SoftPOSIT: Simultaneous Pose and Correspondence Determination

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 object points and image points are not known. The algorithm combines the iterative *softassign* algorithm [Gold 1996, Gold 1998] for computing correspondences and the iterative POSIT algorithm [DeMenthon 1995] 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.

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 object 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 object 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 [Fischler 1981, Haralick 1991, Horaud 1989, Yuan 1989]. For six or more matches, linear and approximate nonlinear methods are generally used [DeMenthon 1995, Fiore 2001, Hartley 2000, Horn 1986, Lu 2000].

2. Solving the *correspondence* problem consists of finding matching object and image features. If the object pose is known, one can relatively easily determine the matching features. Projecting the object in the known pose into the original image, one can identify matches among the object

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 [Grimson 1991].

The classic approach to solving these coupled problems is the hypothesize-and-test approach [Grimson 1990]. In this approach, a small set of object feature to image feature correspondences are first hypothesized. Based on these correspondences, the pose of the object is computed. Using this pose, the object 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 [Fischler 1981] for the case that no information is available to constrain the correspondences of object 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 $O(MN^3 \log N)$ time when there are M object points and N image points (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 [Murase 1995] 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., [Cass 1994, Cass 1998, Ely 1995, Jacobs 1992, Lamdan 1988, Procter 1997]) 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 [Baird 1985], with exponential time complexity for unequal point sets, and Ullman's alignment method [Ullman 1989] with time complexity $O(N^4 M^3 \log M)$.

The geometric hashing method [Lamdan 1988] 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) [Burns 1993], this method can only be applied to planar scenes.

In [DeMenthon 1993], we proposed an approach using binary search by bisection of pose boxes in two 4D spaces, extending the research of [Baird 1985, Cass 1992, Breuel 1992] on affine transforms, but it had high-order complexity. The approach taken by Jurie [Jurie 1999] 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 [DeMenthon 1993], 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 [Wunsch 1996] 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 object 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 object feature, and the pose that brings the object 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 [Beis 1999] 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 object-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 [Olson 1997] gives a randomized algorithm for pose clustering whose time complexity is $O(MN^3)$.

The method of Beveridge and Riseman [Beveridge 1992, Beveridge 1995] 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 object-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 object's pose after making a move to a new set of correspondences. The time complexity of this algorithm was empirically determined to be $O(M^2N^2)$.

When there are M object points and N image points, the dimension of the solution space for this problem is $M + 6$ since there are M correspondence variables and 6 pose variables. Each correspondence variable has the domain $\{1, 2, \ldots, N, \emptyset\}$ representing a match of an object point to one of the N image points or to no image point (represented by $\emptyset$), and each pose variable has a continuous domain determined by the allowed range of object translations and rotations. Most algorithms don't explicitly search this $M + 6$-dimensional space, but instead assume that pose is determined by correspondences or that correspondences are determined by pose, and so search either an M-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 [Gold 1996, Gold 1998] when they solved the correspondence and pose problem in matching two images or two 3D objects. We extend it to the more difficult problem of registration between a 3D object 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) [DeMenthon 1995], and an iterative correspondence assignment technique called *softassign* [Gold 1996, Gold 1998] 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 an object with 15 points. Notice that it would be impossible to make hard correspondence decisions for the initial pose (frame 1), where the object's 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 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 A New Formulation of the POSIT Algorithm

One of the building blocks of the SoftPOSIT is the POSIT algorithm, presented in detail in [DeMenthon 1995], which determines pose from known correspondences. The presentation given in [DeMenthon 1995] requires that an object point with a known image be selected as the origin of the object coordinate system. This is possible with POSIT because correspondences are assumed to be known. Later, however, when we assume that correspondences are unknown, this will not be possible. Hence, we give a new formulation of the POSIT algorithm below that has no preferential treatment of the object origin, and then we present a variant of this 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). The point p is the perspective projection of the 3D object point P with homogeneous coordinates $\mathbf{P} = (X, Y, Z, 1)^{\mathsf{T}}$ in the frame of reference of the object whose origin is at P_0 in the camera frame. See Figure 3. The Euclidean coordinates of P in the object frame are represented by the vector $\widetilde{\mathbf{P}} = (X, Y, Z)^{\mathsf{T}}$ from P_0 to P.

