CAR-TR-974 EAR-99-05844
CS-TR-4340 IIS-00-86116
UMIACS-TR-2002-22 July 2002

Evaluation of the SoftPOSIT Model-to-Image Registration Algorithm

Philip David[1,2], Daniel DeMenthon[1], Ramani Duraiswami[1], and Hanan Samet[1]

[1]University of Maryland
Institute for Advanced Computer Studies
College Park, MD 20742
[2]Army Research Laboratory
2800 Powder Mill Road
Adelphi, MD 20783-1197

Abstract

The problem of pose estimation arises in many areas of computer vision, including object recognition, object tracking, site inspection and updating, and autonomous navigation when scene models are available. We present a new algorithm, called *SoftPOSIT*, for determining the pose of a 3D object from a single 2D image when correspondences between model points and image points are not known. The algorithm combines Gold's iterative *softassign* algorithm [20, 21] for computing correspondences and DeMenthon's iterative POSIT algorithm [14] for computing object pose under a full-perspective camera model. Our algorithm, unlike most previous algorithms for pose determination, does not have to hypothesize small sets of matches and then verify the remaining image points. Instead, all possible matches are treated identically throughout the search for an optimal pose. The performance of the algorithm is extensively evaluated in Monte Carlo simulations on synthetic data under a variety of levels of clutter, occlusion, and image noise. These tests show that the algorithm performs well in a variety of difficult scenarios, and empirical evidence suggests that the algorithm has an asymptotic run-time complexity that is better than previous methods by a factor of the number of image points. The algorithm is being applied to a number of practical autonomous vehicle navigation problems including the registration of 3D architectural models of a city to images, and the docking of small robots onto larger robots.

The support of NSF grants EAR-99-05844 and IIS-00-86116 is gratefully acknowledged.

1 Introduction

This paper presents an algorithm for solving the *model-to-image registration problem*, which is the task of determining the position and orientation (the *pose*) of a three-dimensional object with respect to a camera coordinate system, given a model of the object consisting of 3D reference points and a single 2D image of these points. We assume that no additional information is available with which to constrain the pose of the object or to constrain the correspondence of model features to image features. This is also known as the *simultaneous pose and correspondence problem.*

Automatic registration of 3D models to images is an important problem. Applications include object recognition, object tracking, site inspection and updating, and autonomous navigation when scene models are available. It is a difficult problem because it comprises two coupled problems, the correspondence problem and the pose problem, each easy to solve only if the other has been solved first:

1. Solving the *pose* (or *exterior orientation*) problem consists of finding the rotation and translation of the object with respect to the camera coordinate system. Given matching model and image features, one can easily determine the pose that best aligns those matches. For three to five matches, the pose can be found in closed form by solving sets of polynomial equations [18, 24, 26, 40]. For six or more matches, linear and nonlinear approximate methods are generally used [14, 17, 25, 27, 31].

2. Solving the *correspondence* problem consists of finding matching image features and model features. If the object pose is known, one can relatively easily determine the matching features. Projecting the model in the known pose into the original image, one can identify matches among the model features that project sufficiently close to an image feature. This approach is typically used for *pose verification*, which attempts to determine how good a hypothesized pose is [23].

The classic approach to solving these coupled problems is the hypothesize-and-test approach [22]. In this approach, a small set of image feature to model feature correspondences are first hypothesized. Based on these correspondences, the pose of the object is computed. Using this pose, the model points are back-projected into the image. If the original and back-projected images are sufficiently similar, then the pose is accepted; otherwise, a new hypothesis is formed and this process is repeated. Perhaps the best known example of this approach is the RANSAC algorithm [18] for the case that no information is available to constrain the correspondences of model points to image points. When three correspondences are used to determine a pose, a high probability of success can be achieved by the RANSAC algorithm in $O(J^4K)$ time when there are J image points and K model points[1] (see Appendix A for details).

The problem addressed here is one that is encountered when taking a model-based approach to the object recognition problem, and as such has received considerable attention. (The other main approach to object recognition is the appearance-based approach [34] in which multiple views of the object are compared to the image. However, since 3D models are not used, this approach doesn't provide accurate object pose.) Many investigators (e.g., [10, 11, 16, 28, 30, 36]) approximate the nonlinear perspective projection via linear affine approximations. This is accurate when the relative depths of object features are small compared to the distance of the object from the camera. Among the pioneer contributions were Baird's tree-pruning method [1], with exponential time complexity for unequal point sets, and Ullman's alignment method [38] with time complexity $O(J^4K^3 \log K)$.

[1] Some authors use N and M instead of J and K, respectively, to denote the numbers of image and model points.

The geometric hashing method [30] determines an object's identity and pose using a hashing metric computed from a set of image features. Because the hashing metric must be invariant to camera viewpoint, and because there are no view-invariant image features for general 3D point sets (for either perspective or affine cameras) [7], this method can only be applied to planar scenes.

In [13], we proposed an approach using binary search by bisection of pose boxes in two 4D spaces, extending the research of [1, 9, 8] on affine transforms, but it had high-order complexity. The approach taken by Jurie [29] was inspired by our work and belongs to the same family of methods. An initial volume of pose space is guessed, and all of the correspondences compatible with this volume are first taken into account. Then the pose volume is recursively reduced until it can be viewed as a single pose. As a Gaussian error model is used, boxes of pose space are pruned not by counting the number of correspondences that are compatible with the box as in [13], but on the basis of the probability of having an object model in the image within the range of poses defined by the box.

Among the researchers who have addressed the full perspective problem, Wunsch and Hirzinger [39] formalize the abstract problem in a way similar to the approach advocated here as the optimization of an objective function combining correspondence and pose constraints. However, the correspondence constraints are not represented analytically. Instead, each model feature is explicitly matched to the closest lines of sight of the image features. The closest 3D points on the lines of sight are found for each model feature, and the pose that brings the model features closest to these 3D points is selected; this allows an easier 3D to 3D pose problem to be solved. The process is repeated until a minimum of the objective function is reached.

The object recognition approach of Beis [2] uses view-variant 2D image features to index 3D object models. Off-line training is performed to learn 2D feature groupings associated with large numbers of views of the objects. Then, the on-line recognition stage uses new feature groupings to index into a database of learned model-to-image correspondence hypotheses, and these hypotheses are used for pose estimation and verification.

The pose clustering approach to model-to-image registration is similar to the classic hypothesize-and-test approach. Instead of testing each hypothesis as it is generated, all hypotheses are generated and clustered in a pose space before any back-projection and testing takes place. This later step is performed only on poses associated with high-probability clusters. The idea is that hypotheses including only correct correspondences should form larger clusters in pose space than hypotheses that include incorrect correspondences. Olson [35] gives a randomized algorithm for pose clustering whose time complexity is $O(J^3K)$.

The method of Beveridge and Riseman [3, 4] is also related to our approach. Random-start local search is combined with a hybrid pose estimation algorithm employing both full-perspective and weak-perspective camera models. A steepest descent search in the space of model-to-image line segment correspondences is performed. A weak-perspective pose algorithm is used to rank neighboring points in this search space, and a full-perspective pose algorithm is used to update the model's pose after making a move to a new set of correspondences. The time complexity of this algorithm was empirically determined to be $O(J^2K^2)$.

When there are J image points and K model points, the dimension of the solution space for this problem is $K + 6$ since there are K correspondence variables and 6 pose variables. Each correspondence variable has the domain $\{1, 2, \ldots, J, \emptyset\}$ representing a match of a model point to one of the J image points or to no image point (represented by $\emptyset$), and each pose variable has a continuous domain determined by the allowed range of model translations and rotations. Most algorithms don't explicitly search this $K+6$-dimensional space, but instead assume that pose is determined by correspondences or that cor-

respondences are determined by pose, and so search either an K-dimensional or a 6-dimensional space. The SoftPOSIT approach is different in that its search alternates between these two spaces.

The SoftPOSIT approach to solving the model-to-image registration problem applies the formalism proposed by Gold, Rangarajan and others [20, 21] when they solved the correspondence and pose problem in matching two images or two 3D models. We extend it to the more difficult problem of registration between a 3D model and its perspective image, which they did not address. The SoftPOSIT algorithm integrates an iterative pose technique called POSIT (Pose from Orthography and Scaling with ITerations) [14], and an iterative correspondence assignment technique called *softassign* [20, 21] into a single iteration loop. A global objective function is defined that captures the nature of the problem in terms of both pose and correspondence and combines the formalisms of both iterative techniques. The correspondence and the pose are determined *simultaneously* by applying a deterministic annealing schedule and by minimizing this global objective function at each iteration step.

Figure 1 shows an example computation of SoftPOSIT for a model with 15 points. Notice that it would be impossible to make hard correspondence decisions for the initial pose (frame 1), where the model image does not match the actual image at all. The deterministic annealing mechanism keeps all the options open until the two images are almost aligned. As another example of SoftPOSIT, Figure 2 shows the trajectory of the perspective projection of a cube model being aligned to an image of a cube.

In the following sections, we examine each step of the method. We then provide pseudocode for the algorithm. We then evaluate the algorithm using Monte Carlo simulations with various levels of clutter, occlusion and image noise, and finally we apply the algorithm to some real imagery.

2 POSIT Algorithm

One of the building blocks of the new algorithm is the POSIT algorithm, presented in detail in [14]. We summarize this algorithm below in its original form with known correspondences, and then present a variant of the algorithm, still with known correspondences, using the closed-form minimization of an objective function. It is this objective function which is modified in the next section to analytically characterize the global pose-correspondence problem (i.e., without known correspondences) in a single equation.

Consider a pinhole camera of focal length f and an image feature point p with Euclidean coordinates x and y and homogeneous coordinates (wx, wy, w). This point p is the perspective projection of the 3D point P with homogeneous coordinates $(X, Y, Z, 1)$ in the frame of reference of an object with origin P_0.

In our problem, there is an unknown coordinate transformation between the object and the camera, represented by a rotation matrix $R = [\mathbf{R}_1\, \mathbf{R}_2\, \mathbf{R}_3]^T$ and a translation vector $\mathbf{T} = (T_x, T_y, T_z)$. The vectors $\mathbf{R}_1^T$, $\mathbf{R}_2^T$, $\mathbf{R}_3^T$ are the row vectors of the rotation matrix; they are the unit vectors of the camera coordinate system expressed in the model coordinate system. The translation vector $\mathbf{T}$ is the vector from the center of projection O of the camera to the origin P_0 of the object expressed in the camera coordinate system. The coordinates of the perspective projection p can be shown to be related to the coordinates of the world point P by

$$\begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} f\mathbf{R}_1^T & fT_x \\ f\mathbf{R}_2^T & fT_y \\ \mathbf{R}_3^T & T_z \end{bmatrix} \begin{bmatrix} \boldsymbol{P_0P} \\ 1 \end{bmatrix},$$

where $\boldsymbol{P_0P} = (X, Y, Z)^T$ is the vector from P_0 to P. The homogeneous image point coordinates are defined up to a multiplicative constant; therefore the validity of the equality is not affected if we multiply

