}_3^\pi$ to the size of change in $\vec{m}_2^\pi$ equals

$$S = \frac{r'}{r}\sqrt{\frac{\sin^2(\phi+\tau') + \sin^2\theta\cos^2(\phi+\tau')}{\sin^2\phi + \sin^2\theta\cos^2\phi}} \tag{16}$$

Note that due to θ and ϕ, this expression depends on the viewpoint the model is observed from, which is not true in the planar case [28]. In the planar case $\tau' = 0$, because all model points are in the basis plane (Fig. 1), and Equation 16 reduces to $\frac{r'}{r}$.

Recall that there are two reflective solutions to the weak-perspective geometry [2]. The two solutions correspond to a reflection of the basis model points about the image plane. From Fig. 3, this reflection corresponds to negating θ and ϕ. Plugging into Equation 16, the constant scaling factors for the two solutions are

$$\begin{aligned} & \frac{r'}{r}\sqrt{\frac{\sin^2(\sigma\phi+\tau') + \sin^2(\sigma\theta)\cos^2(\sigma\phi+\tau')}{\sin^2(\sigma\phi) + \sin^2(\sigma\theta)\cos^2(\sigma\phi)}}, \quad \text{where } \sigma = \pm 1 \\ = \; & \frac{r'}{r}\sqrt{\frac{\sin^2(\sigma\phi+\tau') + \sin^2\theta\cos^2(\sigma\phi+\tau')}{\sin^2\phi + \sin^2\theta\cos^2\phi}} \end{aligned} \tag{17}$$

Thus the two weak-perspective solutions give different scaling constants. Again, this differs from the planar case, in which the scaling constant in both cases is $\frac{r'}{r}$. More generally, in the planar case the two solutions collapse to one when projected onto the image, and so the existence of two solutions makes no difference.

From Equations 12 and 13,

$$\alpha'_\theta = \tan^{-1}(-\sin(\phi+\tau'), \sin\theta\cos(\phi+\tau')) \tag{18}$$

$$\alpha'_\phi = \tan^{-1}(-\sin\theta\cos(\phi+\tau'), -\sin(\phi+\tau')) \tag{19}$$

Through the same calculation that showed $\sin(\alpha_\phi - \alpha_\theta) = 1$, we can calculate that $\sin(\alpha'_\phi - \alpha'_\theta) = 1$, so that

$$\alpha'_\phi - \alpha'_\theta = \alpha_\phi - \alpha_\theta = 90^\circ. \tag{20}$$

Thus the angles between the tangent vectors and their relative sizes are the same for $\vec{m}_2^\pi$ and $\vec{m}_3^\pi$. As a note, this implies that the mapping between curves traced out by changing θ and ϕ for $\vec{m}_2^\pi$ and $\vec{m}_3^\pi$ is *conformal* [1].

Since we are making a first-order approximation, any movement of $\vec{m}_2^\pi$ in the image plane can be viewed as the sum of the effects of changes in θ and ϕ. From $\| \overrightarrow{t'_\theta} \| / \| \overrightarrow{t_\theta} \| = \| \overrightarrow{t'_\phi} \| / \| \overrightarrow{t_\phi} \|$, we see that for any change in $\vec{m}_2^\pi$ by some small amount, there is a change in $\vec{m}_3^\pi$ by that amount times a constant, given by Equation 16. Furthermore, as θ changes, $\vec{m}_2^\pi$ and $\vec{m}_3^\pi$ each moves in some direction, by some amount. Then as ϕ changes, Equation 20 implies that the two points move at right angles to their previous directions. Hence any change in $\vec{m}_2^\pi$ produces a change in $\vec{m}_3^\pi$ that is scaled and rotated by fixed amounts. Consequently, any error region about the nominal position of $\vec{m}_2^\pi$ results in a mathematically similar error region about the nominal position of $\vec{m}_3^\pi$, which means they are related by an image plane translation, rotation, and scaling.

We can explicitly write the relationship between the errors in $\vec{m}_2^\pi$ and $\vec{m}_3^\pi$ using a 2×2 scaled rotation matrix $\mathbf{A}$. ($\mathbf{A}$ is a similarity transform with zero translation. In the sequel we will refer to $\mathbf{A}$ interchangeably as a scaled rotation matrix or a similarity transform.) $\mathbf{A}$ must satisfy

$$\overrightarrow{t'_\theta} = \mathbf{A}\,\overrightarrow{t_\theta} \quad \text{and} \quad \overrightarrow{t'_\phi} = \mathbf{A}\,\overrightarrow{t_\phi}, \tag{21}$$

which gives four equations in four unknowns. Actually only two of the equations are needed: In general, let a_{11}, a_{12}, a_{21}, and a_{22} be the elements of a 2×2 matrix $\mathbf{A}$. Then for a similarity transform, $a_{21} = -a_{12}$ and $a_{22} = a_{11}$. Solving the equations leads to (Appendix A)

$$\begin{aligned} a_{11} &= k(\cos\tau' - \cos^2\theta\cos\phi\cos(\phi+\tau')) && (22)\\ a_{12} &= -k\sin\tau'\sin\theta, && (23)\\ \text{where} \quad k &= \left(\frac{r'}{r}\right)\frac{1}{1-\cos^2\theta\cos^2\phi}\,. && (24) \end{aligned}$$

Note that the constant scale S from Equation 16 must equal $\sqrt{a_{11}^2 + a_{12}^2}$.

2.4 General Formula for the Fourth Point Error

We can use Equations 22-24 to obtain a formula for the error in $\vec{m}_3^\pi$ as a function of the error in $\vec{i}_0$ or $\vec{i}_1$ in the same way. This gives three scaled rotation matrices relating the individual errors in the basis points to the error in $\vec{m}_3^\pi$. Under a first-order approximation, these errors affect the error in $\vec{m}_3^\pi$ independently and the total error in $\vec{m}_3^\pi$ is the sum of the individual errors. Let $\mathbf{A}$

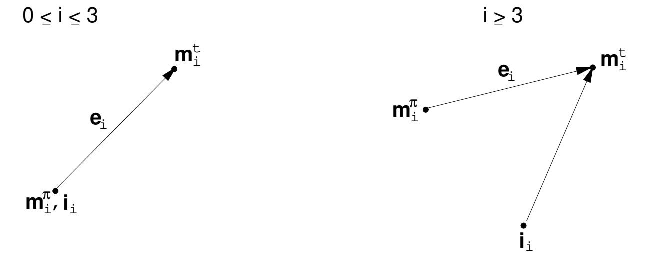

Figure 4: For each model point $\vec{m}_i$, there are three points of interest in the image: (1) its nominal (detected) position $\vec{i}_i$, which is determined by the feature detector, (2) its no-error (true) position $\vec{m}_i^t$, which would equal $\vec{i}_i$ if there were no error, and (3) its predicted position $\vec{m}_i^\pi$, which is computed using the first three point pairs to compute the pose and project $\vec{m}_i$ into the image. For $i \leq 3$, $\vec{m}_i^\pi = \vec{i}_i$. For any i, we define $\vec{e}_i$ to be the correction vector from $\vec{m}_i^\pi$ to $\vec{m}_i^t$.

be the scaled rotation matrix between $\vec{i}_0$ and $\vec{m}_3^\pi$, $\mathbf{B}$ be the scaled rotation matrix between $\vec{i}_1$ and $\vec{m}_3^\pi$, and $\mathbf{C}$ be the scaled rotation matrix between $\vec{i}_2$ and $\vec{m}_3^\pi$. Also let $\vec{e}_0$, $\vec{e}_1$, and $\vec{e}_2$ be the errors in the basis points, and, for $i \geq 3$, let $\vec{e}_i$ be the error vector from the nominal position of $\vec{m}_i^\pi$ to its true position (see Fig. 4). Then the error in $\vec{m}_3^\pi$ is given by the following linear relationship:

$$\vec{e}_3 = \mathbf{A}\vec{e}_0 + \mathbf{B}\vec{e}_1 + \mathbf{C}\vec{e}_2 \tag{25}$$

The two weak-perspective pose solutions, $\pm(\theta, \phi)$, lead to two possibilities for each of $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, and for the error in the fourth point $\vec{e}_3$. When we combine the errors in Equation 25, we must be sure to use the same weak-perspective solution for the three matrices.

Suppose now that the error in each image point is bounded by some amount ϵ. This results in some bounded uncertainty region about $\vec{m}_3^\pi$. From Section 2.3, for each image point separately its ϵ-circle propagates to a circle around $\vec{m}_3^\pi$ that is scaled by S (Equation 16), which gives a radius of $S\epsilon$. The error in each image point affects the fourth point independently, and so the uncertainty region around the fourth point is a circle centered at $\vec{m}_3^\pi$, and the three radii simply sum together (where we must be careful to use the radii from the same weak-perspective solution). In Equation 25, let $S_0 = \sqrt{a_{11}^2 + a_{12}^2}$, $S_1 = \sqrt{b_{11}^2 + b_{12}^2}$, and $S_2 = \sqrt{c_{11}^2 + c_{12}^2}$. Then the radius of the uncertainty circle for $\vec{m}_3^\pi$ is

$$R = S_0\epsilon_0 + S_1\epsilon_1 + S_2\epsilon_2 + \epsilon_3, \tag{26}$$

where ϵ_3 is added to account for error in sensing $\vec{i}_3$ (an image point that corresponds to $\vec{m}_3^\pi$).

When the error in the image points is normally distributed, the linear relationship allows us to determine the propagated uncertainty distribution about any fourth point. If the Gaussian error in the ith image point has standard deviation σ_i, then the uncertainty distribution about $\vec{m_3}^\pi$ is normally distributed with standard deviation (Appendix C)

$$\sigma = \sqrt{S_0^2\sigma_0^2 + S_1^2\sigma_1^2 + S_2^2\sigma_2^2 + \sigma_3^2}. \tag{27}$$

In summary, we have explained why Alter and Grimson found that the uncertainty regions for three matched points are circular. Moreover, we have provided a simple, analytic expression for those uncertainty regions. To illustrate the value of this expression, we outline a robust version of Huttenlocher and Ullman's 3D-from-2D alignment algorithm [27]. Given a model and a cluttered image of a scene containing the model, the following steps are repeated until the model is identified:

1. Hypothesize a pairing of three model and image points.
2. Using the hypothesis, project all of the model line segments into the image.
3. Use Equation 26 to compute the uncertainty circles for the two endpoints of every projected line segment. Then construct a tight overestimate of every line segment's uncertainty region from the two uncertainty circles, as in Alter and Grimson [4] (see Fig. 5).
4. Accept or reject the hypothesis, based on the number of line segments for which there exist candidate image segments, and on the sizes of the uncertainty regions (as in [4]). If accepted, return the identified model and pose.

[4] demonstrated that this algorithm is expected to be insensitive to false positives in cluttered scenes.