In our problem, there is an unknown coordinate transformation between the object and the camera,

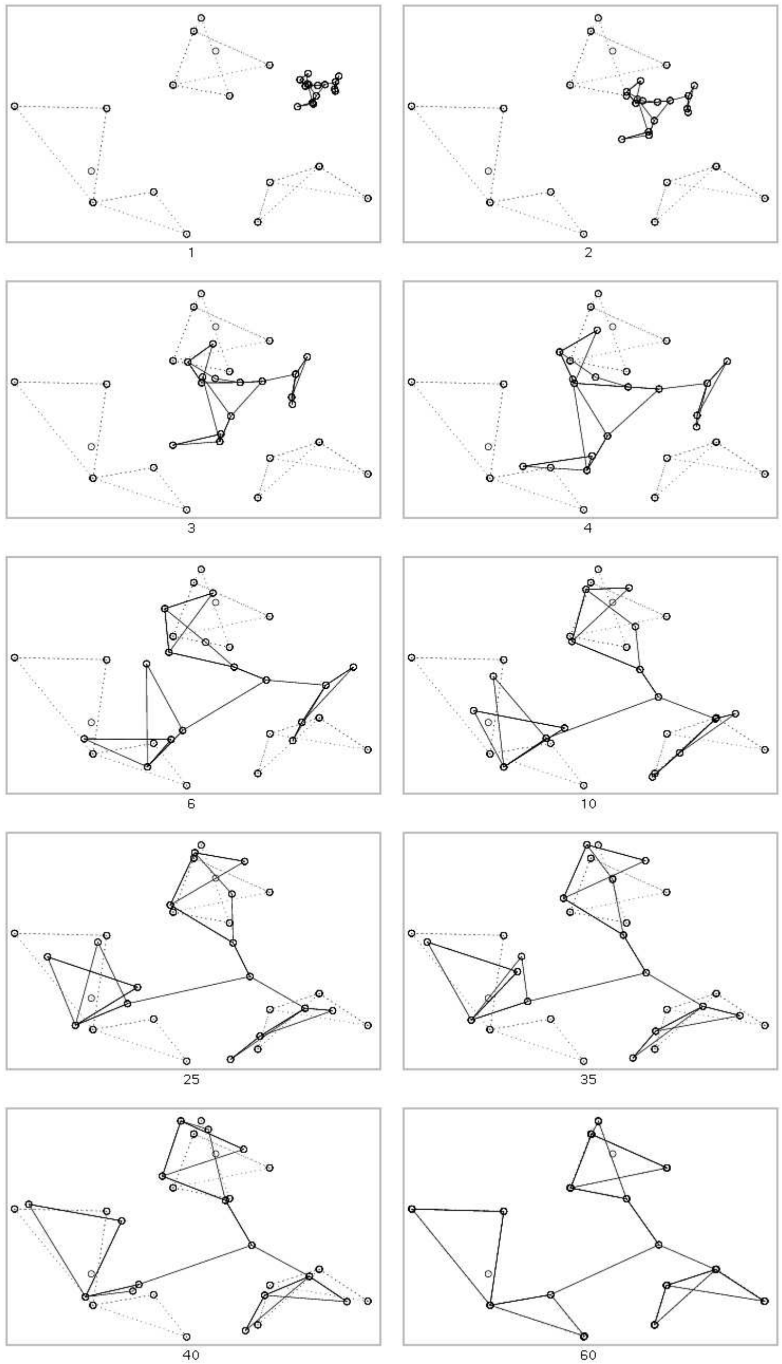


Figure 1: Evolution of perspective projections for a 15-point object (solid lines) being aligned by the SoftPOSIT algorithm to an image (dashed lines) with one occluded object point and two clutter points. The iteration step of the algorithm is shown under each frame.