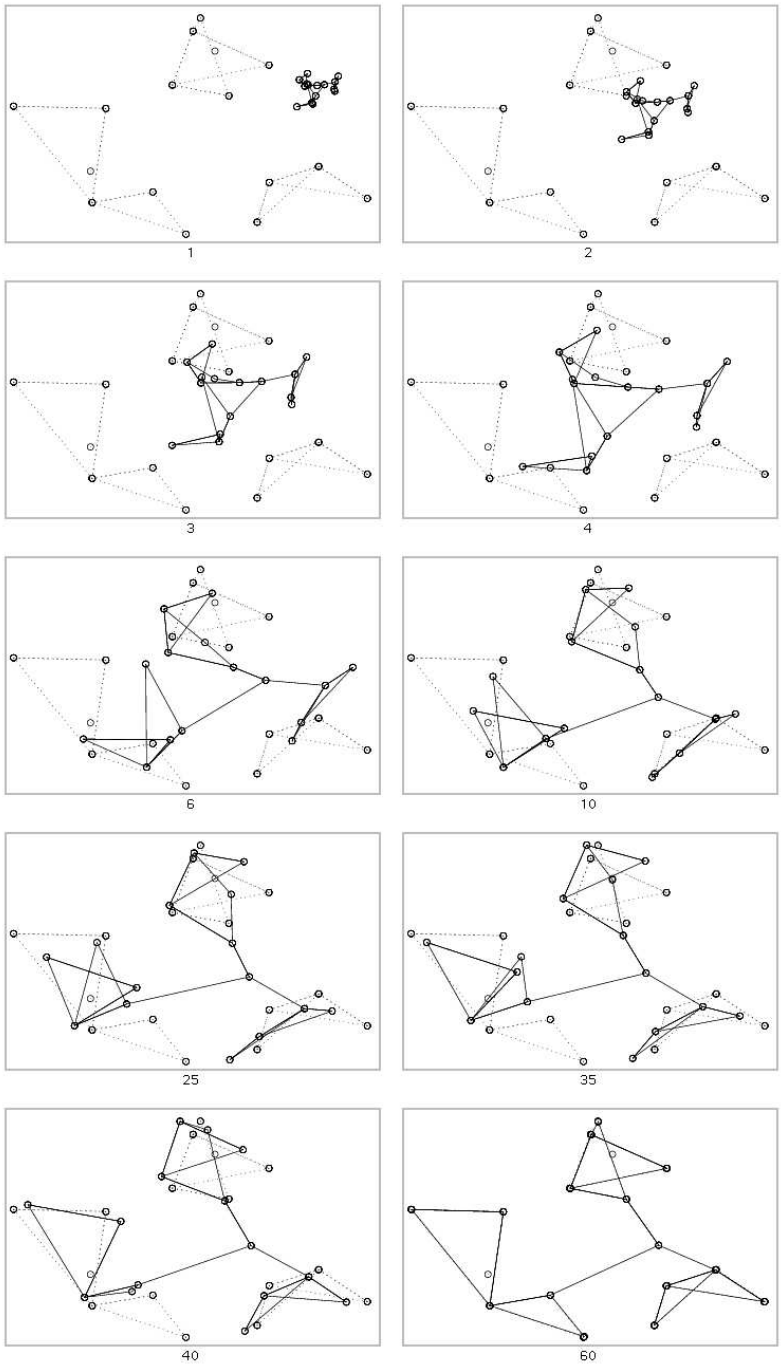


Figure 1: Evolution of perspective projections for a 15-point object (solid lines) being aligned by the SoftPOSIT algorithm to an image (dashed lines) with one occluded point and two clutter points. The iteration step of the algorithm is shown under each frame.

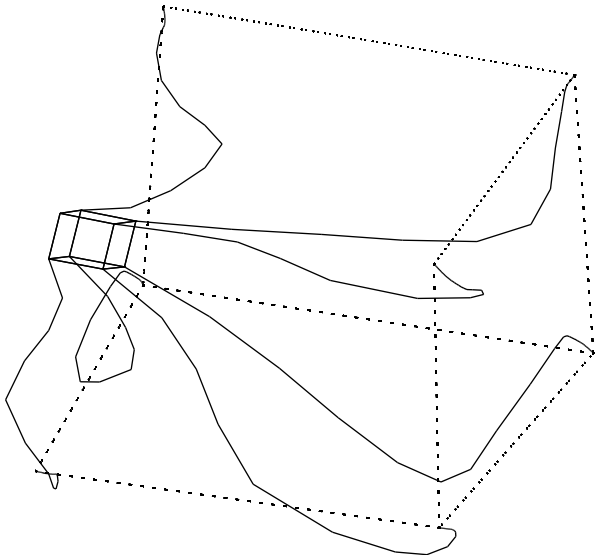


Figure 2: The trajectory of the perspective projection of a cube model (solid lines) being aligned by the SoftPOSIT algorithm to an image of a cube (dashed lines), where one vertex of the cube is occluded. A simple model is used for the sake of clarity.

all the elements of the perspective projection matrix by $1/T_z$. We also introduce the scaling factor $s = f/T_z$ (the reason for this terminology becomes clear below). We obtain

$$\begin{bmatrix} wx \\ wy \end{bmatrix} = \begin{bmatrix} s\mathbf{R}_1^T & sT_x \\ s\mathbf{R}_2^T & sT_y \end{bmatrix} \begin{bmatrix} \boldsymbol{P_0 P} \\ 1 \end{bmatrix}. \tag{1}$$

with

$$w = \mathbf{R}_3 \cdot \boldsymbol{P P_0}/T_z + 1. \tag{2}$$

In the expression for w the dot product $\mathbf{R}_3 \cdot \boldsymbol{P_0 P}$ represents the projection of the vector $\boldsymbol{P_0 P}$ onto the optical axis of the camera. Indeed, in the world coordinate system where P is defined, $\mathbf{R}_3$ is the unit vector of the optical axis. When the depth range of the model along the optical axis of the camera is small with respect to the model distance, $\mathbf{R}_3 \cdot \boldsymbol{P_0 P}$ is small with respect to T_z, and therefore w is close to 1. In this case, perspective projection gives results that are similar to the following transformation:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} s\mathbf{R}_1^T & sT_x \\ s\mathbf{R}_2^T & sT_y \end{bmatrix} \begin{bmatrix} \boldsymbol{P_0 P} \\ 1 \end{bmatrix}. \tag{3}$$

This expression defines the *scaled orthographic* projection p' of the 3D point P. The factor s is the scaling factor of this scaled orthographic projection. When $s = 1$, this equation expresses a transformation of points from a world coordinate system to a camera coordinate system, and uses two of the three world point coordinates in determining the image coordinates; this is the definition of a pure orthographic projection. With a factor s different from 1, this image is scaled and approximates a perspective image because the scaling is inversely proportional to the distance T_z from the camera center of projection to the object origin P_0 ($s = f/T_z$).

The general perspective equation (1) can be rewritten as

$$\begin{bmatrix} X & Y & Z & 1 \end{bmatrix} \begin{bmatrix} s\mathbf{R}_1 & s\mathbf{R}_2 \\ sT_x & sT_y \end{bmatrix} = \begin{bmatrix} wx & wy \end{bmatrix}. \tag{4}$$

Assume that for each image point p with coordinates x and y the corresponding homogeneous coordinate w has been computed at a previous computation step and is known. Then we are able to calculate wx and wy, and the previous equation expresses the relationship between the unknown pose components $s\mathbf{R}_1$, $s\mathbf{R}_2$, sT_x, sT_y, and the known image components wx and wy and known world coordinates X, Y, Z of $\boldsymbol{P_0P}$. If we know K world points P_k, $k = 1, \ldots, K$, their corresponding image points p_k, and their homogeneous components w_k, then we can then write two linear systems of K equations that can be solved for the unknown components of vectors $s\mathbf{R}_1$, $s\mathbf{R}_2$ and the unknowns sT_x and sT_y, provided the rank of the matrix of world point coordinates is at least 4. Thus, at least four of the points of the model for which we use the image points must be noncoplanar. After the unknowns $s\mathbf{R}_1$ and $s\mathbf{R}_2$ are obtained, we can extract s, $\mathbf{R}_1$, and $\mathbf{R}_2$ by imposing the condition that $\mathbf{R}_1$ and $\mathbf{R}_2$ must be unit vectors. Then we can obtain $\mathbf{R}_3$ as the cross-product of $\mathbf{R}_1$ and $\mathbf{R}_2$:

$$s = \left(|s\mathbf{R}_1|\,|s\mathbf{R}_2|\right)^{1/2} \quad \text{(geometric mean)},$$

$$\mathbf{R}_1 = (s\mathbf{R}_1)/s,\ \mathbf{R}_2 = (s\mathbf{R}_2)/s,$$

$$\mathbf{R}_3 = \mathbf{R}_1 \times \mathbf{R}_2,$$

$$T_x = (sT_x)/s,\ T_y = (sT_y)/s,\ T_z = f/s.$$

An additional intermediary step that improves performance and quality of results consists of using unit vectors $\mathbf{R}'_1$ and $\mathbf{R}'_2$ that are mutually perpendicular and closest to $\mathbf{R}_1$ and $\mathbf{R}_2$ in the least square sense. These vectors can be found by singular value decomposition (SVD) (see the Matlab code in [15]).

How can we compute the w_k components required to compute the right-hand side rows $(w_k x_k, w_k y_k)$ corresponding to image point p_k? We saw that setting $w_k = 1$ for every point is a good first step because it amounts to solving the problem with a scaled orthographic model of projection. Once we have the pose result for this first step, we can compute better estimates for the w_k using equation (2). Then we can solve the system of equations (4) again to obtain a refined pose. This process is repeated, and the iteration is stopped when the process becomes stationary.

3 Geometry and Objective Function

We now look at a geometric interpretation of this method in order to propose a variant using an objective function. As shown in Figure 3, consider a pinhole camera with center of projection at O, optical axis aligned with Oz, image plane Π at distance f from O, and image center (principal point) at c. Consider an object, the origin of its coordinate system at P_0, a point P of this object, a corresponding image point p, and the line of sight L of p. The image point p' is the scaled orthographic projection of object point P.

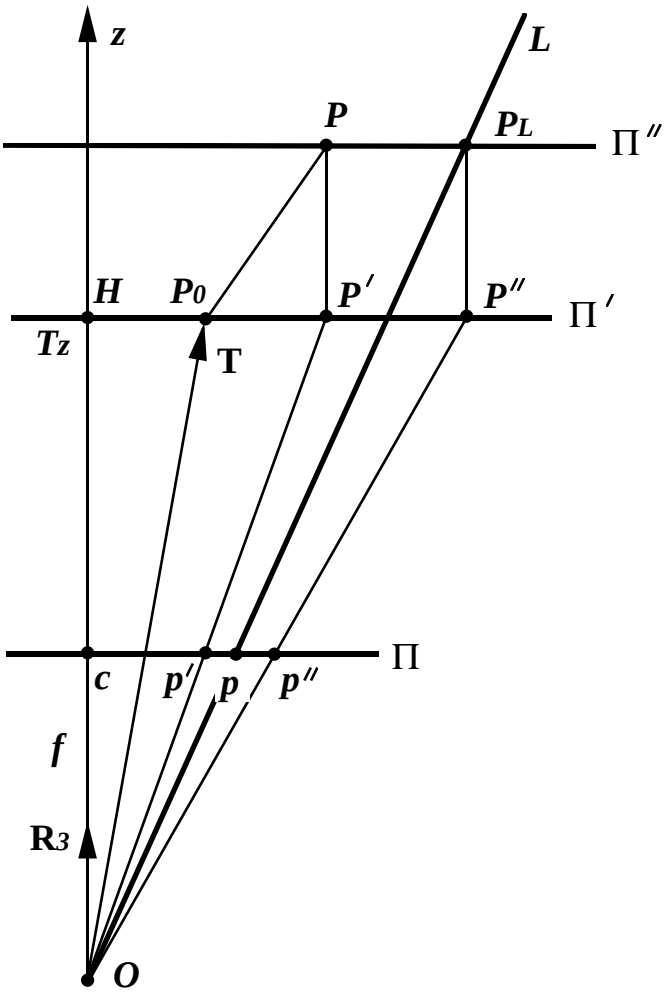


Figure 3: Geometric interpretation of the POSIT computation. Image point p', the scaled orthographic projection of world point P, is computed by one side of the POSIT equations. Image point p'', the scaled orthographic projection of point P_L on the line of sight of p, is computed by the other side of the equation. The equations are satisfied when the two points are superposed, which requires that the world point P be on the line of sight of image point p. The plane of the figure is chosen to contain the optical axis and the line of sight L. The points P_0, P, P', and p' are generally out of this plane.

The image point p'' is the scaled orthographic projection of point P_L obtained by shifting P to the line of sight of p in a direction parallel to the image plane.