3 A Study of Uncertainty from One Basis Point

We showed in Section 2 that for any shape traced out by the third model point, the fourth model point traces out a mathematically similar shape, up to an approximation. This section provides

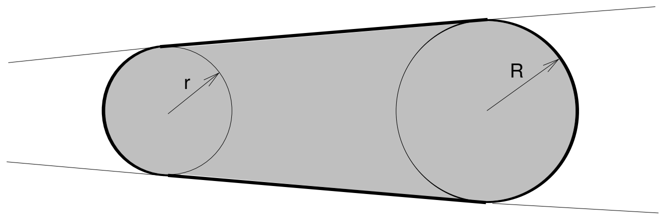

Figure 5: Region to search for candidate line segments.

a study of the true shape of the uncertainty region. We begin by considering exactly how larger changes in θ and ϕ effect the appearance of each additional model point. Using Equation 7, it is straightforward to see that, as ϕ changes with θ held constant, (x, y) traces out an ellipse, with center at $(s_0 d, 0)$ and with the major axis parallel to the y axis. We rewrite this equation as

$$(x, y) = (s_0 d, 0) + (-a \sin\phi, b \cos\phi), \tag{28}$$

with minor axis $a = s_0 r \tan\theta$, major axis $b = s_0 r \sec\theta$, and eccentricity $e = \sqrt{b^2 - a^2} = s_0 r$. For $\theta = 0$, the ellipse equation becomes $(x, y) = (s_0 d, 0) + (0, s_0 r \cos\phi)$, which forms a line segment between the points $(s_0 d, s_0 r)$ and $(s_0 d, -s_0 r)$. As θ increases from 0, the center of the ellipse is unchanged; Fig. 6 shows the 3D interpretation. In addition, as θ increases both axes a and b grow, and the ratio $a/b = \sin\theta$ approaches 1. Consequently increasing θ sweeps out a growing family of concentric ellipses that become increasingly circular.

As θ changes, (x, y) traces out a hyperbola, which we can see by eliminating θ from Equation 7. From the x and y coordinates, we have respectively:

$$\left(\frac{x - s_0 d}{s_0 r \sin\phi}\right)^2 = \tan^2\theta, \qquad \text{and} \qquad \left(\frac{y}{s_0 r \cos\phi}\right)^2 = \sec^2\theta.$$

Using $1 + \tan^2\theta = \sec^2\theta$,

$$-\left(\frac{x - s_0 d}{s_0 r \sin\phi}\right)^2 + \left(\frac{y}{s_0 r \cos\phi}\right)^2 = 1. \tag{29}$$

This equation is a hyperbola centered at $(s_0 d, 0)$, with focii $(s_0 d, \pm s_0 r)$, vertices $(s_0 d, \pm s_0 r \cos\phi)$, and asymptotes $x = \pm y \tan\phi + s_0 d$. For $\phi = 0, \pi$, the hyperbola becomes $(x, y) = (s_0 d, 0) \pm (0, \frac{s_0 r}{\cos\theta})$, which gives two vertically infinite half-lines starting at the focii. As ϕ increases, the center and focii are unchanged, and the asymptotes rotate about $(s_0 d, 0)$, becoming increasingly

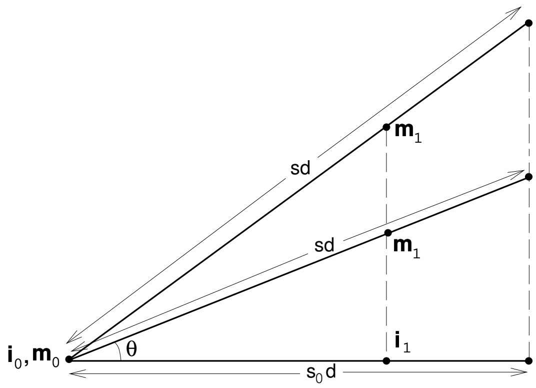

Figure 6: As θ changes, $\vec{m}_{01}$ is scaled so that $\vec{m}_1$ always projects onto $\vec{i}_1$. The figure shows that any point on the line through $\vec{m}_{01}$ always projects onto the same location in the image. When ϕ changes, $\vec{m}_2$ rotates about a point on this line, and that point projects to the ellipse center; therefore the ellipse center does not change.

parallel to the x axis. Consequently, changing ϕ from 0 or π sweeps out a concentric family of hyperbolas that reach the x axis at $\phi = \frac{\pi}{2}, \frac{3\pi}{2}$.

By a simple translation and scale of (x, y), we get the equation

$$(X, Y) = \frac{1}{s_0 r}((x, y) - (s_0, d)) = \left(-\tan\theta\sin\phi,\ \frac{\cos\phi}{\cos\theta}\right). \tag{30}$$

Fig. 7 plots families of curves for changing ϕ and θ for this equation. The elliptical curves are functions of ϕ, and the hyperbolic curves are functions of θ. The four plots show the same figure at different scales and for different ranges of θ. In left-to-right, top-to-bottom order, each plot is a close-up of the center area in the next plot.

The plots in Fig. 7 are for $\vec{m}_2^{\pi}$ translated and scaled. For $\vec{m}_3^{\pi}$ we get

$$(X', Y') = \frac{1}{s_0 r'}((x', y') - (s_0, d')) = \left(-\tan\theta\sin(\phi + \tau'),\ \frac{\cos(\phi + \tau')}{\cos\theta}\right). \tag{31}$$

This involves a different scale factor and translation, but otherwise, the only difference for the fourth model point is in τ'. That is, varying ϕ causes the third and fourth model points to traverse ellipses that may differ in translation and scale, but that do not differ in shape. In Fig. 7, we vary θ and ϕ and show the image regions that the projected points $\vec{m}_2^{\pi}$ and $\vec{m}_3^{\pi}$ occupy as a result. Since we normalize the plots of the two points, as the third model varies along an ellipse, the fourth model point projects onto the same ellipse. If (X, Y) moves along

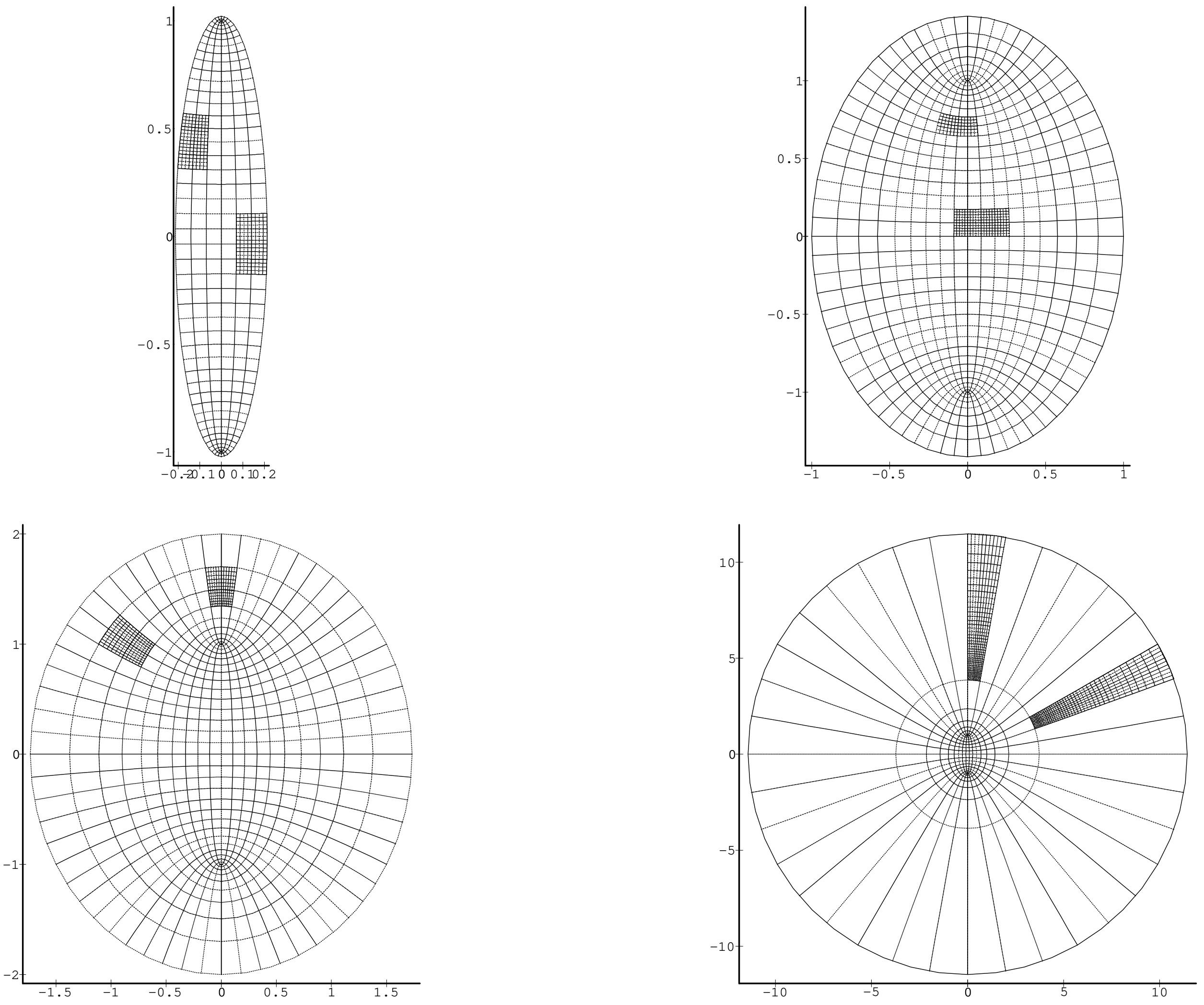

Figure 7: Plots of (X,Y) for different ranges of θ. Upper left: $\theta \in [0,12]$ deg. Upper right: $\theta \in [0,45]$ deg. Lower left: $\theta \in [0,60]$ deg. Lower right: $\theta \in [5,85]$ deg. The curves are separated by changes in angle of 4 deg, 5 deg, 6 deg, and 10 deg (in left-to-right, top-to-bottom order).

the curves in Fig. 7 and traces out a closed region, then so will (X', Y'). The two filled-in areas in each plot illustrate a different pair of bounded error regions that could be traced out simultaneously by some (X, Y) and (X', Y'); (X, Y) or (X', Y') could be either error region. In this case, as one point varies within one of the bounded error regions, the other point will vary within the other error region, illustrating how uncertainty propagates.

In the figure, there are two distinguished points where the grid collapses. These points are the hyperbola focii $(0, \pm 1)$, which were $(s_0 d, \pm s_0 r)$ in the original (x, y) curve. These focii correspond to $(\theta, \phi) = (0, 0)$ and $(0, \pi)$, which occur when the plane containing the basis model points is parallel to the image. From the areas around $(0, 1)$ in Figs. 7 and 8, it is clear that an (X', Y') uncertainty region may not be simply a scaled and rotated version of the (X, Y) region. Therefore, circular error in the third point in general will not produce circular uncertainty, exactly, in the fourth point. In fact, we can see from Fig. 8-left that when θ and ϕ are small, uncertainty in the third image point can lead to strange shapes for the propagated uncertainty region: Suppose that (X, Y) traces out the large region on the right (region A). If $\tau' = 128\,\text{deg}$, then (X', Y') traces out the similar-looking, large region on the left (region B). But if $\tau' = 72\,\text{deg}$, then (X', Y') traces out the non-convex, curved region on top (region C). It should be kept in mind that the shapes not the sizes of these regions are what matters here, since the plot is normalized: After being scaled by $\frac{r'}{r}$, the region on top may be significantly larger than then the region on the right, but its unusual shape would be unchanged. As a result, odd-shaped uncertainty regions can occur even when there is little error in the basis points.