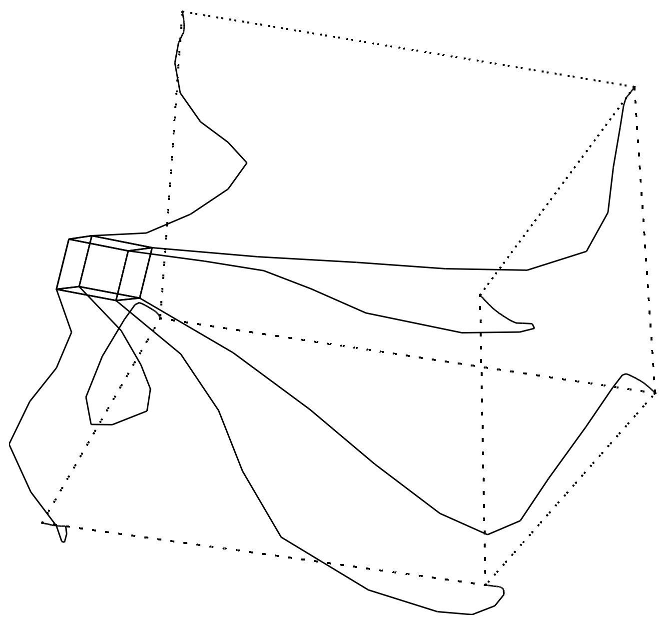


Figure 2: The trajectory of the perspective projection of a cube (solid lines) being aligned by the SoftPOSIT algorithm to an image of a cube (dashed lines), where one vertex of the cube is occluded. A simple object is used for the sake of clarity.

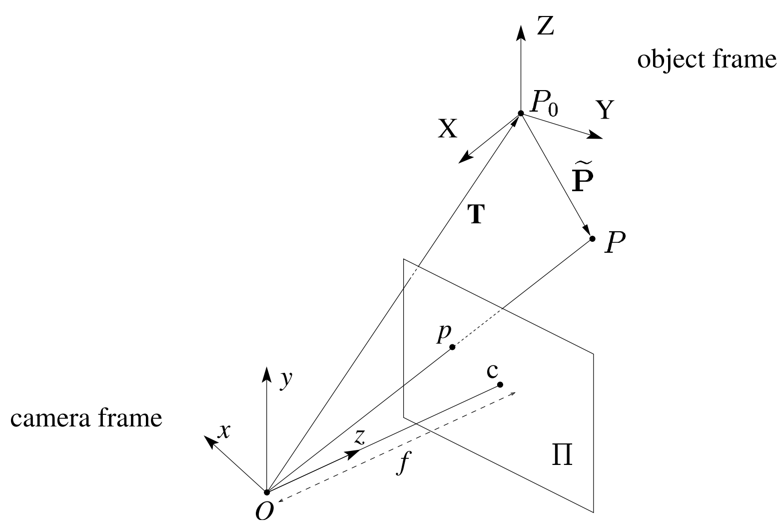


Figure 3: Camera geometry. A camera with center of projection O, focal length f, image center c, and image plane Π, projects object point P onto image point p. $\mathbf{T}$ is the translation between the camera frame and the object frame, whose origin is at P_0 with respect to the camera frame. The coordinates of point P with respect to the object frame are given by the 3-vector $\widetilde{\mathbf{P}}$.

represented by a rotation matrix $R = [\mathbf{R}_1\,\mathbf{R}_2\,\mathbf{R}_3]^\mathsf{T}$ and a translation vector $\mathbf{T} = (T_x, T_y, T_z)^\mathsf{T}$. The vectors $\mathbf{R}_1^\mathsf{T}$, $\mathbf{R}_2^\mathsf{T}$, $\mathbf{R}_3^\mathsf{T}$ are the row vectors of the rotation matrix; they are the unit vectors of the camera coordinate system expressed in the object 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. The coordinates of the perspective image point p can be shown to be related to the coordinates of the object point P by

$$\begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} f\mathbf{R}_1^\mathsf{T} & fT_x \\ f\mathbf{R}_2^\mathsf{T} & fT_y \\ \mathbf{R}_3^\mathsf{T} & T_z \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{P}} \\ 1 \end{bmatrix}.$$