One can show (see Appendix B) that the image plane vector from c to p' is

$$\boldsymbol{cp'} = s(\mathbf{R}_1 \cdot \boldsymbol{P_0 P} + T_x, \mathbf{R}_2 \cdot \boldsymbol{P_0 P} + T_y).$$

In other words, the left-hand side of equation (4) represents the vector $\boldsymbol{cp'}$ in the image plane. One can also show that the image plane vector from c to p'' is $\boldsymbol{cp''} = (wx, wy) = w\boldsymbol{cp}$. In other words, the right-hand side of equation (4) represents the vector $\boldsymbol{cp''}$ in the image plane. The image point p'' can be interpreted as a correction of the image point p from a perspective projection to a scaled orthographic projection of a point P_L located on the line of sight at the same distance as P. P is on the line of sight L of p if, and only if, the image points p' and p'' are superposed. Then $\boldsymbol{cp'} = \boldsymbol{cp''}$, i.e. equation (4) is satisfied.

When we try to match the object points P_k to the lines of sight L_k of image points p_k, it is unlikely that all or even any of the points will fall on their corresponding lines of sight, or equivalently that $\boldsymbol{cp'_k} = \boldsymbol{cp''_k}$ or $\boldsymbol{p'_k p''_k} = \mathbf{0}$. The least squares solution of equations (4) for pose enforces these constraints. Alternatively, we can minimize a global objective function E equal to the sum of the squared distances $d_k^2 = |\boldsymbol{p'_k p''_k}|^2$ between image points p'_k and p''_k:

$$\begin{aligned} E &= \textstyle\sum_k d_k^2 = \sum_k \left|\boldsymbol{cp'_k} - \boldsymbol{cp''_k}\right|^2 \\ &= \textstyle\sum_k ((\mathbf{M} \cdot \mathbf{S}_k - w_k x_k)^2 + (\mathbf{N} \cdot \mathbf{S}_k - w_k y_k)^2) \end{aligned} \tag{5}$$

where we have introduced the vectors **M**, **N**, and $\boldsymbol{S}_k$with four homogeneous coordinates to simplify the subsequent notation:

$$\begin{aligned} \mathbf{M} &= (M_1, M_2, M_3, M_4) = s(\mathbf{R}_1, T_x), \\ \mathbf{N} &= (N_1, N_2, N_3, N_4) = s(\mathbf{R}_2, T_y), \\ \mathbf{S}_k &= (\boldsymbol{P_0 P_k}, 1). \end{aligned}$$

We call **M** and **N** the *pose vectors*.

Referring again to Figure 3, notice that $\boldsymbol{p'p''} = s\boldsymbol{P'P''} = s\boldsymbol{PP_L}$. Therefore minimizing this objective function consists of minimizing the scaled sum of squared distances of model points to lines of sight, when distances are taken along directions parallel to the image plane.

This objective function is minimized iteratively. Initially, the w_k are all set to 1. Then the following two operations take place at each iteration step:

1. Compute the pose vectors **M** and **N** assuming the terms w_k are known (equation (5)).
2. Compute the correction terms w_k using the pose vectors **M** and **N** just computed (equation (2)).

We now focus on the optimization of the pose vectors **M** and **N**. The pose vectors that will minimize the objective function E at a given iteration step are those for which all the partial derivatives of the objective function with respect to the coordinates of these vectors are zero. This condition provides 4×4 linear systems for the coordinates of **M** and **N** whose solutions are

$$\mathbf{M} = (\sum_k \mathbf{S}_k \mathbf{S}_k^T)^{-1} (\sum_k w_k x_k \mathbf{S}_k), \tag{6}$$

$$\mathbf{N} = (\sum_k \mathbf{S}_k \mathbf{S}_k^T)^{-1} (\sum_k w_k y_k \mathbf{S}_k). \tag{7}$$

The matrix $L = (\sum_k \mathbf{S}_k \mathbf{S}_k^T)$ is a 4×4 matrix that can be precomputed.

With either method, the point p'' can be viewed as the image point p "corrected" for scaled orthographic projection using w computed at the previous step of the iteration. The next iteration step finds the pose such that the scaled orthographic projection of each point P is as close as possible to its corrected image point.

4 Pose Calculation with Unknown Correspondences

When correspondences are unknown, each image feature point p_j can potentially match any of the model feature points P_k, and therefore must be corrected using the value of w specific to the coordinates of P_k:

$$w_k = \mathbf{R}_3 \cdot \boldsymbol{P_0 P_k} / T_z + 1. \tag{8}$$

Therefore for each image point p_j and each model point P_k we generate a corrected image point p''_{jk}, aligned with the image center c and with p_j, and defined by

$$\boldsymbol{cp''_{jk}} = w_k \boldsymbol{cp_j}. \tag{9}$$

We make use of the squared distances between these corrected image points p''_{jk} and the scaled orthographic projections p'_k of the points P_k whose positions are provided by

$$\boldsymbol{cp'_k} = \begin{bmatrix} \mathbf{M} \cdot \mathbf{S}_k \\ \mathbf{N} \cdot \mathbf{S}_k \end{bmatrix}. \tag{10}$$

These squared distances are

$$d_{jk}^2 = \left| \boldsymbol{p'_k p''_k} \right|^2 = (\mathbf{M} \cdot \mathbf{S}_k - w_k x_j)^2 + (\mathbf{N} \cdot \mathbf{S}_k - w_k y_j)^2, \tag{11}$$

where x_j and y_j are the image coordinates of the image point p_j, $\mathbf{S}_k$ is the vector $(S_{k1}, S_{k2}, S_{k3}, S_{k4}) = (\boldsymbol{P_0 P_k}, 1)$, and $\mathbf{M}$ and $\mathbf{N}$ are pose vectors introduced in the previous section and recomputed at each iteration step. The term w_k is defined by equation (8).

The simultaneous pose and correspondence problem can then be formulated as a minimization of the global objective function

$$\begin{aligned} E &= \sum_{j=1}^{J} \sum_{k=1}^{K} m_{jk}\, d_{jk}^2 \\ &= \sum_{j=1}^{J} \sum_{k=1}^{K} m_{jk} ((\mathbf{M} \cdot \mathbf{S}_k - w_k x_j)^2 + (\mathbf{N} \cdot \mathbf{S}_k - w_k y_j)^2) \end{aligned} \tag{12}$$

where the m_{jk} are weights, equal to 0 or 1, for each of the squared distances d_{jk}^2, and where *J* and *K* are the number of image and model points, respectively. The m_{jk} are correspondence variables that define the assignments between image and model feature points. Note that when all the assignments are well-defined, this objective function becomes equivalent to the objective function defined in equation (5).

This objective function is minimized iteratively, with the following three operations at each iteration step:

1. Compute the correspondence variables assuming everything else is fixed (see below).
2. Compute the pose vectors **M** and **N** assuming everything else is fixed (see below).
3. Compute the correction terms w_k using the pose vectors **M** and **N** just computed (as described in the previous section).

This iterative approach is related to the general expectation-maximization (EM) algorithm [32]. In EM, given a guess for the unknown parameters (the pose in our problem) and a set of observed data (the image points in our problem), the expected value of the unobserved variables (the correspondence matrix in our problem) is estimated. Then, given this estimate for the unobserved variables, the maximum likelihood estimates of the parameters are computed. This process is repeated until these estimates converge.

4.1 Pose Problem

We now focus on finding the optimal poses **M** and **N**, assuming the correspondence variables m_{jk} are known and fixed. As in the previous section, the pose vectors that will minimize the objective function E at a given iteration step are those for which all the partial derivatives of the objective function with respect to the coordinates of these vectors are 0. This condition provides 4×4 linear systems for the coordinates of **M** and **N** whose solutions are

$$\mathbf{M} = (\sum_{k=1}^{K} m'_k \mathbf{S}_k \mathbf{S}_k^{\mathsf{T}})^{-1} (\sum_{j=1}^{J} \sum_{k=1}^{K} m_{jk} w_k x_j \mathbf{S}_k), \tag{13}$$

$$\mathbf{N} = (\sum_{k=1}^{K} m'_k \mathbf{S}_k \mathbf{S}_k^{\mathsf{T}})^{-1} (\sum_{j=1}^{J} \sum_{k=1}^{K} m_{jk} w_k y_j \mathbf{S}_k), \tag{14}$$

with $m'_k = \sum_{j=1}^{J} m_{jk}$. The terms $\mathbf{S}_k \mathbf{S}_k^{\mathsf{T}}$ are 4×4 matrices. Therefore computing **M** and **N** requires the inversion of a single 4×4 matrix, $L = (\sum_{k=1}^{K} m'_k \mathbf{S}_k \mathbf{S}_k^{\mathsf{T}})$, a fairly inexpensive operation (note that because the term in column k and slack row $J+1$ (see below) is generally greater than 0, $m'_k = \sum_{j=1}^{J} m_{jk}$ is generally not equal to 1, and L generally cannot be precomputed).

4.2 Correspondence Problem

We optimize the correspondence variables m_{jk} assuming that the parameters d_{jk}^2 in the expression for the objective function E are known and fixed. Our aim is to find a zero-one *assignment matrix*, $M = \{m_{jk}\}$, that explicitly specifies the matchings between a set of J image points and a set of K model points, and that minimizes the objective function E. M has one row for each of the J image points p_j and one column for each of the K model points P_k. The assignment matrix must satisfy the constraint that each image point match at most one model point, and vice versa (i.e., $\Sigma_i m_{ji} = \Sigma_i m_{ik} = 1$ for all j and k). A *slack row* $J+1$ and a *slack column* $K+1$ are added. A 1 in the slack column $K+1$ at row j indicates that image point p_j has not found any match among the model features. A 1 in the slack row $J+1$ at column k indicates that the feature point P_k is not seen in the image and does not match any image feature. The objective function E will be minimum if the assignment matrix M matches image and model points with

the smallest distances d_{jk}^2. This problem can be solved by the iterative softassign technique [20, 21]. The iteration for the assignment matrix *M* begins with a matrix M_0 in which element m_{jk}^0 is initialized to $\exp(-\beta(d_{jk}^2 - \alpha))$, with β very small, and with all elements in the slack row and slack column set to a small constant. The parameter α determines how far apart two points must be before considering the points unmatchable. See [21] for an analytical justification. The continuous matrix M_0 converges toward the discrete matrix *M* due to two mechanisms that are used concurrently:

1. First, a technique due to Sinkhorn [37] is applied. When each row and column of a square correspondence matrix is normalized (several times, alternatingly) by the sum of the elements of that row or column respectively, the resulting matrix has positive elements with all *rows and columns summing to 1*.

2. The term β is increased as the iteration proceeds. As β increases and each row or column of M_0 is renormalized, the terms m_{jk}^0 corresponding to the smallest d_{jk}^2 tend to converge to 1, while the other terms tend to converge to 0. This is a deterministic annealing process [19] known as *Softmax* [6]. This is a desirable behavior, since it leads to an assignment of correspondences that satisfy the matching constraints and whose sum of distances in minimized.

This combination of deterministic annealing and Sinkhorn's technique in an iteration loop was called softassign by Gold and Rangarajan [20, 21]. The matrix *M* resulting from an iteration loop that comprises these two substeps is the assignment that minimizes the global objective function $E = \sum_{j=1}^{J} \sum_{k=1}^{K} m_{jk}\, d_{jk}^2$. As the following pseudocode shows, these two substeps are interleaved in the iteration loop of SoftPOSIT, along with the substeps that optimize the pose and correct the image points by scaled orthographic distortions.