When the third point's error region contains one of the focii, then the two uncertainty regions for the fourth point's two weak-perspective pose solutions are one and the same; otherwise the regions will be distinct, although they may overlap. As an example, in Fig. 8-left suppose now that region C is traced out by (X, Y). If $\tau' = 72\,\text{deg}$, then the two large regions correspond to the two weak-perspective solutions for the (X', Y') region, obtained by alternating the sign of (θ, ϕ). If $\tau' = 8\,\text{deg}$, then the two uncertainty regions for the two solutions would overlap, but they still would be distinct. On the other hand, suppose that the region traced by (X, Y) additionally includes the point $(0, 1)$, as in Fig. 8-right (region D). Then when $\tau' = 72\,\text{deg}$ the two large regions merge into the single, "H"-shaped uncertainty region shown in the figure

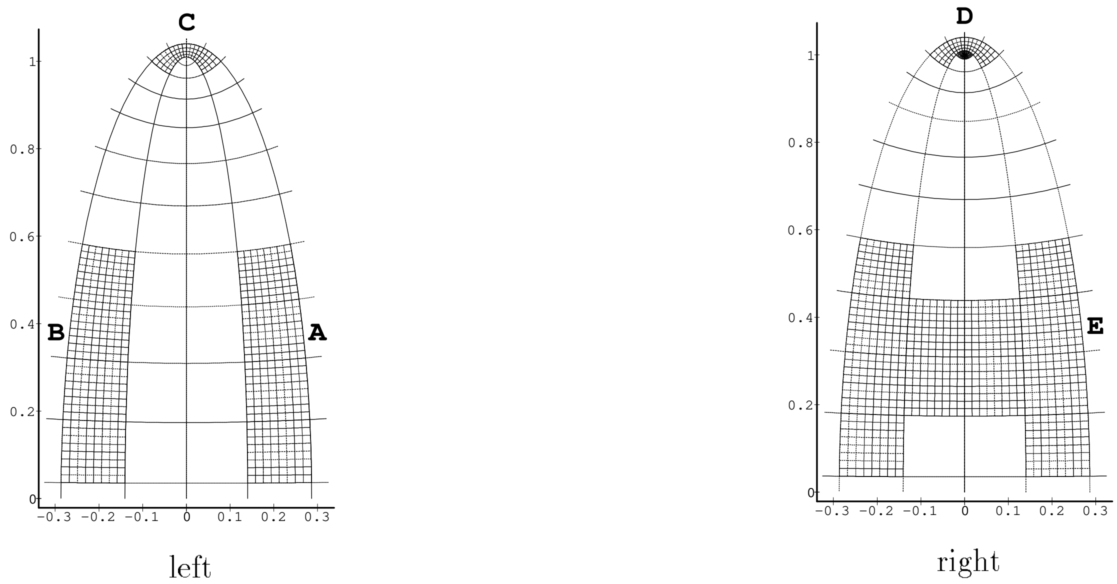

Figure 8: Close-up where $\phi \in [0, 90]$ deg. In both plots, $\theta \in [0, 16]$ deg and the curves are separated by changes in θ and ϕ of 8 deg.

(region E). So we see that the similarity transform can be a poor approximation for poses with small values of θ and ϕ even with small amounts of error.

For large θ (bottom, right picture in Fig. 7), the ellipses become concentric circles, and the hyperbolas become straight lines. In this circumstance, any (X', Y') region is the same as the corresponding (X, Y) region, except for a rotation about the origin. In this case, a similarity transform will exactly relate the errors, and will be independent of the model pose (as it was for planar objects).

Another conclusion we can draw applies when the third point has a bounded error region that is very large. Suppose there is circular error of radius ϵ in the third point. As ϵ grows, the error circle will include the point where the two pose solutions merge, and the boundaries of the error circle and the fourth point's propagated uncertainty region will reach the range of θ where they are related by a similarity transformation. For large enough ϵ, then, the error in the third point will result in a single, circular uncertainty region for the fourth point, regardless of where the image and model points are nominally located. This is surprising because one might expect that the error incurred by using the similarity-transform approximation grows as ϵ grows. Even

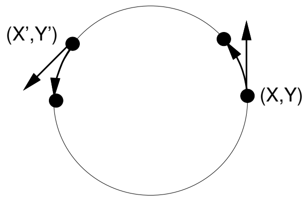

Figure 9: For large θ, the ellipses traced out by changing ϕ are circles. In this case, changing ϕ causes movements in (X, Y) and (X', Y') that are exactly related by a similarity transform. First-order approximations to the movements, however, assume that the two points move along the tangent lines at (X, Y) and (X', Y'), which is correct only if the movements are small.

though this may be true when ϵ is small, for large enough ϵ the error will decrease.

Finally, the above discussion shows that the similarity-transform approximation may hold further than the two first-order approximations that we used to compute it. To obtain an analytic expression for the propagated uncertainty, Section 2 took first-order approximations to the errors in the third and fourth points in terms of the 3D pose parameters, θ and ϕ. Then these approximations were combined to get a similarity transform that directly relates the errors in the points. This similarity transform may hold further than the two first-order approximations. For instance, for high values of θ, where the similarity transform holds exactly, the first-order approximations do not (see Fig. 9). This suggests that higher-order error terms may be cancelling when we combine the two first-order approximations to get a similarity transform. Section 4.3 empirically shows that the similarity transform can hold further than the first-order approximation. This indicates that it may be wise to propagate errors directly in image space rather than from transformation space, as was done for example in [21].

4 Experiments

We have run three experiments to test the results in Sections 2 and 3. The first experiment compares our results to those of Alter and Grimson [4], who studied the case of bounded, ϵ error. In particular, we test our formula for the radius of the fourth point error circle. The second experiment looks more generally at the accuracy of the similarity transforms in Equation 25 for predicting where the fourth point moves when there is error in the basis points. This experiment is repeated for uniform and Gaussian error. The third experiment compares the accuracy of the similarity transform with the first order approximation used to derive it.

4.1 Comparison to past results

Alter and Grimson assumed bounded error in the image points using circles of radius ϵ. They showed experimentally that the uncertainty region for a fourth model point is closely approximated by a circle centered at the nominal point, which is the uncertainty region we derived analytically in Section 2.4. Following Alter and Grimson, we refer to these fourth-point error circles as *uncertainty circles*. Alter and Grimson computed the radius of an uncertainty circle by densely sampling the fourth-point uncertainty region, and then took the radius to be the maximum distance from the nominal point to a sampled point. As a result, the computed uncertainty circle was an upper bound on the uncertainty region, up to the fineness of the sampling.

To see how our uncertainty circles compare to Alter and Grimson's, we run a series of trials like those in [4]: At each trial, we randomly generate three pairs of model and image points and a set of seven unmatched model points. We let $\epsilon = 5$ pixels. For each unmatched model point, we use Equation 26 to compute our predicted radius (R_f). To compute the "maximum radius" (R_M) as in [4], we take 25 samples along the boundary of each ϵ-circle, and then take all triples between the samples to get 15,625 triples. For each triple, we solve for the model pose and use the pose to project each unmatched model point into the image. This gives 15,625 projected model points per uncertainty region. For each uncertainty region, we compute the maximum distance from the nominal point to any of its projected model points. For an error measure, we use the relative error from our radius to the maximum radius, that is, $\frac{R_M - R_f}{R_f}$.

We tested 1,163 uncertainty circles, over which the average relative error was 1.45%. Table 1 shows the percent of uncertainty circles for which relative error was less than some threshold. The results show that our circles reasonably approximate the maximum-radius circles from Alter and Grimson, who showed that the maximum-radius circles reasonably approximate the true regions. The results also show that the maximum-radius circles can at times overestimate our circles by a significant amount. For instance, 1.2% of the time our circles will be 10% smaller than the maximum-radius circles.

Table 1 could be used to determine a correction factor for the circle radius: Suppose that a

Percent relative error	2	4	6	8	10	12
Percent of uncertainty circles	86.7	94.2	97.1	98.2	98.8	99.0

Table 1: The percent of uncertainty circles for which the relative error between the circles predicted by our formula and by Alter and Grimson's method is less than a given percent.

recognition system has matched three model and image points and is using the analytic solution for the uncertainty circles to decide what region in the image to search for additional matches. Suppose further that we want the system to conservatively estimate the effects of error, so that it can give some guarantee of no false negatives (i.e., that no valid image points will be missed) some high percentage of the time, while at the same time increasing the chance of false positives as little as possible. Then, for a chosen percent of the uncertainty circles, the percent relative error in the table tells us by how much to increase the radii of the circles so that all points in the true uncertainty region are included.

4.2 Accuracy of the similarity transform

Section 2.4 showed how we can approximate the effects of error in the basis points using three similarity transforms. Equation 25 gives the error in the fourth point as a function of the errors in the basis points. The next experiment estimates how well this approximation works when the error in the image points is distributed uniformly, or according to a Gaussian. We run a series of trials like those in the previous experiment, where at each trial we generate a random model and a random image triple. Assuming uniform error for now, we then uniformly perturb each image point within a circle of radius ϵ. Using the perturbed image points, we compute the distances between where the similarity transforms predict the fourth model point will appear and its actual projected location. For each unmatched model point, there are two such distances corresponding to the two weak-perspective solutions. For each trial, there are seven unmatched model points, giving fourteen distances per trial. Over 10,000 trials, 140,000 unmatched model points were tested.

For uniform error, the average distance between the similarity point (the point predicted by the similarity transforms) and the true point was .06 pixels. Table 2 gives the percent of unmatched points for which the distance is less than various amounts of tolerated error. The

Allowed distance error (px)	1	2	3	4	5
Uniform error (percent)	98.25	99.60	99.79	99.87	99.90
Gaussian error (percent)	98.53	99.64	99.82	99.88	99.91

Table 2: For uniform and Gaussian error, the percent of unmatched model points for which the distance between the similarity point and the true point is less than some tolerated number of pixels.

results show that the similarity transforms very accurately predict the movement of the fourth point, with 98% probability of being in error by less than one pixel.

For Gaussian error, we run the same experiment but instead sample a 2D Gaussian distribution, using a standard deviation of 2.5 pixels as was done by Sarachik and Grimson [39]. We still use a bound of $\epsilon = 5$ pixels on the allowed error in the image points to guarantee that Gaussian error will lead to better results than uniform error. In fact, we might expect the results to be significantly better, since the sampled points will tend to contain less error. It turns out, however, that the results are only slightly better than for uniform (see Table 2), with the same average error of .06 pixels. It appears that for Gaussian error to improve over simple uniform error, the Gaussian distributions would have to be significantly more peaked around their nominal positions.

In conclusion, the two experiments on random model and image points indicate that the linear approximation is reasonably accurate for up to five pixels of error. Fig. 10 shows circular uncertainty regions that we computed using a real model and image of a telephone. By hand, we measured corners of the telephone to obtain a model and selected corresponding corners in the image. The three smaller circles are the error bounds for the matched image points. The remaining circles are the correct search regions for finding additional matches.