The homogeneous image point coordinates are defined up to a multiplicative constant; therefore the validity of the equality is not affected if we multiply 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

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

with

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

In the expression for w the dot product $\mathbf{R}_3 \cdot \widetilde{\mathbf{P}}$ represents the projection of the vector $\widetilde{\mathbf{P}}$ onto the optical axis of the camera. Indeed, in the object coordinate system where P is defined, $\mathbf{R}_3$ is the unit vector of the optical axis. When the depth range of the object along the optical axis of the camera is small with respect to the object distance, $\mathbf{R}_3 \cdot \widetilde{\mathbf{P}}$ 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:

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

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 an object coordinate system to a camera coordinate system, and uses two of the three object 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 (Eq. (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 object coordinates X, Y, Z of $\widetilde{\mathbf{P}}$. If we know M object points P_k, $k = 1, \ldots, M$, with Euclidean coordinates $\widetilde{\mathbf{P}}_k = (X_k, Y_k, Z_k)^{\mathsf{T}}$, their corresponding image points p_k, and their homogeneous components w_k, then we can then write two linear systems of M 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 object point coordinates is at least 4. Thus, at least four of the points of the object 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 $\mathbf{R}_2$:

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

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

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

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

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 [DeMenthon 2001]).

How can we compute the w_k components in Equation (4) that determine 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 4, 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. 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

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

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

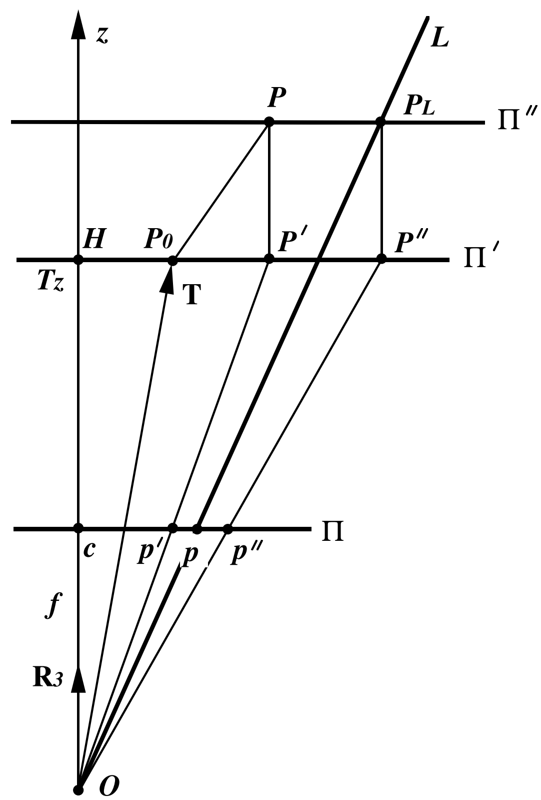


Figure 4: Geometric interpretation of the POSIT computation. Image point p', the scaled orthographic projection of object point P, is computed by the left-hand side of equation (4). Image point p'', the scaled orthographic projection of point P_L on the line of sight of p, is computed by the right-hand side of this equation. The equation is satisfied when the two points are superposed, which requires that the object 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.