4.3 Pseudocode for SoftPOSIT

The SoftPOSIT algorithm can be summarized as follows:

Inputs:

1. A list of J image feature points $p_j = (x_j, y_j)$.

2. A list of K world points $\mathbf{S}_k = (X_k, Y_k, Z_k, 1) = (\boldsymbol{P_0 P_k}, 1)$ in the object.

Initialize slack elements of assignment matrix *M* to $\gamma = 1/(\max\{J, K\}+1)$, β to β_0 (β_0 is around 0.0004 if nothing is known about the pose, and is larger if an initial pose can be guessed).
Initialize pose vectors **M** and **N** using the expected pose or a random pose within the expected range.
Initialize $w_k = 1$.
Do A until $\beta > \beta_{final}$ (β_{final} around 0.5) (*Deterministic annealing loop*)

- Compute the squared distances $d_{jk}^2 = (\mathbf{M} \cdot \mathbf{S}_k - w_k x_j)^2 + (\mathbf{N} \cdot \mathbf{S}_k - w_k y_j)^2$
- Compute $m_{jk}^0 = \gamma \exp(-\beta(d_{jk}^2 - \alpha))$
- **Do B until** ΔM **small** (*Sinkhorn's method*)
  - Update matrix *M* by normalizing across all rows: $m_{jk}^1 = m_{jk}^0 / \sum_{k=1}^{K+1} m_{jk}^0$

– Update matrix M by normalizing across all columns: $m_{jk}^0 = m_{jk}^1 / \sum_{j=1}^{J+1} m_{jk}^1$

- **End Do B**
- Compute 4×4 matrix $L = (\sum_{k=1}^{K} m'_k \mathbf{S}_k \mathbf{S}_k^{\mathsf{T}})$ with $m'_k = \sum_{j=1}^{J} m_{jk}$
- Compute L^{-1}
- Compute $\mathbf{M} = L^{-1}(\sum_{j=1}^{J} \sum_{k=1}^{K} m_{jk} w_k \, x_j \mathbf{S}_k)$
- Compute $\mathbf{N} = L^{-1}(\sum_{j=1}^{J} \sum_{k=1}^{K} m_{jk} w_k \, y_j \mathbf{S}_k)$
- Compute $s = \|(M_1,\ M_2,\ M_3)\|$, $\mathbf{R}_1 = (M_1, M_2, M_3)/s$, $\mathbf{R}_2 = (N_1, N_2, N_3)/s$, $\mathbf{R}_3 = \mathbf{R}_1 \times \mathbf{R}_2$
- Compute $w_k = \mathbf{R}_3 \cdot \boldsymbol{P_0 P_k}/T_z + 1$
- $\beta = \beta_{update}\beta$ (β_{update} is around 1.05)

End Do A
Outputs: A rotation matrix $R = [\mathbf{R}_1\ \mathbf{R}_2\ \mathbf{R}_3]^{\mathsf{T}}$, a translation vector $\mathbf{T} = (T_x, T_y, T_z)$, and an assignment matrix $M = \{m_{jk}\}$ between the list of image points and the list of world points of the input.

5 Random Start SoftPOSIT

The SoftPOSIT algorithm described above performs a deterministic annealing search starting from an initial guess at the object's pose. Because this is a local search, there is no guarantee of finding the global optimum. The probability of finding the globally optimal object pose and correspondences starting from an initial guess depends on a number of factors including the accuracy of the initial guess, the number of model points, the number of image points, the number of occluded model points, the amount of clutter in the image, and the image measurement noise. A common method of searching for a global optimum, and the one used here, is to run the search algorithm starting from a number of different initial guesses, and keep the first solution that meets a specified termination criterion. Our initial guesses range over the range $[-\pi, \pi]$ for the three Euler rotation angles, and over a 3D space of translations known to contain the true translation. In this section, we describe our procedure for generating initial guesses for pose when no knowledge of the correct pose is available, and then we discuss our termination criterion.

5.1 Generating Initial Guesses

Given an initial pose that lies in a valley of the cost function in the parameter space, we expect the algorithm to converge to the minimum associated with that valley. To examine other valleys, we must start with points that lie in them. One possibility for generating new starting poses is to use a multi-dimensional pseudo-random number generator.

However, this leads to a set of problems. First, a pseudo random number generator will generate points that may cluster together or be far apart in the parameter space. Thus we may revisit some regions of the space that have already been studied, or we may miss some regions altogether. A possible solution is to allow for rejection of some of the generated initial parameter values. However, this adds a layer

of complexity to the software, and also does not provide a mathematical guarantee that the space is in some sense being optimally covered. Indeed, since each search for correspondence and pose is relatively expensive, we would like to have a mathematical statement that allows us to make the claim that, for a given number of starting points, our starting points sample the parameter space in some optimal manner.

Another problem with such searches is that sometimes the minima may lie in valleys that are likely to be of complex shapes, or some of the minima may be embedded in a lower-dimensional manifold in the space. If the sampling is to be successful in recovering these minima, not only must the distributions of the initial guesses sample the parameter space well, but so must their lower-dimensional projections. Intuitively, the points must be distributed such that any subvolume in the space should contain points in proportion to its volume (or other appropriate measure). This property must also hold for projections onto a manifold.

Fortunately, there exists a set of deterministic points that have such properties. These are the quasi-random, or low-discrepancy, sequences. These points are optimally self-avoiding and uniformly space filling. Uniformity of a distribution of points can be characterized by the mathematical definition of discrepancy. Let a region with unit volume have N points distributed in it. Then for uniform point distributions, any subregion with volume α should have αN points in it. The difference between this quantity and the actual number of points in the region is called the "discrepancy." Quasi-random sequences have low discrepancies and are also called low-discrepancy sequences. The error in uniformity for a sequence of N points in the *k*-dimensional unit cube is measured by its discrepancy, which is $O((\log N)^k N^{-1})$ for a quasi-random sequence, as opposed to $O((\log \log N)^{1/2} N^{-1/2})$ for a pseudo-random sequence [33].

Figure 4 compares the uniformity of distributions of quasi-random points and pseudo-random points. Figure 4a shows a set of random points generated in $(0,1)^2$ using a pseudo-random number generator. If the distribution of points were uniform one would expect that any region of area larger than $1/512$ would have at least one point in it. As can be seen, however, many regions considerably larger than this are not sampled at all, while points in other regions form rather dense clusters, thus oversampling those regions. Figure 4b shows the same number of quasi-random points for the same area. These points do not clump together, and fill the spaces left by the pseudo-random points.

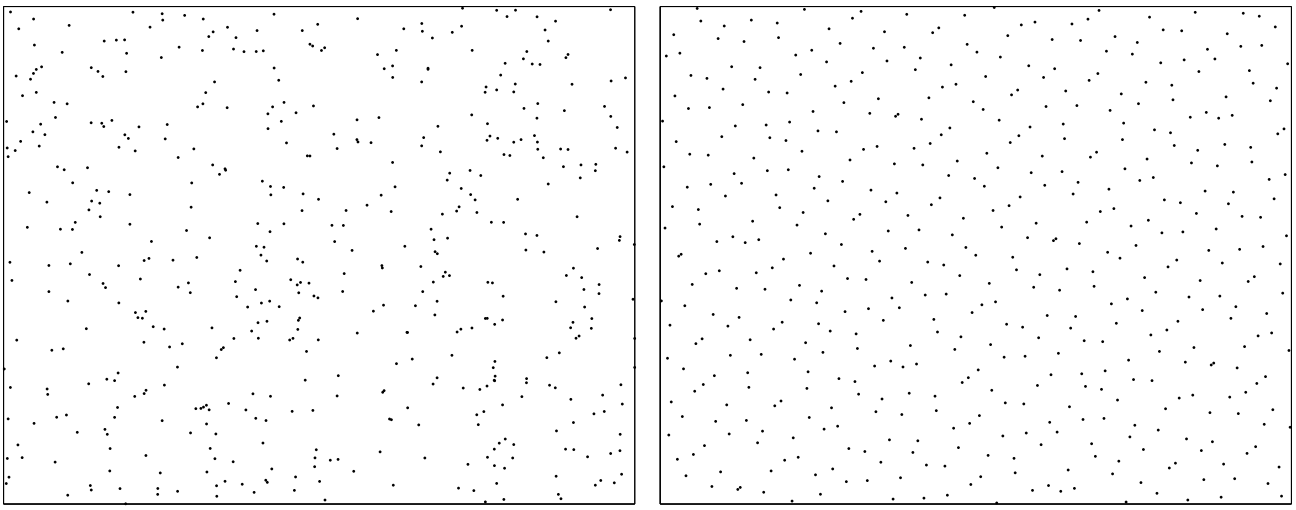
We use a standard quasi-random generator [12] to generate quasi-random 6-vectors in a unit 6D hypercube. These points are scaled to cover the expected ranges of translation and rotation.

5.2 Search Termination

Ideally, one would like to repeat the search from a new starting point whenever the number of model-to-image correspondences determined by the search is not maximal. With real data, however, one usually does not know what this maximal number is. Instead, we repeat the search when the number of model points that match image points is less than some threshold t_m. Due to occlusion and imperfect image feature extraction algorithms, not all model points will be detected as features in an image of that object. Let the fraction of detected model features be

$$p_d = \frac{\text{number of model points detected as image features}}{\text{total number of model points}}.$$

In the Monte Carlo simulations described below, p_d is known. With real imagery, however, p_d must be estimated based on the scene complexity and on the reliability of the image processing algorithm in detecting model features.



(a) (b)

Figure 4: 512 points in $(0,1)^2$ generated with (a) a pseudo-random number generator, and (b) a quasi-random number generator.

We terminate the search for better solutions when the current solution is such that the number of model points that match any image point is greater than or equal to the threshold $t_m = \rho p_d K$, where ρ determines what percent of the detected model points must be matched $(0 < \rho \leq 1)$, and K is the total number of model points, so that $p_d K$ is the number of detected model points. ρ accounts for measurement noise that typically prevents some detected model features from being matched even when a good pose is found. In the experiments discussed below, we take $\rho = 0.8$. This test is not perfect, as it is possible for a pose to be very accurate even when the number of matched points is less than this threshold; this occurs mainly in cases of high noise. Conversely, a wrong pose may be accepted when the ratio of clutter features to detected model points is high. It has been observed, however, that these situations are relatively uncommon.

We note that Grimson and Huttenlocher [23] have derived an expression for a threshold on the number of matched model points necessary to accept a local optimum; their expression is a function of the numbers of image and model points and of the sensor noise, and guarantees with a specified probability that the globally optimal solution has been found.

5.3 Early Search Termination

The deterministic annealing loop of the SoftPOSIT algorithm iterates over a range of values for the annealing parameter β. In the experiments reported here, β is initialized to $\beta_0 = 0.0004$ and is updated according to $\beta = 1.05 \times \beta$, and the annealing iteration ends when the value of β exceeds 0.5. (The iteration may end earlier if convergence is detected.) This means that the annealing loop can run for up to 147 iterations. It is usually the case that, by viewing the original image and, overlayed on top of it, the projected model points produced by SoftPOSIT, a person can determine very early (e.g., around iteration

30) whether or not the algorithm is going to converge to the correct pose. It is desired that the algorithm make this determination itself, so that whenever it detects that it seems to be heading down an unfruitful path, it can end the current search for a local optimum and restart from a new random initial condition, thereby saving a significant amount of processing time.

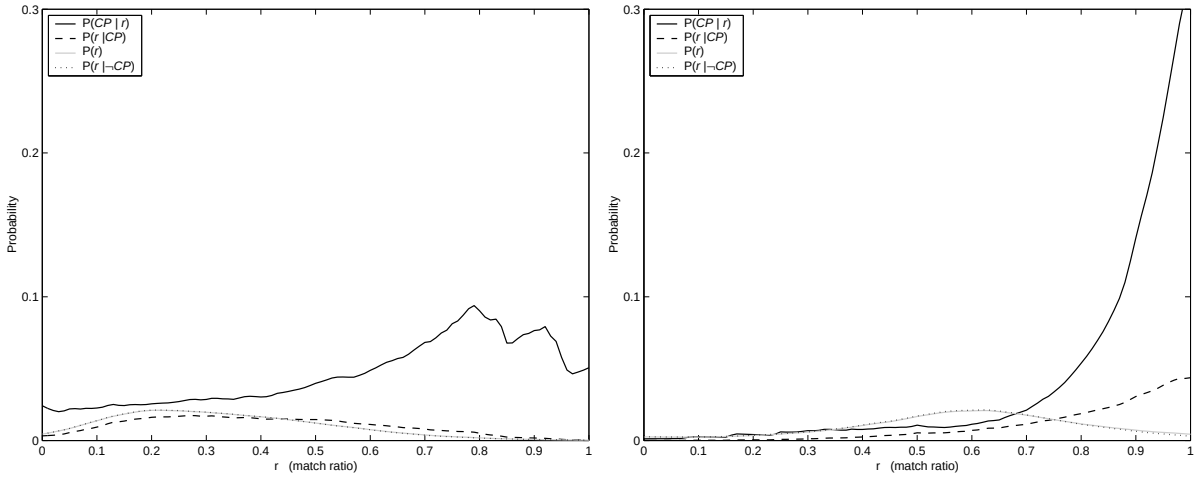
A simple test is performed at *each* iteration of SoftPOSIT to determine if it should continue with the iterations or restart. At iteration i of SoftPOSIT, the match matrix $M^i = \{m^i_{j,k}\}$ is used to predict the final correspondences of model to image points. upon convergence of SoftPOSIT, one would expect image point j to correspond to model point k if $m^i_{j,k} > m^i_{u,v}$ for all $u \neq j$ and all $v \neq k$ (though this is not guaranteed). The number of predicted correspondences at iteration i, n_i, is just the number of pairs (j, k) that satisfy this relation. We then define the match ratio at iteration i as $r_i = n_i/(p_d K)$ where p_d is the fraction of detected model features as defined above.