4.3 Similarity transform vs. a first-order approximation

We have used two first order approximations. One relates changes in $\vec{m}_2^\pi$ to changes in pose. The second relates changes in pose to changes in $\vec{m}_3^\pi$. When we use both approximations, we obtain a similarity transform. We will now compare this to the results of using an exact determination of the effect of changes in $\vec{m}_2^\pi$ on pose, along with a first-order approximation to the subsequent effect of pose changes on $\vec{m}_3^\pi$. If the effects of these two first-order approximations

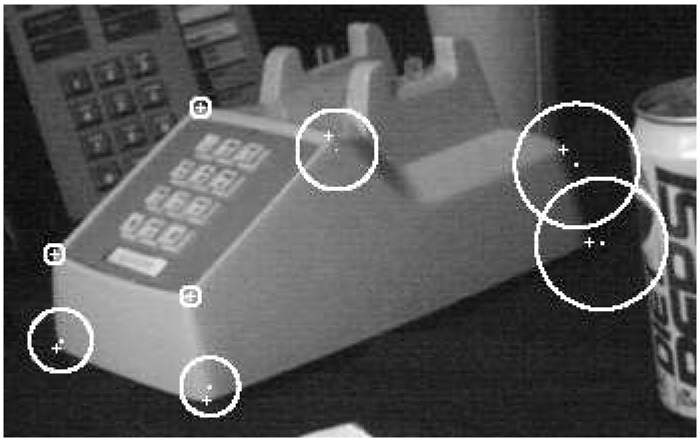

Figure 10: Circular uncertainty regions computed for points at the corners of a telephone. The three smaller circles show error regions of five pixels about the three points used to generate the model pose. The remaining, larger circles show uncertainty circles. Small crosses show the actual locations of the image points, which were selected by hand but are still a bit noisy. Small dots show the projected locations of the model points in the determined pose.

are uncorrelated, we would expect to obtain more accurate results when we replace one with an exact value. On the other hand, if error in the first-order approximations tends to cancel, the similarity transform that follows from both approximations will tend to be more accurate. Section 3 showed analytically that this can happen in some circumstances. We now explore this possibility experimentally.

In the same way as in the previous experiment, we generate trials of random models and image triples, and project the random unmatched model points into the image. As in Section 3, we put error in only one of the basis points, say $\vec{i}_2$. To better compare the first-order and similarity approximations, at each trial we generate an error basis point as follows: We change either ϕ or θ from its value at the nominal pose until $\vec{i}_2$ moves 5 pixels from its nominal position. This gives us an error vector in the third image point. For each of the projected, unmatched model points we compute the scaled rotation matrix $\mathbf{A}$ as in Equation 21, and apply it to this error vector. This gives the error point predicted by the similarity transform.

To compute the error point predicted by a first-order approximation from pose space, we move the projected point along one of the tangent lines $\overrightarrow{t'_\phi}$ or $\overrightarrow{t'_\theta}$, defined in Section 2.3. The amount we move is determined by which pose parameter we changed to compute the error in $\vec{i}_2$.

Absolute error (px)	99%: Changing	ϕ	θ	90%: Changing	ϕ	θ
Similarity trans.		0.57	0.51		0.09	0.09
1st-order approx.		1.02	1.67		0.26	0.45

Table 3: The amount of error allowed in the fourth point that includes 99% or 90% of the tested points. The error is the distance from the true location of the fourth point to the location predicted by the similarity transform (which is the approximation we suggest) or the location predicted by the first-order approximation. In this experiment there was error in one basis point, which was moved 5 pixels by changing either ϕ or θ from its nominal value.

In particular, for changing ϕ the first-order error vector is $\Delta\phi \overrightarrow{t'_\phi}$ and for changing θ it is $\Delta\theta \overrightarrow{t'_\theta}$, where we know $\Delta\phi$ and $\Delta\theta$ exactly from our generation of the error basis point.

To measure propagated error, we first use the error basis point to calculate the two possible locations where the fourth point actually goes. To measure the error in the similarity transform, we calculate the distance from the point predicted by the similarity transform to the closer of the two actual points, and similarly for the first-order approximation. Table 3 shows the results from histogramming the error data over a series of 10,000 trials with 7 projected points per trial. In the table the similarity transform is about two to five times more accurate than the first-order approximation. This suggests that better results may be obtained by propagating error directly from the basis image points to the predicted locations of the fourth points. By first estimating the errors in pose space, and then propagating these errors back to image space, some accuracy may be lost, unless special care is taken to make sure the appropriate high-order error terms cancel.

5 Perspective Projection

In this section we apply the same technique to perspective projection to obtain again a linear relationship between the errors, but in this case the form of the linear relationship is much more complicated. As before, we begin by looking at the effects of error in exactly one of the basis points. We assume we know the camera focal length f and the center of projection $\vec{c}$.

We introduce a similar representation for any third model point to the one we used in the scaled-orthographic case. Since $\vec{i}_0$ and $\vec{i}_1$ are fixed, the line segment between $\vec{m}_0$ and $\vec{m}_1$ is free

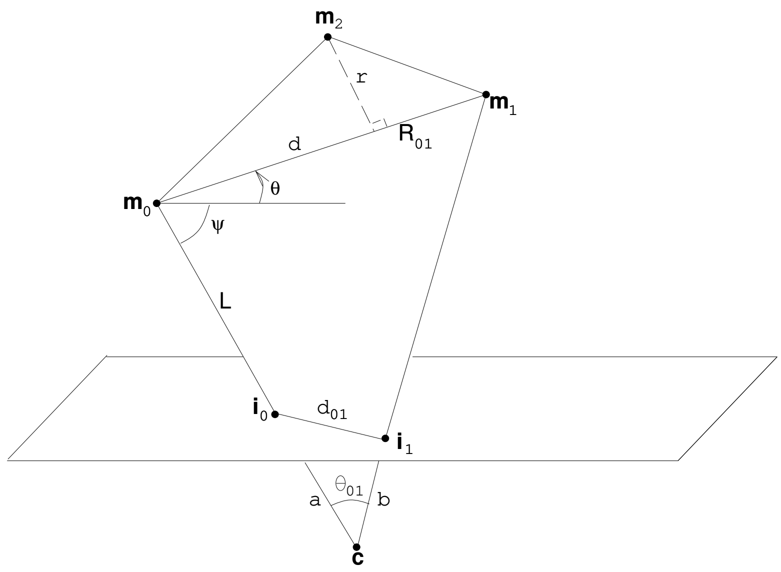

Figure 11: a, b, and L are the distances from $\vec{c}$ to $\vec{i}_0$, $\vec{i}_1$, and $\vec{m}_0$, respectively.

to rotate and translate in the plane through $(\vec{c}, \vec{i}_0, \vec{i}_1)$, as long as $\vec{m}_0$ and $\vec{m}_1$ remain on the lines through $(\vec{c}, \vec{i}_0)$ and $(\vec{c}, \vec{i}_1)$, respectively (see Fig. 11). After this rotation and translation, the only remaining degree of freedom is a rotation about the line through $(\vec{m}_0, \vec{m}_1)$.

Initially, we rigidly transform the model so that $\vec{m}_0 = \vec{i}_0$, $\vec{m}_{01}$ points in the same direction as $\vec{i}_{01}$, and $\vec{m}_2$ is in the $z = 0$ plane. For the image coordinate system, we let the origin be at $\vec{i}_0$, the z direction be orthogonal to the image plane, and the positive x axis be along $\vec{i}_{01}$. Also, we let $\hat{n}$ be the unit vector that is normal to the plane through $(\vec{c}, \vec{i}_0, \vec{i}_1)$. Next the model is rotated by ϕ about the x axis, then rotated by θ about $\hat{n}$ (denoted by $\mathbf{R}_{\{\theta,\hat{n}\}}$), and then translated by $\vec{u}$. Let $\vec{p}$ equal the second model point after the ϕ rotation. Then using the relative coordinates for the model points from Fig. 1, $\vec{p} = (d, r\cos\phi, r\sin\phi)$, so that

$$\vec{m}_2 = \vec{u} + \mathbf{R}_{\{\theta,\hat{n}\}}\vec{p}. \tag{32}$$

For the translation $\vec{u}$, let L be the distance from $\vec{c}$ to $\vec{m}_0$, and let $\hat{v}$ be the unit vector pointing from $\vec{c}$ to $\vec{i}_0 = (0, 0, 0)$, which implies $\hat{v} = -\vec{c}/\parallel \vec{c} \parallel$. Then $\vec{u} = \vec{c} + L\hat{v}$, where L must still be determined. In Fig. 11, let ψ be the angle between $\vec{c} - \vec{i}_0$ and $\vec{i}_1 - \vec{i}_0$, and let θ_{01} be the angle between $\vec{i}_0 - \vec{c}$ and $\vec{i}_1 - \vec{c}$. Also let R_{01} be the distance betweeen $\vec{m}_0$ and $\vec{m}_1$. Through some trigonometry, Appendix B computes that

$$L = \sin(\theta + \psi + \theta_{01}) \left(\frac{R_{01}}{\sin \theta_{01}} \right). \tag{33}$$

Let $\vec{m}_2 = (\overline{x}, \overline{y}, \overline{z})$. Since the image coordinate system is based at $\vec{i}_0$ with the camera center point at $\vec{c} = (c_x, c_y, -f)$, projecting $\vec{m}_2$ into the image gives

$$\vec{m}_2^\pi = (x, y) = \left(f \frac{\overline{x} - c_x}{\overline{z} + f} + c_x,\ f \frac{\overline{y} - c_y}{\overline{z} + f} + c_y \right). \tag{34}$$

From this equation, we can compute the partial derivatives of x and y with respect θ and ϕ, which give $\overrightarrow{t_\theta}$ and $\overrightarrow{t_\phi}$. By substituting r' for r, d' for d, and $\phi + \tau'$ for ϕ, we can analogously compute $\overrightarrow{t'_\theta}$ and $\overrightarrow{t'_\phi}$. Then we can solve Equation 21 for $\mathbf{A}$, and we will get a linear transform relating the error in $\vec{i}_2$ to the error in $\vec{m}_3^\pi$ (see Appendix B): Let $x_\theta = \frac{\partial x}{\partial \theta}$, $y_\theta = \frac{\partial y}{\partial \theta}$, and similarly for x_ϕ, y_ϕ, x'_θ, y'_θ, x'_ϕ, and y'_ϕ. Then

$$\mathbf{A} = \frac{1}{x_\theta y_\phi - x_\phi y_\theta} \begin{bmatrix} y_\phi x'_\theta - y_\theta x'_\phi & -x_\phi x'_\theta + x_\theta x'_\phi \\ y_\phi y'_\theta - y_\theta y'_\phi & -x_\phi y'_\theta + x_\theta y'_\phi \end{bmatrix}. \tag{35}$$

If there is bounded, circular error in $\vec{i}_2$, the linear transform will produce an elliptical uncertainty region for $\vec{m}_3^\pi$, whose parameters could be determined analytically. This differs from the scaled-orthographic case, where the uncertainty region is a disc. To handle circular error in all three points for scaled-orthographic projection, we had to convolve together three circles. For perspective, we would have to convolve three ellipses. To handle bounded error in three basis points using perspective projection, we can apply the algorithm that will be proposed in Section 6.

For Gaussian error, on the other hand, the method in Appendix C applies equally well to linear transforms as to similarity transforms. As before, we get an analytic solution for the uncertainty region of a projected model point, and that uncertainty region is normally distributed. The only difference is that the normal distribution need not be circularly symmetric.

6 nth-Point Uncertainty Region

It has been shown [21] that recognition algorithms that use a small number of randomly-matched points to determine pose are sensitive to false positive identifications of objects, because these poses are not sufficiently stable and lead to large uncertainty regions. One solution is to use poses based on more information to derive smaller uncertainty regions. Assuming a bounded error model, this section shows how to compute the uncertainty region of an $n+1$'st model point given n matched model and image points, for any value of n. Our linearized models of projection allow us to determine linear constraints on the set of feasible error vectors consistent with a match between model and image points. Having expressed our knowledge about pose using linear constraints, we apply linear programming to optimize a set of objective functions whose solution tightly bounds the uncertainty region. In general, this technique applies to any linear projection model, including affine models and including the linearized perspective and weak-perspective models that we derived in this paper.