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 a set of object points P_k, $k = 1, \ldots, M$, to the lines of sight L_k of their 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 $d_k^2 = \mid \boldsymbol{p'_k p''_k} \mid^2$ between image points p'_k and p''_k:

$$
\begin{aligned}
E &= \sum_k d_k^2 = \sum_k \left| \boldsymbol{cp'_k} - \boldsymbol{cp''_k} \right|^2 \\
&= \sum_k ((\mathbf{Q}_1 \cdot \mathbf{P}_k - w_k x_k)^2 + (\mathbf{Q}_2 \cdot \mathbf{P}_k - w_k y_k)^2)
\end{aligned}
\qquad (5)
$$

where we have introduced the vectors $\mathbf{Q}_1$, $\mathbf{Q}_2$, and $\mathbf{P}_k$ with four homogeneous coordinates to simplify the subsequent notation:

$$
\begin{aligned}
\mathbf{Q}_1 &= s(\mathbf{R}_1, T_x), \\
\mathbf{Q}_2 &= s(\mathbf{R}_2, T_y), \\
\mathbf{P}_k &= (\widetilde{\mathbf{P}}_k, 1).
\end{aligned}
$$

We call $\mathbf{Q}_1$ and $\mathbf{Q}_2$ the *pose vectors*.

Referring again to Figure 4, 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 object 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 $\mathbf{Q}_1$ and $\mathbf{Q}_2$ assuming the terms w_k are known (Eq. (5)).

2. Compute the correction terms w_k using the pose vectors $\mathbf{Q}_1$ and $\mathbf{Q}_2$ just computed (Eq. (2)).

We now focus on the optimization of the pose vectors $\mathbf{Q}_1$ and $\mathbf{Q}_2$. 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 $\mathbf{Q}_1$ and $\mathbf{Q}_2$ whose solutions are

$$\mathbf{Q}_1 = (\sum_k \mathbf{P}_k \mathbf{P}_k^\mathsf{T})^{-1} (\sum_k w_k x_k \mathbf{P}_k), \tag{6}$$

$$\mathbf{Q}_2 = (\sum_k \mathbf{P}_k \mathbf{P}_k^\mathsf{T})^{-1} (\sum_k w_k y_k \mathbf{P}_k). \tag{7}$$

The matrix $L = (\sum_k \mathbf{P}_k \mathbf{P}_k^\mathsf{T})$ 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 object feature points P_k, and therefore must be corrected using the value of w specific to the coordinates of P_k:

$$w_k = \mathbf{R}_3 \cdot \widetilde{\mathbf{P}}_k / T_z + 1. \tag{8}$$

Therefore for each image point p_j and each object 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

$$\boldsymbol{cp'_k} = \begin{bmatrix} \mathbf{Q}_1 \cdot \mathbf{P}_k \\ \mathbf{Q}_2 \cdot \mathbf{P}_k \end{bmatrix}. \tag{10}$$

These squared distances are

$$d_{jk}^2 = \left|\boldsymbol{p'_k p''_k}\right|^2 = (\mathbf{Q}_1 \cdot \mathbf{P}_k - w_k x_j)^2 + (\mathbf{Q}_2 \cdot \mathbf{P}_k - w_k y_j)^2, \tag{11}$$

where x_j and y_j are the image coordinates of the image point p_j, $\mathbf{P}_k$ is the vector $(\widetilde{\mathbf{P}}_k, 1)$, and $\mathbf{Q}_1$ and $\mathbf{Q}_2$ 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

$$\begin{aligned} E &= \textstyle\sum_{j=1}^{N} \sum_{k=1}^{M} m_{jk}(d_{jk}^2 - \alpha) \\ &= \textstyle\sum_{j=1}^{N} \sum_{k=1}^{M} m_{jk}((\mathbf{Q}_1 \cdot \mathbf{P}_k - w_k x_j)^2 + (\mathbf{Q}_2 \cdot \mathbf{P}_k - w_k y_j)^2 - \alpha) \end{aligned} \tag{12}$$

where the m_{jk} are weights, equal to 0 or 1, for each of the squared distances d_{jk}^2, and where *M* and *N* are the number of object and image points, respectively. The m_{jk} are correspondence variables that define the assignments between image and object feature points; these must satisfy a number of correspondence constraints as discussed below. The α term encourages the match of p_j to P_k when $d_{jk}^2 < \alpha$ (provided the correspondence constraints are satisfied), and it penalizes this match when $d_{jk}^2 > \alpha$. This moves the minimum away from the trivial solution $m_{jk} = 0$ for all j and k. Note that when all the assignments are well-defined, i.e., m_{jk} are equal to 0 or 1, and when $\alpha = 0$, this objective function becomes equivalent to that 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 $\mathbf{Q}_1$ and $\mathbf{Q}_2$ assuming everything else is fixed (see below).