The early termination test centers around this match ratio measure. This measure is commonly used [23] at the end of a local search to determine if the current solution for correspondence and pose is good enough to end the search for the global optimum. We, however, use this metric within the local search itself. Let *C* denote the event that the SoftPOSIT algorithm eventually converges to the correct pose. Then the algorithm restarts after the i^{th} iteration if $P(C \mid r_i) < \alpha P(C)$, where $0 < \alpha \leq 1$. That is, the search is restarted from a new random starting condition whenever the posterior probability of eventually finding a correct pose given r_i drops to less than some fraction of the prior probability of finding the correct pose. Notice that a separate posterior probability function is required for each iteration i because the ability to predict the eventual outcome using r_i changes as the iterations progress. Although this test may result in the termination of some local searches which would have eventually produced good poses, it is expected that the total time required to find a good pose will be less. Our experiments show that this is indeed the case; we obtain a speedup by a factor of 2.

The posterior probability function for the i^{th} iteration can be computed from $P(C)$, the prior probability of finding a correct pose on one random local search, and from $P(r_i \mid C)$ and $P(r_i \mid \overline{C})$, the probabilities of observing a particular match ratio on the i^{th} iteration given that the eventual pose is either correct or incorrect, respectively:

$$P(C \mid r_i) = \frac{P(C)P(r_i \mid C)}{P(C)P(r_i \mid C) + P(\overline{C})P(r_i \mid \overline{C})}.$$

$P(C)$, $P(\overline{C})$, $P(r_i \mid C)$, and $P(r_i \mid \overline{C})$ are estimated in Monte Carlo simulations of the algorithm in which the number of model vertices and the levels of image clutter, occlusion, and noise are all varied. The details of these simulations are described in Section 6. To estimate $P(r_i \mid C)$ and $P(r_i \mid \overline{C})$, the algorithm is repeatedly run on random test data. For each test, the values of the match ratio r_i computed at each iteration are recorded. Once a SoftPOSIT iteration is completed, ground truth information is used to determine whether or not the correct pose was found. If the pose is correct, the recorded values of r_i are used to update histograms representing the probability functions $P(r_i \mid C)$; otherwise, histograms representing $P(r_i \mid \overline{C})$ are updated. Upon completing this training, the histograms are normalized. $P(C)$ is easily estimated based on the percent of the random tests that produced the correct pose. We also have $P(\overline{C}) = 1 - P(C)$. Two of these estimated probability functions are shown in Figure 5.



(a) (b)

Figure 5: Probability functions estimated for (a) the first iteration, and (b) the 31st iteration, of the SoftPOSIT algorithm.

6 Experiments

The two most important questions related to the performance of the SoftPOSIT algorithm are (a) How often does it find a "good" pose? and (b) How long does it take? Both of these issues are investigated in this section.

6.1 Monte Carlo Evaluation

The random-start SoftPOSIT algorithm has been extensively evaluated in Monte Carlo simulations. The simulations and the performance of the algorithm are discussed in this section. The simulations are characterized by the five parameters: n_t, K, p_d, p_c, and σ. n_t is the number of independent random trials to perform for each combination of values of the remaining four parameters. K is the number of points (vertices) in a 3D model. p_d is the probability that the image of any particular model point will be detected as a feature point in the image. p_d takes into account occlusion of the 3D model points as well as the fact that real image processing algorithms do not detect all desired feature points, even when the corresponding 3D points are not occluded. p_c is the probability that any particular image feature point is clutter, that is, is not the image of some 3D model point. Finally, σ is the standard deviation of the normally distributed noise in the x and y coordinates of the non-clutter feature points, measured in pixels for a 1000×1000 image, generated by a simulated camera having a 37-degree field of view (a focal length of 1500 pixels). The current tests were performed with $n_t = 100$, $K \in \{20, 30, 40, 50, 60, 70, 80\}$, $p_d \in \{0.4, 0.6, 0.8\}$, $p_c \in \{0.2, 0.4, 0.6\}$, and $\sigma \in \{0.5, 1.0, 2.5\}$[2]. With these parameters, 18,900 independent trials were

[2]Because one of our main application is autonomous navigation in cities, and because image corner points of the type produced by buildings can be located with an accuracy of $1/10$th of a pixel [5], these values of σ are larger than what is expected in real imagery.

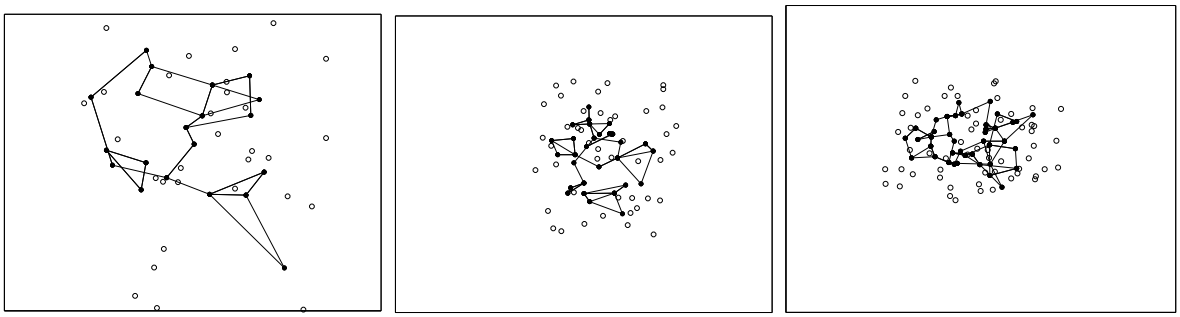
performed.

For each trial, a 3D model is created in which the K model vertices are randomly located in a sphere centered at the origin. Because the SoftPOSIT algorithm works with points, not with line segments, it is only the model vertices that are important in the current tests. However, to make the images produced by the algorithm easier to understand, each model vertex is connected by an edge to the two closest of the remaining model vertices. These connecting edges are not used by the SoftPOSIT algorithm. The model is then rotated into some arbitrary orientation, and translated to some random point in the field of view of the camera. Next, the model is projected into the image plane of the camera; each projected model point is detected with probability p_d. For those points that are detected, normally distributed noise with mean zero and standard deviation σ is added to both the x and y coordinates of the feature points. Finally, randomly located clutter feature points are added to the true (non-clutter) feature points, so that $100 \times p_c$ percent of the total number of feature points are clutter; to achieve this, $Kp_dp_c/(1-p_c)$ clutter points must be added. The clutter points are required to lie in the general vicinity of the true feature points. However, to prevent the clutter points from replacing missing true feature points, each clutter point must be further than $\sqrt{2}\sigma$ from any projected model point, whether or not the point was detected. Figure 6 shows a few examples of cluttered images of random models that are typical of those used in our experiments.

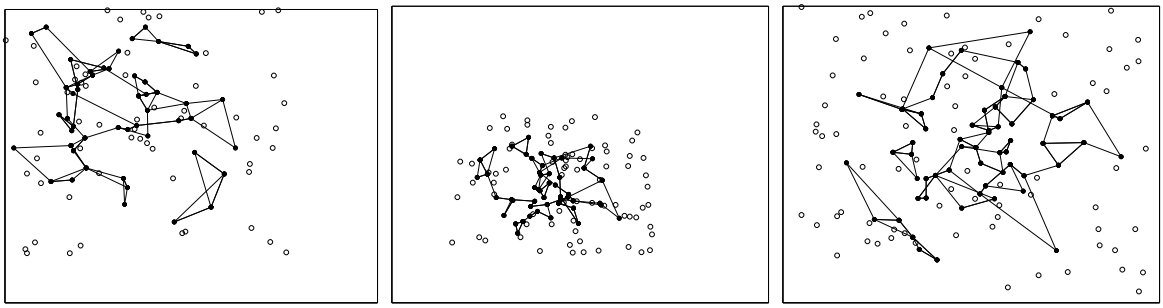
In our experiments, we consider a pose to be *good* when it allows 80% ($\rho = 0.8$ in Section 5.2) or more of the detected model points to be matched to some image point. The number of random starts for each trial was limited to 10,000. Thus, if a good pose is not found after 10,000 starts, the algorithm gives up. Figures 7 and 8 show a number of examples of poses found by SoftPOSIT when quasi-random 6-vectors are used as the initial guesses for pose.

Figure 9 shows the success rate of the algorithm (percent of trials for which a good pose was found in 10,000 starts, given no knowledge of the correct pose) as a function of the number of model points for $\sigma = 2.5$ and for all combinations of the parameters p_d and p_c. (The algorithm performs a little better for $\sigma = 0.5$ and $\sigma = 1.0$.) It can be seen from this figure that, for more than 92% of the different combinations of simulation parameters, a good pose is found in 90% or more of the associated trials. For the remaining 8% of the tests, a good pose is found in 75% or more of the trials. Overall, a good pose was found in 96.4% of the trials. As expected, the higher the occlusion rate (lower p_d) and the clutter rate (higher p_c), the lower the success rate. For the high-clutter tests, the success rate increases as the number of model points decreases. This is due to the algorithm's ability to more easily match a smaller number of model points to clutter than a larger number of model points to the same level of clutter.

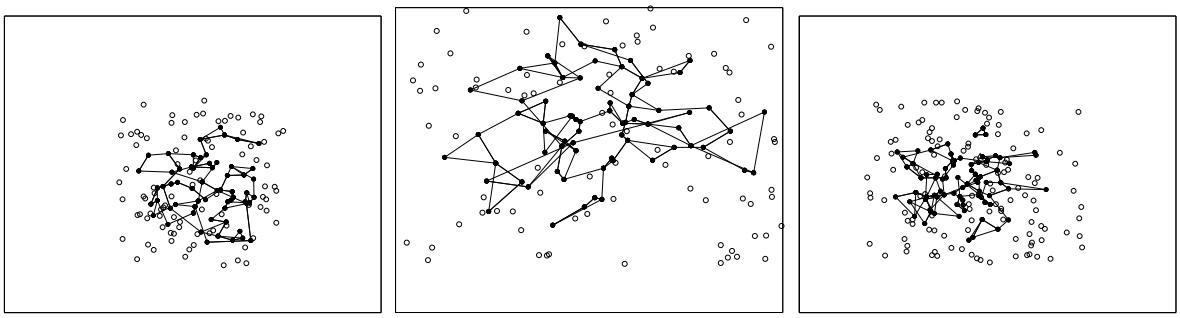
Figure 10 shows the average number of random starts required to find a good pose. These numbers generally increase with increasing image clutter and occlusion. However, for the reason given in the previous paragraph, the performance for small numbers of model points is better at higher levels of occlusion and clutter. Other than in the highest occlusion and clutter case, the mean number of starts is about constant or increases very slowly with increasing number of model points. Also, there does not appear to be any significant increase in the standard deviation of the number of random starts as the number of model points increases. The mean number of starts over all of the tests is approximately 500; the mean exceeds 1100 starts only in the single hardest case. Figure 11 shows the same data but plotted as a function of the number of image points. Again, except for the two highest occlusion and clutter cases, the mean number of starts is about constant or increases very slowly as the number of image points increases.



(a) (b) (c)



(d) (e) (f)



(g) (h) (i)

Figure 6: Typical images of randomly generated models and images. The black points are projected model points and the white points (circles) are clutter points. The black lines, which connect the model points, are included in these pictures to assist the reader in understanding the pictures; they are not used by the algorithm. The number of points in the models are 20 for (a), 30 for (b), 40 for (c), 50 for (d) and (e), 60 for (f) and (g), 70 for (h), and 80 for (i). In all cases shown here, $p_d = 1.0$ and $p_c = 0.6$. This is the best case for occlusion (none), but the worst case for clutter. In the actual experiments, p_d and p_c vary.