To demonstrate the idea, we suppose that the error in each image point is bounded by a square of width 2ϵ. We emphasize, however, that the same reasoning will apply to any convex, polygonal error bound, so that we may approximate a circle, or any other convex error bound, as closely as we wish. With square error bounds, a match between image and model points can be consistent only if there exists a pose that brings the x- and y-coordinates of all matched points to within ϵ pixels of each other. Let $\vec{m}_i^\pi = (x_i^\pi, y_i^\pi)$ be the projection of the i'th matched model point, in some nominal pose. Let $\vec{i}_i = (x_i, y_i)$ be the location of the corresponding image point. Also, let $\vec{e}_i = (x_i^e, y_i^e)$ be a vector representing the deviation between a model point's projected position in the nominal pose and its true position (see Fig. 4). Since we choose a pose by aligning the first three model points with image points, for $0 \leq i \leq 2$ this deviation is also the actual error that occurred in sensing the image points. For $i \geq 3$, $\vec{e}_i$ does not depend on where we have sensed the i'th image point, but rather is a function of the sensing error in the first three image points. We then model error by assuming a model point's true position, $\vec{m}_i^\pi + \vec{e}_i$, is within ϵ of its corresponding image point, $\vec{i}_i$. That is,

$$x_i - \epsilon \leq x_i^\pi + x_i^e \leq x_i + \epsilon \quad \text{and} \quad y_i - \epsilon \leq y_i^\pi + y_i^e \leq y_i + \epsilon \tag{36}$$

Previously, we used the matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ to represent linear transformations between error in the first three points and the fourth point. We now let $\mathbf{A_i}, \mathbf{B_i}, \mathbf{C_i}$ be the corresponding matrices for the $i+1$'st point (e.g., $\mathbf{A_3} = \mathbf{A}$). We let $\overrightarrow{a_i^1}$ and $\overrightarrow{a_i^2}$ be the first and second rows of the matrix $\mathbf{A_i}$, and define $\overrightarrow{b_i^1}$, $\overrightarrow{b_i^2}$, $\overrightarrow{c_i^1}$, $\overrightarrow{c_i^2}$ similarly. In both the cases of scaled-orthographic and perspective projections, we know that we can write

$$\vec{e}_i = (\overrightarrow{a_i^1} \cdot \vec{e}_0, \overrightarrow{a_i^2} \cdot \vec{e}_0) + (\overrightarrow{b_i^1} \cdot \vec{e}_1, \overrightarrow{b_i^2} \cdot \vec{e}_1) + (\overrightarrow{c_i^1} \cdot \vec{e}_2, \overrightarrow{c_i^2} \cdot \vec{e}_2), \tag{37}$$

for $3 \leq i \leq n-1$. Consequently, for matched points 3 through $n-1$, we may substitute a linear combination of the first three error vectors for $\vec{e}_i$, giving us additional constraints that the first three points must meet in order to lead to a consistent solution. In all, we get the following constraints:

$$\text{For } i \in [0,2], \qquad -\epsilon \leq x_i^e \leq \epsilon, \qquad\qquad -\epsilon \leq y_i^e \leq \epsilon, \tag{38}$$

$$\text{For } i \in [3, n-1], \quad \begin{cases} x_i - \epsilon \leq x_i^\pi + \overrightarrow{a_i^1} \cdot \vec{e}_0 + \overrightarrow{b_i^1} \cdot \vec{e}_1 + \overrightarrow{c_i^1} \cdot \vec{e}_2 \leq x_i + \epsilon, \\ y_i - \epsilon \leq y_i^\pi + \overrightarrow{a_i^2} \cdot \vec{e}_0 + \overrightarrow{b_i^2} \cdot \vec{e}_1 + \overrightarrow{c_i^2} \cdot \vec{e}_2 \leq y_i + \epsilon. \end{cases} \tag{39}$$

This set of constraints can be satisfied if and only if there is a set of error vectors for the first three points that will bring all projected model points to within the error bounds of their matching image points.

We have formulated our knowledge of model pose, based on a match between n image and model points, in terms of linear constraints on the components of the three error vectors $\vec{e}_0$, $\vec{e}_1$, and $\vec{e}_2$. We may now use linear programming to maximize any linear objective function, subject to these constraints. In particular, we can formulate linear objective functions that express, for several directions, the errors in an additional model point's predicted position, and then extremize these functions. For example, if we maximize the linear objective function

$$x_n^e = \overrightarrow{a_n^1} \cdot \vec{e}_0 + \overrightarrow{b_n^1} \cdot \vec{e}_1 + \overrightarrow{c_n^1} \cdot \vec{e}_2, \tag{40}$$

we will find the maximum x displacement that the projection of the $n+1$'st model point can have from its nominal position. By maximizing the negation of this expression, and similar expressions for the y values, we may put a rectangle about the possible locations of the $n+1$'st point.

Using a similar method, we can in general place a convex polygon of any shape about the possible locations of the $n+1$'st point. Suppose we wish to bound the location of the $n+1$'st point in some direction other than along the x or y axis. Let $(\Delta x, \Delta y)$ be a vector in that direction. Then we may achieve this by maximizing the objective function formed by taking the dot product of $(\Delta x, \Delta y)$ and $\vec{e}_n$. By substituting our expression for $\vec{e}_n$ as a linear combination of the first three error vectors, we get a linear expression in these values. By finding the extreme values of the feasible positions of the $n+1$'st model point, we may put a convex polygon about these positions which will be more accurate than a square.

We should note that linear programming is very efficient. It is known to be polynomial time in the worst case. In practice, for problems with l variables and m constraints, the most common algorithm, simplex, is found to usually take time proportional to lm^2 (Strang [41]), and many highly optimized commercial implementations of simplex exist. Our problem has 6 variables and $4n$ constraints when n points are matched. When the number of variables in a problem is fixed and only the number of constraints grows, as in our case, there are algorithms that take linear expected time (see Seidel [40], for example).

When the errors in the image points are Gaussian distributed and there are more than three matched points, the Kalman filter can be used to recursively compute Gaussian distributions for the error vectors $\vec{e}_0$, $\vec{e}_1$, and $\vec{e}_2$, similar to Hel-Or and Werman [24]. Given Gaussian distributions for $\vec{e}_0$, $\vec{e}_1$, and $\vec{e}_2$, Appendix C shows how to obtain a Gaussian distribution for the possible locations of any $n+1$'st point.

These methods could be quite useful to the indexing or alignment approaches to recognition that we previously described. To illustrate, a recognition system that uses the linear programming method might work as follows:

1. Match k image and model points, using a search or indexing method. Assume that the error in each image point is bounded by some m-sided polygon.

2. Use the matches to generate km linear constraints on the possible errors in the first three matched points.

3. For each unmatched model point, run l linear programs to compute an l-gon bounding the point's possible image locations. Look there for a matching image point.

Recognition systems commonly use more complex features such as line segments for verification, instead of points. It would be straightforward to use our results to bound the uncertainty regions of line segments by finding the uncertainty regions of their endpoints (as in [4]). Additionally, our results allow us to measure experimentally the extent to which additional point matches decrease our uncertainty about the location of unmatched points. We discuss this in the next section.

7 Experiments

To demonstrate the value of narrowing the feasible region in which model points may appear by using additional matches, we have implemented a test system for the case of weak-perspective projection. In this system, we match some image and model points. We then examine the regions in which additional points might appear. We can see how much smaller these feasible regions become as we derive additional constraints from more matches.

We first describe the results of this system on synthetic data. This allows us to systematically explore two key issues. First, since our linear transformation is only an approximation, how often might it cause us to make errors? Second, how much can the addition of further matches reduce the space in which we must search for even more matches?

We use the following experimental conditions. First, we generate random sets of seven model points inside a cube. We form an image by projecting these points orthographically, scaling so that the cube projects to a 1000×1000 square image. We then add error so that each sensed point shows up inside a circle of radius five pixels centered at the projected position of the point. For error, a uniform random distribution is used.

We then match the first three noisy image points and model points, and use this match to generate a noisy pose of the model. This pose is then used to compute the linear transformations describing the location of each additional model point as a function of the error in the first three image points. Of the two possible model poses that can be derived using a match of three points, we automatically select the one that is closer to the correct pose. In a real system, of course, we would have to explore both possibilities, and see which led to more confirming evidence.

Given these linear transformations, we allow for a square error bound of width 10 pixels around each error circle. As described above, we then compute a rectangular bound on the image location of each additional model point using linear programming. Next we check to see which image points actually appear within the predicted rectangular boundary. Since we have perturbed the image points within their error circles, any time we fail to find a model's image point in the predicted rectangle, this mistake must be due to limitations in our linear approximation. When we do find an image point, we record the size of the rectangle in which we looked.

We then augment our hypothesis by matching the fourth image and model points, and, using the additional constraints, we further narrow down the location of the remaining model points in the image. Again we keep track of how often we fail to find a model point in a predicted rectangle, and we record the areas of the successful rectangles. We continue this process with additional matched points. We repeat this experiment $2,500$ times, continuing to perturb each image point by up to five pixels, but allowing for varying levels of error in our predictions. We can use these results to see how many mistakes of the system could be eliminated by overestimating the expected error, and how much of a price we would pay for this by producing larger rectangles.

Table 4 lists the results. There are several conclusions we may draw. First, we see that few overall mistakes are made. The predictions are generally between 95% and 99% accurate. The significance of this will depend on exactly how we incorporate these error regions into a recognition algorithm. But typically, recognition systems search through many hypothetical matches between image and model points, and it is understood that a system may have to consider more than one correct hypothesis before recognizing an object. This is because even a correct set of matches may lead to an inaccurate pose. We can quantitatively see that our method of computing error regions leads to few such unstable poses.

Second, we can see that additional matches do provide considerable extra constraint in determining the locations of unmatched points. The most dramatic effect occurs when one matches a fourth point. This can reduce the size of possible error regions by a factor of fifteen or more. But even after the fourth point, there is a continuing significant benefit in using additional matches to constrain the error regions. These results also help us to in general assess the stability of poses generated by a small number of feature matches. We can see that if we use three points to compute a pose, small changes in these points can result in large changes in the locations of additional points. Poses computed from more points would be much more stable.

Error Allowed for	Num. Points Matched	Rectangle Average Area	Percentage Correct
5.0	3	16,700	98.9
	4	1,460	97.6
	5	823	96.2
	6	599	94.6
5.25	3	18,300	99.0
	4	1,600	97.8
	5	898	96.7
	6	655	95.5
5.5	3	10,800	98.9
	4	1,750	98.0
	5	969	97.0
	6	730	96.4
6.0	3	12,700	99.2
	4	2,200	98.8
	5	1,370	98.4
	6	948	97.9
6.5	3	12,400	99.2
	4	2,380	98.7
	5	1,410	98.4
	6	1,080	98.0
7.0	3	16,600	99.4
	4	2,950	99.0
	5	1,740	98.9
	6	1,350	98.8

Table 4: This shows the size of error regions computed as more points are matched, and the frequency with which noisy model points fail to appear in these error regions. Image points are always perturbed by a uniform error bounded by five pixels. The first column gives half the width of the square error bound that we allow for in each image point. The second column gives the number of matches used in computing the error region for an additional point. The third column gives the average size of this error region, and the fourth column gives the percentage of times that a model point's image shows up in this predicted error region.