3. Compute the correction terms w_k using the pose vectors $\mathbf{Q}_1$ and $\mathbf{Q}_2$ just computed (as described in the previous section).

This iterative approach is related to the general expectation-maximization (EM) algorithm [Moon 1996]. 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 the problem of finding the optimal poses $\mathbf{Q}_1$ and $\mathbf{Q}_2$, 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 $\mathbf{Q}_1$ and $\mathbf{Q}_2$ whose solutions are

$$\mathbf{Q}_1 = (\sum_{k=1}^{M} m'_k \mathbf{P}_k \mathbf{P}_k^{\mathsf{T}})^{-1} (\sum_{j=1}^{N} \sum_{k=1}^{M} m_{jk} w_k x_j \mathbf{P}_k), \tag{13}$$

$$\mathbf{Q}_2 = (\sum_{k=1}^{M} m'_k \mathbf{P}_k \mathbf{P}_k^{\mathsf{T}})^{-1} (\sum_{j=1}^{N} \sum_{k=1}^{M} m_{jk} w_k y_j \mathbf{P}_k), \tag{14}$$

with $m'_k = \sum_{j=1}^{N} m_{jk}$. The terms $\mathbf{P}_k \mathbf{P}_k^{\mathsf{T}}$ are 4×4 matrices. Therefore computing $\mathbf{Q}_1$ and $\mathbf{Q}_2$ requires the inversion of a single 4×4 matrix, $L = (\sum_{k=1}^{M} m'_k \mathbf{P}_k \mathbf{P}_k^{\mathsf{T}})$, a fairly inexpensive operation (note that because the term in column k and slack row $N + 1$ (see below) is generally greater than 0, $m'_k = \sum_{j=1}^{N} m_{jk}$ is generally not equal to 1, and L generally cannot be precomputed).

4.2 Correspondence Problem

We next find the optimal values of 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* (or *match*) *matrix*, $m = \{m_{jk}\}$, that explicitly specifies the matchings between a set of N image points and a set of M object points, and that minimizes the objective function E. m has one row for each of the N image points p_j and one column for each of the M object points P_k. The assignment matrix must satisfy the constraint that each image point match at most one object point, and vice versa. By adding an extra row and column to m, *slack row* $N+1$ and *slack column* $M+1$, these constraints can be expresses as $m_{jk} \in \{0,1\}$ for $1 \leq j \leq N+1$ and $1 \leq k \leq M+1$, $\sum_{i=1}^{M+1} m_{ji} = 1$ for $1 \leq j \leq N$, and $\sum_{i=1}^{N+1} m_{ik} = 1$ for $1 \leq k \leq M$. A value of 1 in the slack column $M+1$ at row j indicates that image point p_j has not found any match among the object points. A value of 1 in the slack row $N+1$ at column k indicates that the object 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 matches image and object points with the smallest distances d_{jk}^2. This problem can be solved by the iterative softassign technique [Gold 1996, Gold 1998]. The iteration for the assignment matrix m begins with a matrix $m^0 = \{m_{jk}^0\}$ 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. See [Gold 1998] for an analytical justification. The exponentiation has the effect of ensuring that all elements of the assignment matrix are positive. The parameter α determines how far apart two points must be before they are considered unmatchable. It should be set to the maximum allowed squared distance between an image point and the matching projected object point. This should be a function of the noise level in the image. With normally distributed *x* and *y* noise of zero mean and standard deviation σ, the squared distance between a true 2D point and the measured 2D point has a χ^2-squared distribution with 2 degrees of freedom [Hartley 2000, p. 549]. Thus, to ensure with probability 0.99 that a measured point is allowed to match to a true point, we should take $\alpha = 9.21 \times \sigma^2$.