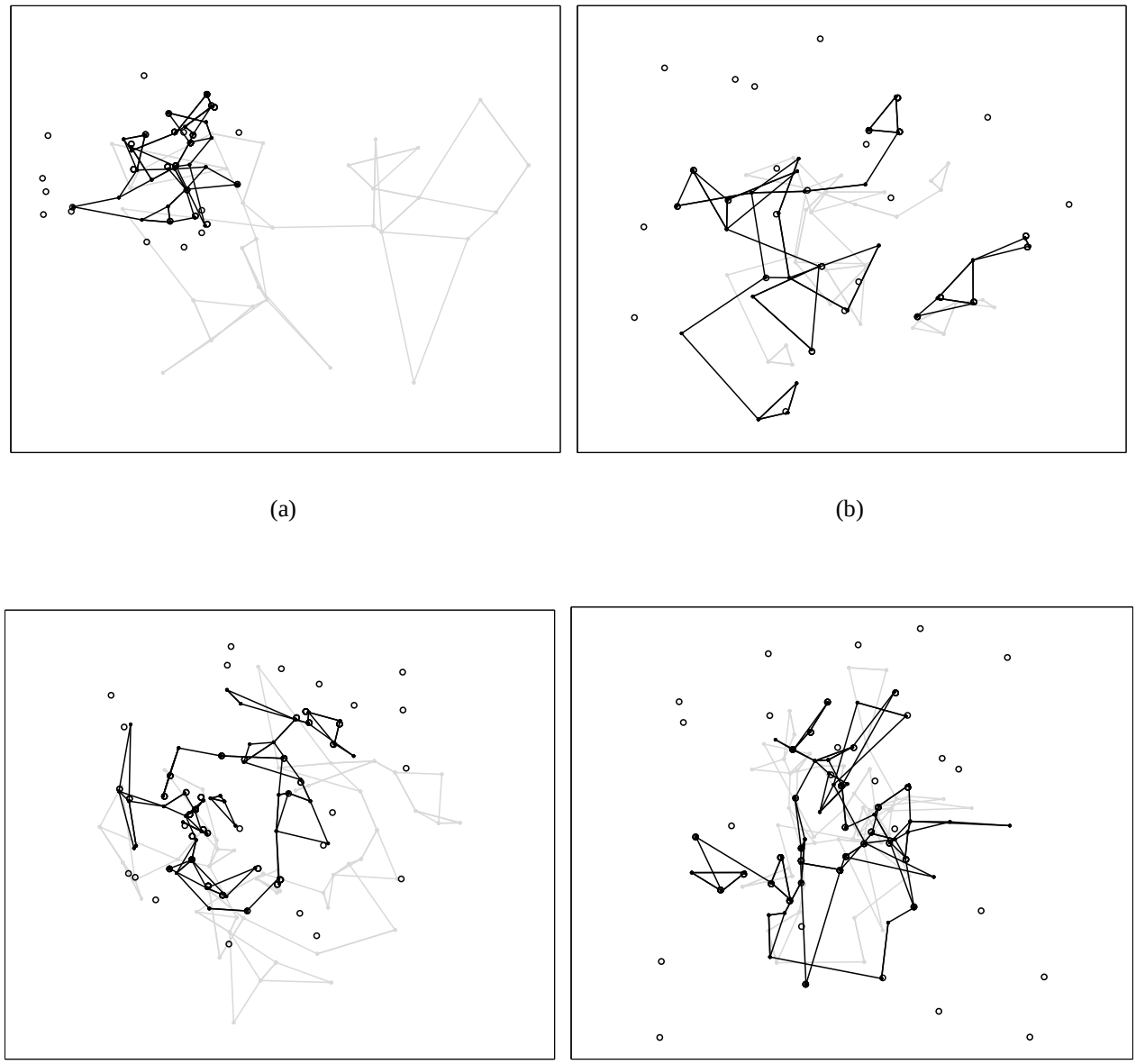
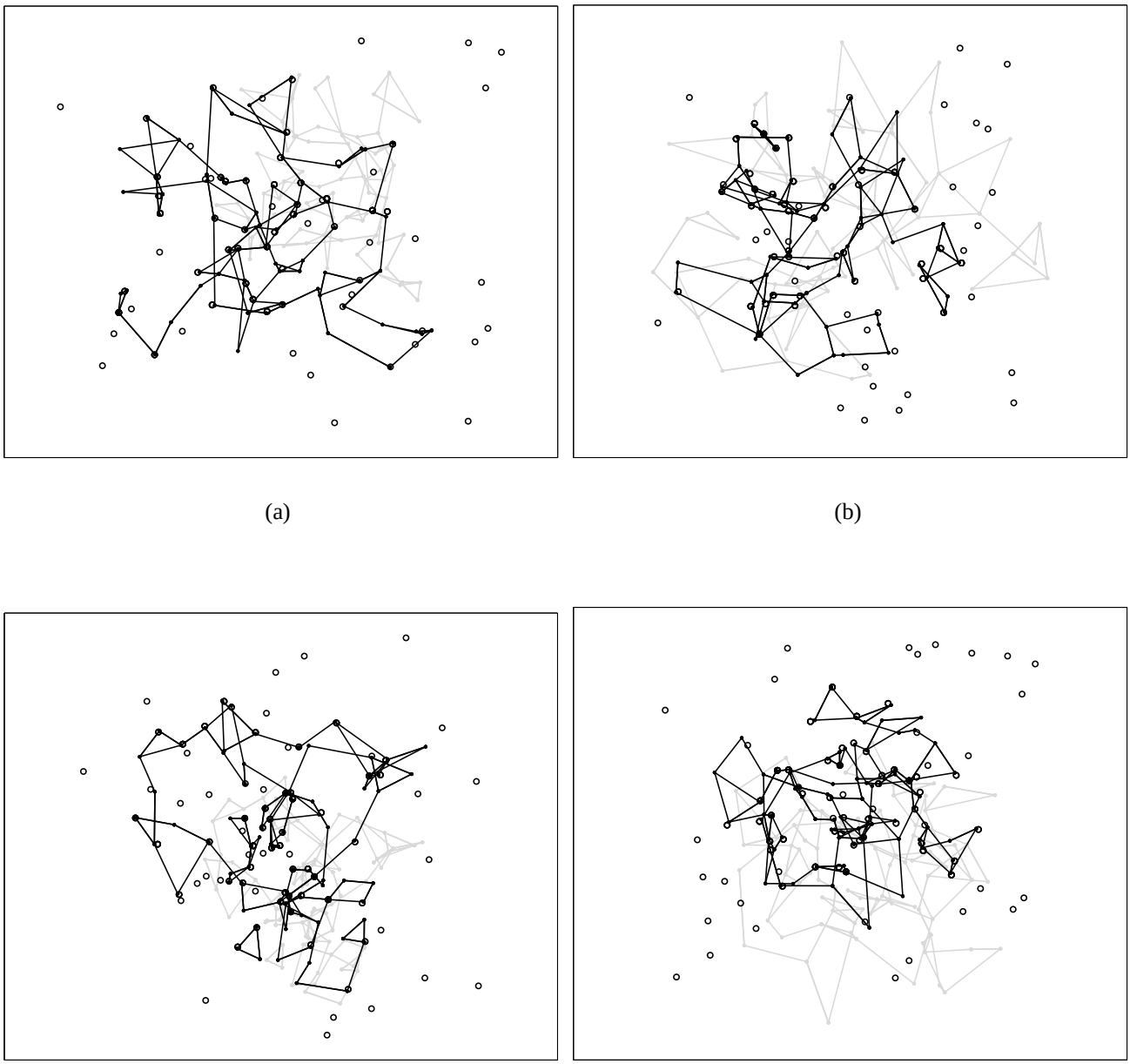


Figure 7: Some projected models and cluttered images for which SoftPOSIT was successful. The small circles are the image points (including projected model and clutter) to which the models must be matched. The light gray points and lines show the projections of the models in the initial poses (random guesses) which lead to good poses being found. The black points and lines show the projections of the models in the good poses that are found. The black points that are not near any circle are occluded model points. Circles not near any black point are clutter. Again, the gray and black lines are included in these pictures to assist the reader in understanding the pictures; they are not used by the algorithm. The Monte Carlo parameters for these tests are $p_d = 0.6$, $p_c = 0.4$, $\sigma = 2.5$, $K = 30$ for (a) and (b), $K = 50$ for (c) and (d).



(a) (b)

(c) (d)

Figure 8: More projected models and cluttered images for which SoftPOSIT was successful. The Monte Carlo parameters for these tests are $p_d = 0.6$, $p_c = 0.4$, $\sigma = 2.5$ and $K = 70$ for (a) and (b), $K = 80$ for (c) and (d).

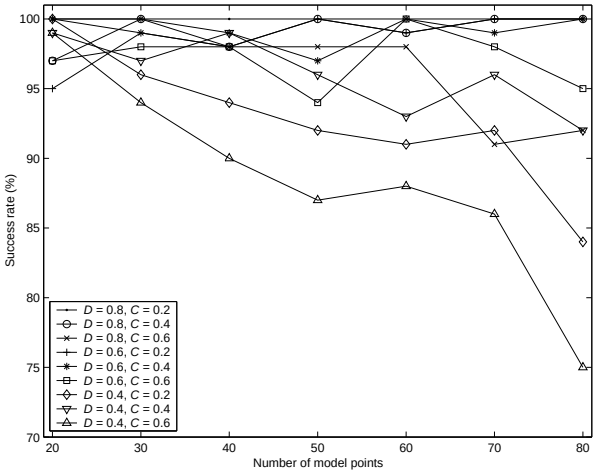
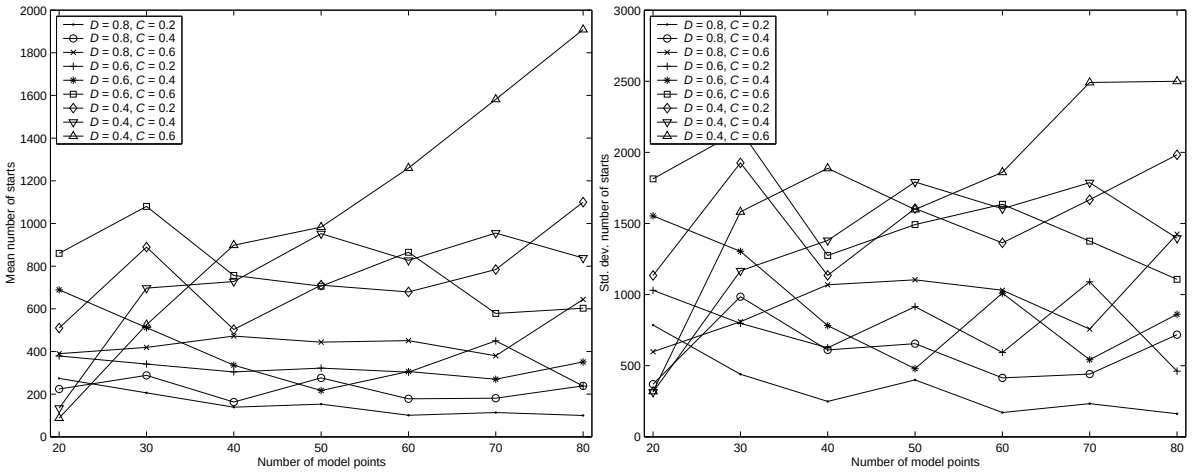
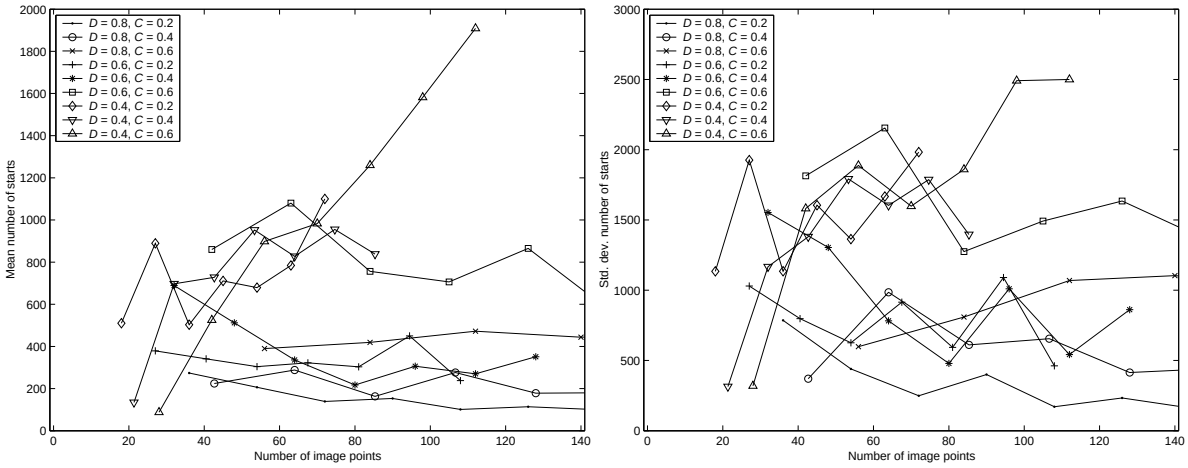


Figure 9: Success rate as a function of the number of model points for fixed values of p_d and p_c. (Note that p_d and p_c are denoted by D and C, respectively, in the legend of this figure and in the next few figures.)