The error regions we compute are in general quite large. Several factors, however, may have exaggerated the sizes of the error regions. First, if we truly wish to consider error as bounded within a disc we should use polygonal error regions that more closely approximate a circle. Rectangles were used in this example only for the sake of simplicity. Second, a uniform error distribution bounded by five pixels may be pessimistic. In real systems, sensed image points probably tend to cluster around the point's true, error-free position, and the error may well be less than five pixels. Therefore a system that allowed for less error may produce much smaller error regions, without making many mistakes. Of course, allowing for less error should only make our linear approximation more accurate.

It is also interesting to note that the accuracy of the error regions drops a bit as we add more point matches. It seems that errors in the linear approximation accumulate as we compute the feasible set of error vectors. One way to compensate for this effect would be to allow for slightly more error as we use more matched points. For example, if we allow for 5 pixels of error when we match three points, we might allow for 6 pixels when matching four points. This would allow us to significantly reduce the size of the error regions, while keeping the error rate essentially constant.

As before, we have run this system on a real model and image to further illustrate its performance. Fig. 12 shows the resulting rectangular error regions for $\epsilon = 5.25$. The figure demonstrates how the uncertainty regions shrink as we match more points, while still containing the true image points.

In summary, we have used linear programming to compute the propagated uncertainty regions in simulation and in a real image for matches with more than three model and image points. The experiments demonstrate that additional matched points can significantly reduce the uncertainty regions with little loss in accuracy.

8 Applications

We have shown how to approximate the effect of changes in model pose using a linear relationship between the error vectors. For predicting the locations of unmatched points, we have demonstrated that this approximation is quite good within the range of error usually considered

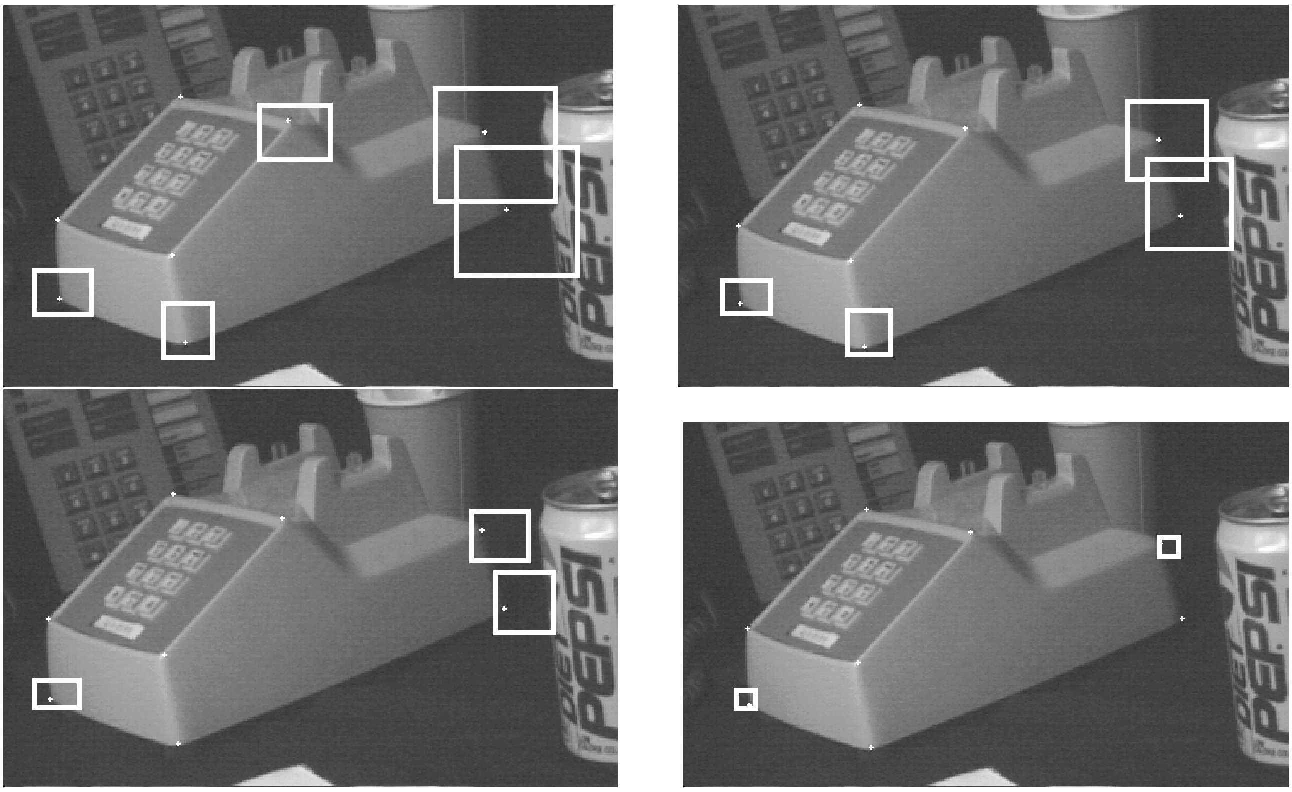

Figure 12: Rectangular uncertainty regions from matching more points. The rectangles in the top left image were computed after matching only three points. The following images show the rectangles that resulted from successively matching one more point. When an additional point was matched, we stopped computing rectangles for that point.

by object recognition systems. This suggests that for many recognition applications we may model this relationship linearly.

In past research, the use of linear projection models has led to algorithmic simplicity. Projection of a 3D object may be approximated as a 3D-to-2D affine transformation to gain the advantages of linearity, at the loss of fully capturing the rigidity of objects. Also, scaled-orthographic projection of a planar object is equivalent to a 2D affine transformation of the object, which is linear. Many algorithms have taken advantage of this linearity, either to find matches that are consistent with a bounded error model (e.g., [6, 12, 10, 28]) or to find likely sets of matches assuming Gaussian error (e.g., [36, 39, 47, 7]). We can now extend some of these algorithms to full 3D-from-2D recognition while maintaining object rigidity.

In Section 6, we outlined an algorithm which could be useful to most alignment and indexing approaches to recognition. Some alignment approaches use grouping methods to generate an initial match of more than three points (such as Lowe's [34], Roberts' [37], Jacobs' [29], and Wayner's [45]), and some alignment approaches create an initial alignment using only three points [37, 17, 34, 27, 44]. In the latter case, a recognition system might attempt to add matches, and use these additional matches to narrow the area in which it must search for still more consistent matches. Additionally, the algorithm from Section 6 may be useful in methods that match image to model features by indexing, and then verify these matches [32, 14, 29, 43, 38, 45]. In these approaches, some model features are matched to image features to determine a model pose, and then this pose is used to find matches for additional model features. Our results show exactly where to search for these matches when we have matched three image and model points.

As mentioned above, other approaches to recognition have derived linear constraints on model poses using linear projection models. The linear constraints were used to robustly match models and images in the presence of bounded uncertainty. This line of work originated with Baird [6], who considered models of 2D points undergoing 2D rotation, translation, and scaling. Baird used convex polygons to bound the errors in the image points. He then showed that, when we match an image point to a model point, each side of the polygon places a linear constraint on the set of feasible model poses, if the transformation from matched image points to projected, unmatched model points is linear.

Baird used these constraints as part of an interpretation-tree approach to recognition. His system searched a tree that represented all possible ways of matching image and model points. At each node of the tree, linear programming was used to decide whether the proposed matches were consistent with the polygonal error bounds. In Section 6, we went beyond Baird, not only in handling the scaled-orthographic projection of 3D objects, but also in showing how to use linear programming to find the uncertainty regions of unmatched points.

Breuel [10] used a modification of Baird's approach to produce a tree-search algorithm that in the worst case runs in polynomial time. Cass [12] used linear constraints to show that finding the pose of the model that aligns the most image and model features to within error bounds is inherently a polynomial time problem. Jacobs [28] showed how to perform a Hough transform in error space, instead of model pose space, by discretely computing the feasible region in error space.

Jacobs makes use of a linear relationship between error vectors that exists for the case of affine transformations of 2D objects. Our linearized perspective and weak-perspective models give us a linear relationship for 3D objects as well. As a consequence, Jacobs' method readily can be extended to 3D objects, and without increasing in the dimensionality of the problem.

In addition to Jacobs' method, our linear relationship can be used to extend any of the above methods. To illustrate, we extend Cass' approach to the case of scaled-orthographic projection. Suppose we match three image to three model points and wish to know which pose will match the most additional model and image points. We know that these three matches give us simple linear constraints on the first three error vectors, just from the image points' error bounds. Now match each additional model point to each additional image point. These give us more linear constraints. Each linear constraint describes a 5D hyperplane in a 6D space of the possible error values: $e_{\{0,x\}}, e_{\{0,y\}}, e_{\{1,x\}}, e_{\{1,y\}}, e_{\{2,x\}}, e_{\{2,y\}}$. If we take each set of six linear constraints, the constraints in general will intersect at a point. This point corresponds to a set of error vectors for the first three points, and hence to a possible pose of the object. As Cass showed, if we now consider all of these poses, we will be guaranteed to find one that matches the most model points to image points. In fact we will find all poses that match the model to different collections of image points.

Cass has developed efficient heuristic algorithms for exploring the space of poses in the case of a rigid 2D rotation and translation. For 3D recognition, the algorithm becomes costly, however. In our case, if we have m model points and n image points, for each of the $O(m^3n^3)$ initial matches we must consider $O(m^6n^6)$ poses. Hopefully, however, a polynomial time formulation of matching for scaled-orthographic projection may lead to more efficient heuristic solutions, such as the ones that Cass has found in other domains. Alternately, Jacobs' approach of performing a Hough transform in error space may be more effective since error space is more compact.

9 Conclusion

This work will allow recognition systems to accurately take account of the effects of sensing error, during a process that finds supporting evidence to confirm a hypothetical set of matches.

Uncertainty Region Solution	3 matched points	> 3 matched points
Scaled-orthographic, Gaussian	Circularly-symmetric Gaussian	Gaussian
Scaled-orthographic, Bounded	Circle	Linear Programming
Perspective, Gaussian	Gaussian	Gaussian
Perspective, Bounded	Linear Programming	Linear Programming

Table 5: Propagated uncertainty regions for circularly-symmetric Gaussian and bounded errors in the image points. Our solution for the uncertainty region either is analytic, in which case a description of the analytic solution is given, or is numerical, in which case the solution can be found by Linear Programming.

We showed that a linear approximation to scaled-orthographic projection is accurate when reasonable amounts of sensing error have occurred. In addition, we showed how to compute the propagated uncertainty regions for the rigid projection of a 3D model into a 2D image. The uncertainty regions for three matched points are described by a simple analytic expression, for the projection and error model cases of (scaled-orthographic, Gaussian), (scaled-orthographic, bounded), and (perspective, Gaussian). When more points are matched, there are simple and efficient algorithms for computing the uncertainty regions. Table 5 summarizes the results.

Both analytic results and experiments have demonstrated the value of accurately computing the uncertainty regions. The uncertainty region of a point can vary greatly depending on both the model geometry and pose. Therefore, any naive approach that uses an hypothesized pose to match additional model and image points is likely either to match many of the model points to image points they could not have produced, or to miss many image points they could have produced.

We found that uncertainty regions based on randomly matching only three points tend to be quite large, supporting past work [21, 4] that they will lead to many false matches. We also observed, however, that uncertainty regions shrink dramatically when we match even one more point, and still further when we match more. This demonstrates that matching larger sets of points, while being careful about error, can produce much more accurate recognition systems.

Finally, we extended several existing approaches to handling error in recognition systems, which were previously restricted to domains with linear projection models. For future work, we are looking to implement the robust recognition systems outlined in Sections 2 and 6.