The continuous matrix m^0 converges toward the discrete assignment matrix m due to two mechanisms that are used concurrently:

1. First, a technique due to Sinkhorn [Sinkhorn 1964] 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^0_{jk} corresponding to the smallest d^2_{jk} tend to converge to 1, while the other terms tend to converge to 0. This is a deterministic annealing process [Geiger 1991] known as *softmax* [Bridle 1990]. 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 [Gold 1996, Gold 1998]. The matrix m resulting from an iteration loop that comprises these two substeps is the assignment that minimizes the global objective function $E = \sum_{j=1}^{J} \sum_{k=1}^{K} m_{jk}(d^2_{jk} - \alpha)$. As the pseudocode in Algorithm 1 shows, these two substeps are interleaved in the iteration loop of SoftPOSIT, along with the substeps that find the optimal pose and correct the image points by scaled orthographic distortions.

At the end of the SoftPOSIT iteration, the matrix m will be very close to a true zero-one assignment matrix. If desired, one can obtain discrete correspondences from this matrix and then apply any algorithm that computes pose from known correspondences to obtain the most accurate pose possible.

The SoftPOSIT algorithm has a number of advantages over conventional nonlinear optimization algorithms. Typical nonlinear constrained optimization problems are defined by the minimization of an objective function on a feasible region that is defined by equality and inequality constraints. The simultaneous pose and correspondence problem requires the minimization of an objective function subject to the constraint that the final assignment matrix must be a zero-one matrix whose rows and columns each sum to one. A constraint such as this would be impossible to express using equality and inequality constraints. SoftPOSIT uses deterministic annealing to convert this discrete problem into a continuous

Algorithm 1 SoftPOSIT pseudocode.

1. **Inputs:**
   (a) List of M object points, $\mathbf{P}_k = (X_k, Y_k, Z_k, 1)^\mathsf{T} = (\widetilde{\mathbf{P}}_k, 1), 1 \le k \le M$,
   (b) List of N image points, $p_j = (x_j, y_j), 1 \le j \le N$.
2. **Initialize:**
   (a) Slack elements of assignment matrix m to $\gamma = 1/(\max\{M, N\} + 1)$,
   (b) β to β_0 ($\beta_0 \approx 0.0004$ if nothing is known about the pose, and is larger if an initial pose can be guessed),
   (c) Pose vectors $\mathbf{Q}_1$ and $\mathbf{Q}_2$ using the expected pose or a random pose within the expected range,
   (d) $w_k = 1, 1 \le k \le M$.
3. **Do A until** $\beta > \beta_{final}$ ($\beta_{final} \approx 0.5$) (*Deterministic annealing loop*)
   (a) Compute squared distances $d_{jk}^2 = (\mathbf{Q}_1 \cdot \mathbf{P}_k - w_k x_j)^2 + (\mathbf{Q}_2 \cdot \mathbf{P}_k - w_k y_j)^2, 1 \le j \le N, 1 \le k \le M$.
   (b) Compute $m_{jk}^0 = \gamma \exp(-\beta(d_{jk}^2 - \alpha)), 1 \le j \le N, 1 \le k \le M$.
   (c) **Do B until** $\left\| m^i - m^{i-1} \right\|$ **small** (*Sinkhorn's method*)
      i. Normalize nonslack rows of m: $m_{jk}^{i+1} = m_{jk}^i / \sum_{k=1}^{M+1} m_{jk}^i,\ 1 \le j \le N, 1 \le k \le M+1$.
      ii. Normalize nonslack columns of m: $m_{jk}^{i+1} = m_{jk}^{i+1} / \sum_{j=1}^{N+1} m_{jk}^{i+1},\ 1 \le j \le N+1, 1 \le k \le M$.
   (d) **End Do B**
4. Compute the 4×4 matrix $L = (\sum_{k=1}^{M} m'_k \mathbf{P}_k \mathbf{P}_k^\mathsf{T})$ with $m'_k = \sum_{j=1}^{N} m_{jk}$.
5. Compute L^{-1}.
6. Compute $\mathbf{Q}_1 = (Q_1^1, Q_1^2, Q_1^3, Q_1^4)^\mathsf{T} = L^{-1}(\sum_{j=1}^{N} \sum_{k=1}^{M} m_{jk} w_k\, x_j \mathbf{P}_k)$.
7. Compute $\mathbf{Q}_2 = (Q_2^1, Q_2^2, Q_2^3, Q_2^4)^\mathsf{T} = L^{-1}(\sum_{j=1}^{N} \sum_{k=1}^{M} m_{jk} w_k\, y_j \mathbf{P}_k)$.
8. Compute $s = \left(\left\|(Q_1^1, Q_1^2, Q_1^3)\right\| \left\|(Q_2^1, Q_2^2, Q_2^3)\right\|\right)^{1/2}$.
9. Compute $\mathbf{R}_1 = (Q_1^1, Q_1^2, Q_1^3)^\mathsf{T}/s,\ \mathbf{R}_2 = (Q_2^1, Q_2^2, Q_2^3)^\mathsf{T}/s,\ \mathbf{R}_3 = \mathbf{R}_1 \times \mathbf{R}_2$.
10. Compute $T_x = Q_1^4/s,\ T_y = Q_2^4/s,\ T_z = f/s$.
11. Compute $w_k = \mathbf{R}_3 \cdot \widetilde{\mathbf{P}}_k / T_z + 1, 1 \le k \le M$.
12. $\beta = \beta_{update}\beta$ ($\beta_{update} \approx 1.05$) .
13. **End Do A**
14. **Outputs:**
   (a) Rotation matrix $R = [\mathbf{R}_1\ \mathbf{R}_2\ \mathbf{R}_3]^\mathsf{T}$,
   (b) Translation vector $\mathbf{T} = (T_x, T_y, T_z)$,
   (c) Assignment matrix $m = \{m_{jk}\}$ between the list of image points and the list of object points.