(a) (b)

Figure 10: Number of random starts required to find a good pose as a function of the number of model points for fixed values of p_d and p_c. (a) Mean . (b) Standard deviation.



(a) (b)

Figure 11: Number of random starts required to find a good pose as a function of the number of image points for fixed values of p_d and p_c. (a) Mean . (b) Standard deviation.

6.2 Algorithm Complexity

The run-time complexity of a single invocation of SoftPOSIT is $O(JK)$ where J is the number of image points and K is the number of model points; this is because the numbers of iterations on all of the loops in the pseudocode in Section 4.3 are bounded by a constant, and each line inside a loop is computed in time at most $O(JK)$. As shown in Figures 10 and 11, the mean number of random starts (invocations of SoftPOSIT) required to find a good pose, to ensure a probability of success of at least 0.95, appears to be bounded by a function that is linear in the size of the input. That is, the mean number of random starts is $O(J)$, assuming that $K < J$, as is normally the case. Then the run-time complexity of SoftPOSIT with random starts is $O(J^2K)$. This is a factor of J better than the complexity of any published algorithm that solves the simultaneous pose and correspondence problem under a full perspective camera model.

6.3 Experiments with Images

6.3.1 Autonomous Navigation Application

The SoftPosit algorithm is being applied to the problem of autonomous vehicle navigation through a city where a 3D architectural model of the city is registered to images obtained from an on-board video camera. Thus far, the algorithm has been applied only to imagery generated by a commercial virtual reality system. Figure 12 shows an image generated by this system and a world model projected into that image using the pose computed by SoftPOSIT. Image feature points are automatically located in the image by detecting corners along the boundary of bright sky regions. Because the 3D world model has over 100,000 data points, we use a rough pose estimate (such as might be generated by an onboard navigation system) to cull the majority of model points that don't project into the estimated field of view. Then the world points that do fall into this estimated field are further culled by keeping only those that project near

(a) (b)

Figure 12: (a) Original image from a virtual reality system. (b) World model (white lines) projected into this image using the pose computed by SoftPOSIT.

the detected skyline. So far, the results have been very good. Although this is not real imagery, the virtual reality system used is very sophisticated, and as such, should give a good indication of how the system will perform on real imagery, which we are currently in the process of acquiring.

6.3.2 Robot Docking Application

The robot docking application requires that a small robot drive onto a docking platform that is mounted on a larger robot. Figure 13 shows a small robot docking onto a larger robot. In order to accomplish this, the small robot must determine the relative pose of the large robot. This is done by using SoftPOSIT to align a 3D model of the large robot to corner points extracted from an image of the large robot.

The model of the large robot consists of a set of 3D points that are extracted from a triangular faceted model of the robot which was generated by a commercial CAD system. To detect the corresponding points in the image, lines are first detected using a combination of the Canny edge detector, the Hough transform, and a sorting procedure used to rank the lines produced by the Hough transform. Corners are then found at the intersections of those lines that satisfy simple length, proximity, and angle constraints. Figure 14 shows the lines and corner points detected in one image of the large robot. In this test there are 70 points in the model; 89% of these are occluded (or not detected in the image), and 58% of the image points are clutter. Figure 15a shows the initial guess generated by SoftPOSIT which led to the correct pose being found, and Figure 15b shows this correct pose.

7 Conclusions

We have developed and evaluated the SoftPOSIT algorithm for determining the poses of objects from images. The correspondence and pose calculation combines into one efficient iterative process the soft-assign algorithm for determining correspondences and the POSIT algorithm for determining pose. This

Figure 13: A small robot docking onto a larger robot.

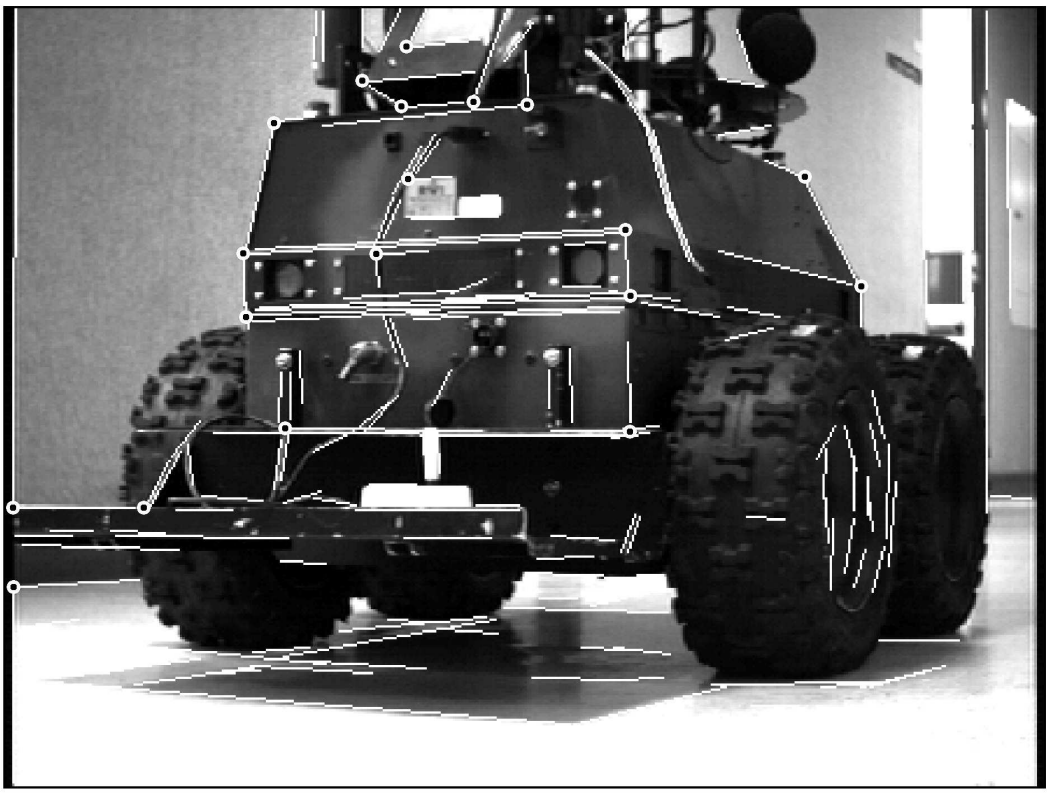
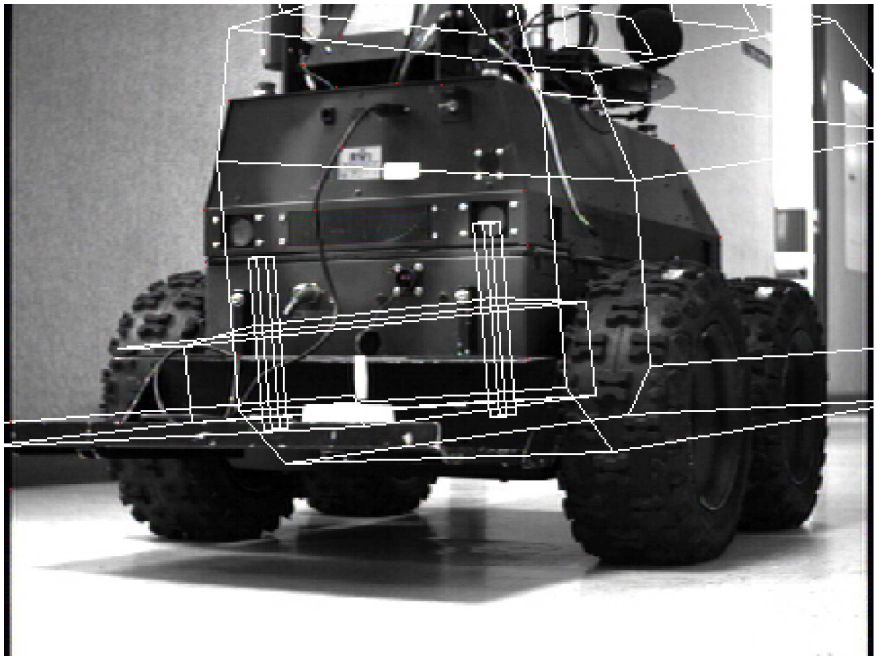
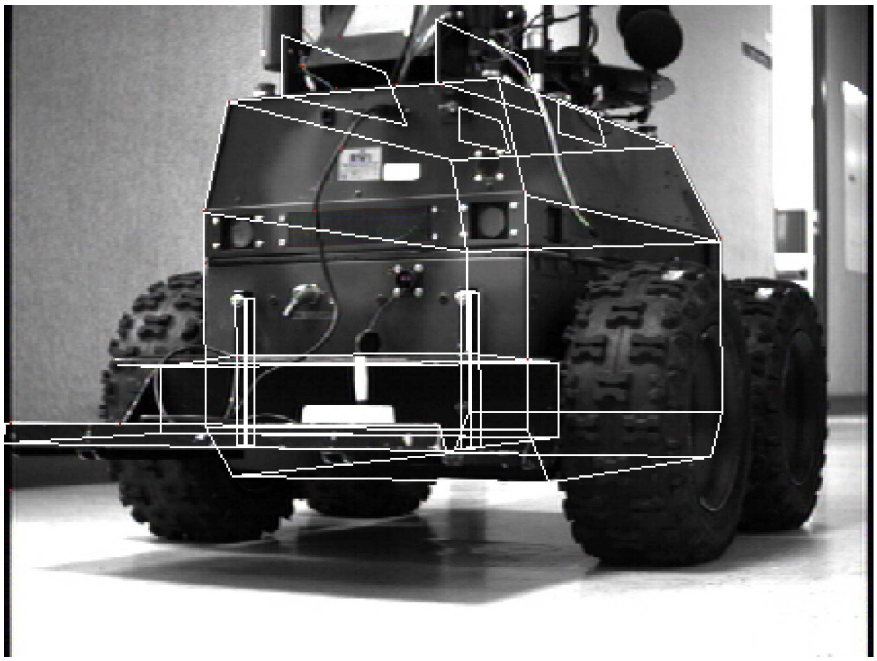


Figure 14: An image of the large robot as seen from the small robot's point of view. Long straight lines detected in the image are shown in white, and their intersections, which ideally should correspond to vertices in the 3D model, are shown in black.



(a)



(b)

Figure 15: The initial guess at the robot's pose (a) that leads to the correct pose as shown in (b).

algorithm will be used as a component in an object recognition system.

Our evaluation indicates that the algorithm performs well under a variety of levels of occlusion, clutter, and noise. The algorithm has been tested on synthetic data for an autonomous navigation application, and we are currently collecting real imagery for further tests with this application. The algorithm has also been tested in an autonomous docking application with good results.