Availability

To facilitate the use of the results in this paper, we have made available our C code for computing the 3D pose solution implied by 3 point correspondences under weak perspective, the three scaled rotation matrices (Equation 25), and the uncertainty circles (Equation 26). To retrieve the code, ftp to "ftp.ai.mit.edu," then log in as "anonymous," then cd to "pub/users/tda/," and then get and uncompress "alignment-code.tar.Z."

A Scaled-Orthographic Similarity Transform

This appendix solves the following equations for $\mathbf{A}$.

$$\overrightarrow{t'_\theta} = \mathbf{A}\,\overrightarrow{t_\theta} \quad \text{and} \quad \overrightarrow{t'_\phi} = \mathbf{A}\,\overrightarrow{t_\phi},$$

Let $\overrightarrow{t_\theta} = (x_\theta, y_\theta)$, $\overrightarrow{t_\phi} = (x_\phi, y_\phi)$, $\overrightarrow{t'_\theta} = (x'_\theta, y'_\theta)$, $\overrightarrow{t'_\phi} = (x'_\phi, y'_\phi)$. Expanding the first row of each equation gives $x'_\theta = a_{11}x_\theta + a_{12}x_\theta$ and $x'_\phi = a_{11}x_\phi + a_{12}y_\phi$, which implies $[a_{11} \quad a_{12}]^T = \mathbf{T}^{-1}[x'_\theta \quad x'_\phi]^T$, where $\mathbf{T} = \begin{bmatrix} x_\theta & y_\theta \\ x_\phi & y_\phi \end{bmatrix}$. Then from Equations 18 and 19,

$$\begin{aligned}
\mathbf{T}^{-1} &= \frac{s_0 r}{\cos^2\theta} \begin{bmatrix} -\sin\phi & \sin\theta\cos\phi \\ -\cos\theta\sin\theta\cos\phi & -\cos\theta\sin\phi \end{bmatrix} \\
&= \left(\frac{\cos^2\theta}{s_0 r}\right) \frac{1}{\cos\theta(\sin^2\phi + \sin^2\theta\cos^2\phi)} \begin{bmatrix} -\cos\theta\sin\phi & -\sin\theta\cos\phi \\ \cos\theta\sin\theta\cos\phi & -\sin\phi \end{bmatrix} \\
&= \left(\frac{\cos^2\theta}{s_0 r}\right) \frac{\cos\theta}{s_0 r(1-\cos^2\theta\cos^2\phi)} \begin{bmatrix} -\cos\theta\sin\phi & -\sin\theta\cos\phi \\ \cos\theta\sin\theta\cos\phi & -\sin\phi \end{bmatrix}.
\end{aligned}$$

Lastly we multiply $\mathbf{T}^{-1}$ by

$$\begin{bmatrix} x'_\theta \\ x'_\phi \end{bmatrix} = \frac{s_0 r'}{\cos^2\theta} \begin{bmatrix} -\sin(\phi+\tau') \\ -\cos\theta\sin\theta\cos(\phi+\tau') \end{bmatrix},$$

to get

$$\begin{aligned}
\begin{bmatrix} a_{11} \\ a_{12} \end{bmatrix} &= \left(\frac{s_0 r'}{\cos^2\theta}\right)\left(\frac{\cos\theta}{s_0 r(1-\cos^2\theta\cos^2\phi)}\right) \\
&\qquad \begin{bmatrix} -\cos\theta\sin\phi & -\sin\theta\cos\phi \\ \cos\theta\sin\theta\cos\phi & -\sin\phi \end{bmatrix} \begin{bmatrix} -\sin(\phi+\tau') \\ -\cos\theta\sin\theta\cos(\phi+\tau') \end{bmatrix}
\end{aligned}$$

$$= \frac{r'}{r}\left(\frac{1}{1-\cos^2\theta\cos^2\phi}\right)\begin{bmatrix} -\sin\phi\sin(\phi+\tau')+\sin^2\theta\cos\phi\cos(\phi+\tau') \\ -\sin\theta(\cos\phi\sin(\phi+\tau')+\sin\phi\cos(\phi+\tau')) \end{bmatrix}$$

$$= \frac{r'}{r}\left(\frac{1}{1-\cos^2\theta\cos^2\phi}\right)\begin{bmatrix} \cos\tau'-\cos^2\theta\cos\phi\cos(\phi+\tau') \\ -\sin\tau'\sin\theta \end{bmatrix} \qquad (41)$$

This equation gives $\mathbf{A}$, since $a_{21} = -a_{12}$ and $a_{22} = a_{11}$.

B Derivation of the Perspective Linear Transform

Using perspective projection, this appendix derives a linear transform that relates the errors in third and fourth points. First we compute the 3D position of $\vec{m}_2$ in camera coordinates, as given in Equation 32. In Fig. 11, let the normal to the plane through $(\vec{c}, \vec{i}_0, \vec{i}_1)$ be

$$\widehat{n} \;=\; \frac{\vec{c}\times\vec{i}_1}{\|\,\vec{c}\times\vec{i}_1\,\|} \;=\; (0, n_y, n_z) \qquad (42)$$

By Rodriguez' formula,

$$\begin{aligned} \mathbf{R}_{\{\theta,\widehat{n}\}}\vec{p} \;&=\; (\cos\theta)\vec{p} + (1-\cos\theta)(\widehat{n}\cdot\vec{p})\widehat{n} + \sin\theta(\widehat{n}\times\vec{p}) \\ &=\; (d\cos\theta + r\sin\theta(n_y\sin\phi - n_z\cos\phi), \\ &\qquad r\cos\phi\cos\theta + r(1-\cos\theta)(n_y\cos\phi + n_z\sin\phi)n_y + n_z d\sin\theta, \\ &\qquad r\sin\phi\cos\theta + r(1-\cos\theta)(n_y\cos\phi + n_z\sin\phi)n_z - n_y d\sin\theta) \end{aligned} \qquad (43)$$

To compute the translation $\vec{u}$, let a and b be the (known) distances from $\vec{c}$ to $\vec{i}_0$ and from $\vec{c}$ to $\vec{i}_1$, respectively. From the Law of Cosines,

$$\theta_{01} \;=\; \cos^{-1}\left(\frac{d_{01}^2 - a^2 - b^2}{2ab}\right), \qquad \psi \;=\; \cos^{-1}\left(\frac{b^2 - a^2 - d_{01}^2}{2ad_{01}}\right), \qquad (44)$$

where $\theta_{01}, \psi \in (0, \pi)$. From the Law of Sines,

$$L \;=\; \sin(\pi - (\theta+\psi) - \theta_{01})\left(\frac{R_{01}}{\sin\theta_{01}}\right) \;=\; \sin(\theta+\psi+\theta_{01})\left(\frac{R_{01}}{\sin\theta_{01}}\right)$$

In total, we have

$$(\overline{x}, \overline{y}, \overline{z}) \;=\; \vec{m}_2 \;=\; \vec{c} + L\widehat{v} + \mathbf{R}_{\{\theta,\widehat{n}\}}(d, r\cos\phi, r\sin\phi). \qquad (45)$$

Substituting $\overline{x}$, $\overline{y}$, and $\overline{z}$ into Equation 34 gives $\vec{m}_2^{\pi}$.

Next we take a first approximation to x and y in Equation 34 with respect to θ and ϕ. Let $x_\theta = \frac{\partial x}{\partial \theta}$, $y_\theta = \frac{\partial y}{\partial \theta}$, and similarly for x_ϕ, y_ϕ, $\overline{x}_\theta$, $\overline{y}_\theta$, $\overline{y}_\phi$, $\overline{y}_\phi$, and L_θ. From Equation 34,

$$x_\theta = f\left(\frac{\overline{x}_\theta}{\overline{z}+f} - \frac{\overline{z}_\theta(\overline{x}-c_x)}{(\overline{z}+f)^2}\right), \qquad y_\theta = f\left(\frac{\overline{y}_\theta}{\overline{z}+f} - \frac{\overline{z}_\theta(\overline{y}-c_y)}{(\overline{z}+f)^2}\right). \tag{46}$$

For x_ϕ and y_ϕ, we substitute ϕ for θ in these equations. Using Equations 43-45,

$$\overline{x}_\theta = -d\sin\theta + r\cos\theta(n_y\sin\phi - n_z\cos\phi) + L_\theta v_x \tag{47}$$

$$\overline{y}_\theta = -r\cos\phi\sin\theta + r\sin\theta(n_y\cos\phi + n_z\sin\phi)n_y + n_z d\cos\theta + L_\theta v_y \tag{48}$$

$$\overline{z}_\theta = -r\sin\phi\sin\theta + r\sin\theta(n_y\cos\phi + n_z\sin\phi)n_z - n_y d\cos\theta + L_\theta v_z \tag{49}$$

$$L_\theta = \cos(\theta+\psi+\theta_{01})\left(\frac{R_{01}}{\sin\theta_{01}}\right) \tag{50}$$

$$\overline{x}_\phi = r\sin\theta(n_y\cos\phi + n_z\sin\phi) \tag{51}$$

$$\overline{y}_\phi = -r\sin\phi\cos\theta + r(1-\cos\theta)(-n_y\sin\phi + n_z\cos\phi)n_y \tag{52}$$

$$\overline{z}_\phi = r\cos\phi\cos\theta + r(1-\cos\theta)(-n_y\sin\phi + n_z\cos\phi)n_z \tag{53}$$

The above equations give $\overrightarrow{t_\theta}$ and $\overrightarrow{t_\phi}$. By substituting r' for r, d' for d, and $\phi+\tau'$ for ϕ, we get $\overrightarrow{t'_\theta}$ and $\overrightarrow{t'_\phi}$. Solving Equation 21 leads to

$$\mathbf{A} = \frac{1}{x_\theta y_\phi - x_\phi y_\theta}\begin{bmatrix} y_\phi x'_\theta - y_\theta x'_\phi & -x_\phi x'_\theta + x_\theta x'_\phi \\ y_\phi y'_\theta - y_\theta y'_\phi & -x_\phi y'_\theta + x_\theta y'_\phi \end{bmatrix} \tag{54}$$

C Gaussian Error Propagation

For this appendix, we adopt Therrian's notation [42]. In general, let $\vec{x}$ be a Gaussian random vector and let $\vec{y} = \mathbf{M}\vec{x} + \vec{b}$, where $\mathbf{M}$ is $n \times m$. For random vectors $\vec{x}$ and $\vec{y}$, respectively, denote their expected values by $\vec{m}_x$ and $\vec{m}_y$ and their covariance matrices by $\mathbf{K_x}$ and $\mathbf{K_y}$. Then

$$p_{\vec{y}}\left(\widehat{\vec{y}}\right) = \frac{1}{(2\pi)^{\frac{n}{2}}\left|\mathbf{K_y}\right|^{\frac{1}{2}}} e^{\left(-\frac{1}{2}(\widehat{\vec{y}}-\vec{m}_y)^{\mathrm{T}}\mathbf{K_y}^{-1}(\widehat{\vec{y}}-\vec{m}_y)\right)},$$

where $\vec{m}_y = \mathbf{M}\vec{m}_x + \vec{b}$ and $\mathbf{K_y} = \mathbf{M}\mathbf{K_x}\mathbf{M}^{\mathrm{T}}$ [42]. In our case, we have four two-dimensional Gaussian distributions, corresponding to the errors in three matched image points and one unmatched image point. This gives eight uncorrelated Gaussian random variables, of which we are taking a linear combination. When $\vec{\imath}_0$, $\vec{\imath}_1$, $\vec{\imath}_2$, and $\vec{\imath}_3$ are normally distributed with standard deviations σ_0, σ_1, σ_2, and σ_3, respectively, $\mathbf{K_x}$ is a diagonol matrix with on-diagonol elements $(\sigma_0^2, \sigma_0^2, \sigma_1^2, \sigma_1^2, \sigma_2^2, \sigma_2^2, \sigma_3^2, \sigma_3^2)$. Further, the linear combination in Equation 25 is given by

$$\mathbf{M} = \begin{bmatrix} a_{11} & a_{12} & b_{11} & b_{12} & c_{11} & c_{12} & 1 & 0 \\ a_{21} & a_{22} & b_{21} & b_{22} & c_{21} & c_{22} & 0 & 1 \end{bmatrix}, \tag{55}$$

where $n = 2$ and $m = 8$. Expanding $\mathbf{M K_x M}^{\mathrm{T}}$ leads to

$$\begin{aligned} \mathbf{K_y} \quad = \quad & \sigma_0^2 \begin{bmatrix} a_{11}^2 + a_{12}^2 & a_{11}a_{21} + a_{12}a_{22} \\ a_{11}a_{21} + a_{12}a_{22} & a_{21}^2 + a_{22}^2 \end{bmatrix} + \sigma_1^2 \begin{bmatrix} b_{11}^2 + b_{12}^2 & b_{11}b_{21} + b_{12}b_{22} \\ b_{11}b_{21} + b_{12}b_{22} & b_{21}^2 + b_{22}^2 \end{bmatrix} \\ & + \sigma_2^2 \begin{bmatrix} c_{11}^2 + c_{12}^2 & c_{11}c_{21} + c_{12}c_{22} \\ c_{11}c_{21} + c_{12}c_{22} & c_{21}^2 + c_{22}^2 \end{bmatrix} + \sigma_3^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{56}$$

Under weak-perspective projection, $a_{21} = -a_{12}$, $a_{22} = a_{11}$, $b_{21} = -b_{12}$, $b_{22} = b_{11}$, $c_{21} = -c_{12}$, and $c_{22} = c_{11}$. Letting $S_0 = \sqrt{a_{11}^2 + a_{12}^2}$, $S_1 = \sqrt{b_{11}^2 + b_{12}^2}$, and $S_2 = \sqrt{c_{11}^2 + c_{12}^2}$, the expression for $\mathbf{K_y}$ simplifies to

$$\mathbf{K_y} = \left(S_0^2\sigma_0^2 + S_1^2\sigma_1^2 + S_2^2\sigma_2^2 + \sigma_3^2 \right) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{57}$$

Then

$$p_{\vec{y}}\left(\widehat{\vec{y}}\right) = \frac{1}{2\pi\sigma} e^{-\frac{\|\widehat{\vec{y}} - \vec{m}_y\|^2}{2\sigma}},$$

where

$$\sigma = |\mathbf{K_y}|^{\frac{1}{2}} = \sqrt{S_0^2\sigma_0^2 + S_1^2\sigma_1^2 + S_2^2\sigma_2^2 + \sigma_3^2} \tag{58}$$

References

[1] Ahlfors, L. V., *Complex Analysis*, 2nd ed., New York: McGraw-Hill, Inc., 1966.

[2] Alter, T. D., "3-D Pose from 3 Points Using Weak-Perspective," *IEEE Transactions on Pattern Anal. and Machine Intell.*, **16**(8):802-808, 1994.

[3] Alter, T. D., "Robust and Efficient 3D Recognition by Alignment," MIT Artif. Intell. Lab. Tech. Report 1410, 1992.

[4] Alter, T. D., and W. E. L. Grimson, "Fast and Robust 3D Recognition by Alignment," *Proc. Fourth Inter. Conf. on Computer Vision*, May 1993.

[5] Ayache, N., and O. Faugeras, "HYPER: A New Approach for the Recognition and Positioning of Two-Dimensional Objects," *IEEE Transactions on Pattern Anal. and Machine Intell.*, **8**(1):44-54, 1986.

[6] Baird, H., *Model-Based Image Matching Using Location*, MIT Press, Cambridge, 1985.

[7] Beveridge, R., R. Weiss, and E. Riseman, "Combinatorial Optimization Applied to Variable Scale 2D Model Matching," *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 18-23, 1990.

[8] Bolles, R. C., and R. A. Cain, "Recognizing and Locating Partially Visible Objects: The Local-Feature-Focus Method," *Inter. Journal of Robotics Research*, **1**(3):57-82, 1982.

[9] Bolles, R. C., L. H. Quam, M. A. Fischler, and H. C. Wolf, "The SRI Road Expert: Image-to-Database Correspondence," *Proc. DARPA IU Workshop*, pp. 163-174, 1978.

[10] Breuel, T., "Model Based Recognition using Pruned Correspondence Search," *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 257-268, 1991.

[11] Brooks, R. A., "Symbolic Reasoning Among 3-D Models and 2-D Images," *Artificial Intell.*, **17**:285-348, 1981.

[12] Cass, T., "Polynomial Time Object Recognition in the Presence of Clutter, Occlusion and Uncertainty," *Second European Conf. on Computer Vision*, pp. 834-842, 1992.

[13] Clark, C. S., W. O. Eckhardt, C. A. McNary, R. Nevatia, K. E. Olin, and E. M. VanOrden, "High-accuracy Model Matching for Scenes Containing Man-Made Structures," *Proc. Symp. Digital Processing of Aerial Images*, SPIE, vol. 186, pp. 54-62, 1979.

[14] Clemens, D., and D. W. Jacobs, "Space and Time Bounds on Model Indexing," *IEEE Transactions on Pattern Anal. and Machine Intell.*, **13**(10):1007-1018, 1991.

[15] Costa, M., R. M. Haralick, and L. G. Shapiro, "Optimal Affine-Invariant Point Matching," *Proc. Sixth Israeli Conf. on Artif. Intell.*, pp. 35-61, 1990.

[16] Ellis, R. E., "Uncertainty Estimates for Polyhedral Object Recognition," *Proc. IEEE Conf. Rob. Aut.*, pp. 348-353, 1989.

[17] Fischler, M. A., and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Analysis and Automated Cartography," *Communications of the Association of Computing Machinery*, **24**(6):381-395, 1981.

[18] Goad, C. A., "Special Purpose Automatic Programming for 3D Model-Based Vision," *Proc. DARPA Image Understanding Workshop*, pp. 94-104, 1983.

[19] Grimson, W. E. L., and D. P. Huttenlocher, "On the Sensitivity of the Hough Transform for Object Recognition," *IEEE Transactions on Pattern Anal. and Machine Intell.*, **12**(3), March 1990.

[20] Grimson, W. E. L., and D. P. Huttenlocher, "On the Sensitivity of Geometric Hashing," *Proc. Third Inter. Conf. Computer Vision*, pp. 334-338, 1990.

[21] Grimson, W. E. L., D. P. Huttenlocher, and T. D. Alter, "Recognizing 3D Objects from 2D Images: An Error Analysis," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1992.

[22] Grimson, W. E. L., D. P. Huttenlocher, and D. W. Jacobs, "A Study of Affine Matching with Bounded Sensor Error," *Proc. Second European Conf. on Computer Vision*, pp. 291-306, 1992.

[23] Grimson, W. E. L., and T. Lozano-Pérez, "Localizing Overlapping Parts by Searching the Interpretation Tree," *IEEE Transactions on Pattern Anal. and Machine Intell.*, **9**(4):469-482, 1987.

[24] Hel-Or, Y., and M. Werman, "Absolute Orientation from Uncertain Point Data: A Unified Approach," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 77-82, 1991.

[25] Horaud, R., "New Methods for Matching 3-D Objects with Single Perspective Views," *IEEE Transactions on Pattern Anal. and Machine Intell.*, **9**(3):401-412, May 1987.

[26] Huttenlocher, D., "Three-Dimensional Recognition of Solid Objects from a Two-Dimensional Image," MIT Artif. Intell. Lab. Tech. Report 1045, 1988.

[27] Huttenlocher, D., and S. Ullman, "Recognizing Solid Objects by Alignment with an Image," *Inter. Journal of Computer Vision*, **5**(2):195-212, 1990.

[28] Jacobs, D. W., "Optimal Matching of Planar Models in 3D Scenes," *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 269-274, 1991.

[29] Jacobs, D. W., "Recognizing 3-D Objects Using 2-D Images," MIT Artif. Intell. Lab. Tech. Report 1416, 1993.

[30] Kumar, R., and A. Hanson, "Robust Estimation of Camera Location and Orientation from Noisy Data having Outliers," *Proc. IEEE Workshop on Interpretation of 3D Scenes*, pp. 52-60, 1989.

[31] Lamdan, Y., J. T. Schwartz, and H. J. Wolfson, "Object Recognition by Affine Invariant Matching" *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 335-344, 1988.

[32] Lamdan, Y., J. T. Schwartz, and H. J. Wolfson, "Affine Invariant Model-Based Object Recognition," *IEEE Transactions Robotics and Automation*, **6**:578-589, 1990.

[33] Lamdan, Y., and H. J. Wolfson, "On the Error Analysis of 'Geometric Hashing'," *IEEE Conf. Computer Vision and Pattern Recognition*, pp. 22-27, 1991.

[34] Lowe, D., *Perceptual Organization and Visual Recognition*, Kluwer Academic Publishers, The Netherlands, 1985.

[35] Rigoutsos, I., "Massively Parallel Bayesian Object Recognition," Ph. D. Thesis, New York University Department of Computer Science, 1992.

[36] Rigoutsos, I., and R. Hummel, "Robust Similarity Invariant Matching in the Presence of Noise," *Eighth Israeli Conf. on Artif. Intell. Computer Vision*, Tel Aviv, 1991.

[37] Roberts, L., "Machine Perception of Three-Dimensional Solid Objects," *Optical and Electro-optical Information Processing*, edited by J. Tippett, MIT Press, Cambridge, 1966.

[38] Rothwell C., A. Zisserman, J. Mundy, and D. Forsyth, "Efficient Model Library Access by Projectively Invariant Indexing Functions," *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 109-114, 1992.

[39] Sarachik, K. B., and W. E. L. Grimson, "Gaussian Error Models for Object Recognition," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 400-406, June 1993.

[40] Seidel, R., "Linear Programming and Convex Hulls Made Easy," *Proc. of the Sixth Annual Symp. on Computational Geometry*, pp. 211-215, 1990.

[41] Strang, G., *Linear Algebra and its Applications*, Harcourt, Brace, Jovanavich, Publishers, 1988.

[42] Therrien, C. H., *Decision, Estimation, and Classification*, Wiley & Sons, 1989.

[43] Thompson, D., and J. L. Mundy, "Three-Dimensional Model Matching From an Unconstrained Viewpoint", *Proc. IEEE Conf. Rob. Aut.*, pp. 208-220, 1987.

[44] Ullman, S., and R. Basri, "Recognition by Linear Combinations of Models," *IEEE Transactions on Pattern Anal. and Machine Intell.*, **13**(10):992-1007, 1991.

[45] Wayner, P. C., "Efficiently Using Invariant Theory for Model-based Matching," *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 473-478, 1991.

[46] Weinshall, D., and R. Basri, "Distance Metric Between 3-D Models and 2-D Images for Recognition and Classification," *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 220-225, 1993.

[47] Wells, W., "MAP Model Matching," *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 486-492, 1991.