The complexity of SoftPOSIT has been empirically determined to be $O(J^2K)$. This is better than any known algorithm that solves the simultaneous pose and correspondence problem for a full perspective camera model. More data should be collected to further validate this claim.

Future work will involve extending the SoftPOSIT algorithm to work with lines in addition to points. We are also interested in performing a more thorough comparison of the performance of SoftPOSIT to that of competing algorithms.

Appendix A The Complexity of the Hypothesize-And-Test Approach

The asymptotic complexity of the general hypothesize-and-test approach to model-to-image registration is derived in this appendix. We first define a few parameters. Let

J be the number of image points,
K be the number of 3D model points,
p_d be the fraction of model points that are present (non-occluded) in the image,
R be the desired probability of success (i.e., of finding a good pose).

Given a set of data with outlier rate w, it is well known [18] that the number k of random samples of the data of size n that must be examined in order to ensure with probability z that at least one of those samples is outlier-free is

$$k = \frac{\log(1-z)}{\log(1-(1-w)^n)}.$$

We need to determine how this number of samples depends on K, J, p_d, and R for the hypothesize-and-test algorithm for large values of J and K.

Because we assume that the hypothesize-and-test algorithm has no *apriori* information about which correspondences are correct, correspondences are formed from randomly chosen model and image points. We assume that three correspondences are used to estimate the object's pose. Let $S = p_d K$ be the number of detected (non-occluded) model points in the image. For a correspondence to be correct, the model point must be non-occluded and the image point must correspond to the model point. The probability that the n^{th} ($n = 1, 2, 3$) randomly chosen correspondence is correct given that all previously chosen correspondences are also correct is

$$\frac{S-n+1}{K-n+1} \cdot \frac{1}{J-n+1}.$$

Then the probability that any sample consists of three correct correspondences is

$$\frac{S(S-1)(S-2)}{K(K-1)(K-2)J(J-1)(J-2)} \approx \frac{S^3}{K^3J^3} = (\frac{p_d}{J})^3.$$

The probability that each of T random samples is bad (i.e., each includes at least one incorrect correspondence) is

$$(1 - (\frac{p_d}{J})^3)^T.$$

Thus to ensure with probability *R* that at least one of the randomly chosen samples consists of three correct correspondences, we must examine *T* samples where

$$1 - (1 - (\frac{p_d}{J})^3)^T \geq R.$$

Solving for *T*, we get

$$T \geq \frac{\log(1-R)}{\log(1-(\frac{p_d}{J})^3)}.$$

Noting that $(p_d/J)^3$ is always less that 10^{-4} in our experiments, and using the approximation $\log(1-x) \approx -x$ for x small, the number of samples that need to be examined is

$$T \approx (\frac{J}{p_d})^3 \log(\frac{1}{1-R}).$$

Since each sample requires $O(JK)$ time for back-projection and verification, the complexity of the general hypothesize-and-test algorithm is

$$(\frac{J}{p_d})^3 \log(\frac{1}{1-R}) \times O(JK) = O(J^4K).$$

Appendix B Scaled Orthographic Image Points

Here we give a geometric interpretation of the relation between perspective and scaled orthographic image points. Consider Figure 3. A plane Π' parallel to the image plane Π is chosen to pass through the origin P_0 of the object coordinate system. This plane cuts the camera axis at H ($OH = T_z$). The point P projects into P' on plane Π', and the image of P' on the image plane Π is called p'.

A plane Π'', also parallel to the image plane Π, passes through point P and cuts the line of sight L at P_L. The point P_L projects onto the plane Π' at P'', and the image of P'' on the image plane Π is called p''.

The plane defined by line L and the camera axis is chosen as the plane of the figure. Therefore, the image points p and p'' are also in the plane of the figure. Generally P_0 and P are out of the plane of the figure, and therefore p' is also out of the plane of the figure.

Consider again the equations of perspective (equations (1, 2)):

$$\begin{bmatrix} wx \\ wy \end{bmatrix} = \begin{bmatrix} s\mathbf{R}_1^T & sT_x \\ s\mathbf{R}_2^T & sT_y \end{bmatrix} \begin{bmatrix} \boldsymbol{P_0P} \\ 1 \end{bmatrix}. \tag{15}$$

with $w = \mathbf{R}_3 \cdot \boldsymbol{P_0P}/T_z + 1$. We can see that $\boldsymbol{cp'} = s(\mathbf{R}_1 \cdot \boldsymbol{P_0P} + T_x, \mathbf{R}_2 \cdot \boldsymbol{P_0P} + T_y)$. Indeed, the terms in parentheses are the x and y camera coordinates of P and therefore also of P', and the factor s

scales down these coordinates to those of the image p' of P'. In other words, the column vector of the right-hand side of equation (15) represents the vector $\boldsymbol{cp'}$ in the image plane.

On the other hand, $\boldsymbol{cp''} = (wx, wy) = w\boldsymbol{cp}$. Indeed the z-coordinate of P in the camera coordinate system is $\mathbf{R}_3 \cdot \boldsymbol{P_0P} + T_z$, i.e. wT_z. It is also the z-coordinate of P_L. Therefore $\boldsymbol{OP_L} = wT_z\boldsymbol{Op}/f$. The x and y camera coordinates of P_L are also those of P'', and the factor $s = f/T_z$ scales down these coordinates to those of the image p'' of P''. Thus $\boldsymbol{cp''} = w\boldsymbol{cp}$. In other words, the column vector of the left-hand side of equation (15) represents the vector $\boldsymbol{cp''}$ in the image plane. The image point p'' can be interpreted as a correction of the image point p from a perspective projection to a scaled orthographic projection of a point P_L located on the line of sight at the same distance as P.

References

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

[2] J.S. Beis and D.G. Lowe, "Indexing Without Invariants in 3D Object Recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 1000–1015, 1999.

[3] J.R. Beveridge and E.M. Riseman, "Hybrid Weak-Perspective and Full-Perspective Matching, ", *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 432–438, 1992.

[4] J.R. Beveridge and E.M. Riseman, "Optimal Geometric Model Matching Under Full 3D Perspective," *Computer Vision and Image Understanding*, vol. 61, pp. 351–364, 1995.

[5] P. Brand and R. Mohr, "Accuracy in Image Measure," *Proc. SPIE, Videometrics III*, pp. 218–228, 1994.

[6] J.S. Bridle, "Training Stochastic Model Recognition as Networks can Lead to Maximum Mutual Information Estimation of Parameters", *Advances in Neural Information Processing Systems*, vol. 2, pp. 211–217, 1990.

[7] J.B. Burns, R.S. Weiss, and E.M. Riseman, "View Variation of Point-Set and Line-Segment Features," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 51–68, 1993.

[8] T.M. Breuel, "Fast Recognition using Adaptive Subdivisions of Transformation Space," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 445–451, 1992.

[9] T.A. Cass, "Polynomial-Time Object Recognition in the Pressense of Clutter, Occlusion, and Uncertainty," *Proc. European Conf. on Computer Vision*, pp. 834–842, 1992.

[10] T.A. Cass, "Robust Geometric Matching for 3D Object Recognition," *Proc. 12th IAPR Int. Conf. on Pattern Recognition*, vol. 1, pp. 477 –482, 1994.

[11] T.A. Cass, "Robust Affine Structure Matching for 3D Object Recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 1265–1274, 1998.

[12] "C" code for quasi-random number generation. Available from http://www.taygeta.com.

[13] D. DeMenthon and L.S. Davis, "Recognition and Tracking of 3D Objects by 1D Search," *Proc. DARPA Image Understanding Workshop*, 1993.

[14] D. DeMenthon and L.S. Davis, "Model-Based Object Pose in 25 Lines of Code", *International Journal of Computer Vision,* vol. 15, pp. 123–141, 1995.

[15] D. DeMenthon and P. David, "SoftPOSIT: An Algorithm for Registration of 3D Models to Noisy Perspective Images Combining Softassign and POSIT", University of Maryland Technical Report CAR-TR-970, 2001.

[16] R.W. Ely, J.A. Digirolamo, and J.C. Lundgren, "Model Supported Positioning," *Proc. SPIE, Integrating Photogrammetric Techniques with Scene Analysis and Machine Vision II*, 1995.

[17] P.D. Fiore, "Efficient Linear Solution of Exterior Orientation," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 23, pp. 140–148, 2001.

[18] M.A. Fischler and R.C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Comm. Association for Computing Machinery,* vol. 24, pp. 381–395, 1981.

[19] D. Geiger and A.L. Yuille, "A Common Framework for Image Segmentation", *International Journal of Computer Vision,* vol. 6, pp. 227–243, 1991.

[20] S. Gold and A. Rangarajan, "A Graduated Assignment Algorithm for Graph Matching", *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 18, pp. 377–388, 1996.

[21] S. Gold, A. Rangarajan, C.P. Lu, S. Pappu, and E. Mjolsness, "New Algorithms for 2D and 3D Point Matching: Pose Estimation and Correspondence", *Pattern Recognition,* vol. 31, pp. 1019–1031, 1998.

[22] E. Grimson, *Object Recognition by Computer: The Role of Geometric Constraints,* MIT Press, Cambridge, MA, 1990.

[23] E. Grimson and D.P. Huttenlocher, "On the Verification of Hypothesized Matches in Model-Based Recognition", *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 13, pp. 1201–1213, 1991.

[24] R.M. Haralick, C. Lee, K. Ottenberg, and M. Nolle, "Analysis and Solutions of the Three Point Perspective Pose Estimation Problem," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* pp. 592–598, 1991.

[25] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* Cambridge University Press, Cambridge, UK, 2000.

[26] R. Horaud, B. Conio, O. Leboulleux, and B. Lacolle, "An Analytic Solution for the Perspective 4-Point Problem," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* pp. 500–507, 1989.

[27] B.K.P. Horn, *Robot Vision*. MIT Press, Cambridge, MA, 1986.

[28] D.W. Jacobs, "Space Efficient 3D Model Indexing," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* pp. 439–444, 1992.

[29] F. Jurie, "Solution of the Simultaneous Pose and Correspondence Problem Using Gaussian Error Model," *Computer Vision and Image Understanding,* vol. 73, pp. 357–373, 1999.

[30] Y. Lamdan and H.J. Wolfson, "Geometric Hashing: A General and Efficient Model-Based Recognition Scheme," *Proc. IEEE Int. Conf. on Computer Vision,* pp. 238-249, 1998.

[31] C.-P. Lu, G.D. Hager, and E. Mjolsness, "Fast and Globally Convergent Pose Estimation from Video Images," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 22, pp. 610–622, 2000.

[32] T.K. Moon, "The Expectation-Maximization Algorithm," *IEEE Signal Processing Magazine,* November 1996, pp. 47–60.

[33] W.J. Morokoff and R. E. Caflisch, "Quasi-Random Sequences and their Discrepancies", *SIAM J. Sci. Comput.*, pp. 1251–1279, 1994.

[34] H. Murase and S.K. Nayar, "Visual Learning and Recognition of 3-D Objects from Appearance," *Int. Journal of Computer Vision,* vol. 14, pp. 5–24, 1995.

[35] C.F. Olson, "Efficient Pose Clustering Using a Randomized Algorithm," *Int. Journal of Computer Vision,* vol. 23, pp. 131–147, 1997.

[36] S. Procter and J. Illingworth, "ForeSight: Fast Object Recognition using Geometric Hashing with Edge-Triple Features," *Proc. Int. Conf. on Image Processing,* vol. 1, pp. 889–892, 1997.

[37] R. Sinkhorn, "A Relationship between Arbitrary Positive Matrices and Doubly Stochastic Matrices", *Annals Math. Statist.*, vol. 35, pp. 876–879, 1964.

[38] S. Ullman, "Aligning Pictorial Descriptions: An Approach to Object Recognition," *Cognition,* vol. 32, pp. 193–254, 1989.

[39] P. Wunsch and G. Hirzinger, "Registration of CAD Models to Images by Iterative Inverse Perspective Matching", *Proc. Int. Conf. on Pattern Recognition*, pp. 78–83, 1996.

[40] J.-C. Yuan, "A General Photogrammetric Method for Determining Object Position and Orientation," *IEEE Trans. on Robotics and Automation,* vol. 5, pp. 129-142, 1989.