one that is indexed by the control parameter β. This has two advantages. First, it allows solutions to the simpler continuous problem to slowly transform into a solution to the discrete problem. Secondly, many local minima are avoided by minimizing an objective function that is highly smoothed during the early phases of the optimization but which gradually transforms into the original objective function and constraints at the end of the optimization.

5 Random Start SoftPOSIT

The SoftPOSIT algorithm performs a search starting from an initial guess for the object's pose. The global objective function that this search attempts to minimize (Eq. (12)) has many local optima. The deterministic annealing process initially smooths this objective function, which eliminates shallow local optima and greatly improves SoftPOSIT's chances of finding the global optimum if it is near the initial guess. However, one cannot expect to smooth the objective function to the extent that it has a single local optimum at the same location as the global optimum of the unsmoothed objective function: too much smoothing can hide the global optimum and lead the search away from this optimum just as quickly as no smoothing at all. Thus, the search performed by SoftPOSIT is local, and there is no guarantee of finding the global optimum given a single initial guess.

Given an initial pose that lies in a valley of the smoothed objective function, 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. The size and shape of these valleys depends on a number of factors including the parameters of the annealing schedule (β_0 and β_{update}), the complexity of the 3D object, the amount of object occlusion, the amount of image clutter, and the image measurement noise. A common method of searching for a global optimum, and the one used here, is to run the local search algorithm starting from a number of different initial guesses, and keep the first solution that meets a specified termination criterion. Our initial guesses span the range $[-\pi, \pi]$ for the three Euler rotation angles, and a 3D space of translations known to contain the true translation. We use a pseudo-random number generator to generate random 6-vectors in a unit 6D hypercube. (Using a quasi-random [Morokoff 1994] coverage of the hypercube did not improve the performance of the algorithm.) These points are then scaled to cover the expected ranges of translation and rotation. The rest of this section describes the search termination criterion that we use.

5.1 Search Termination

Ideally, one would like to repeat the search from a new starting point whenever the number of object-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 object points that match image points is less than some threshold t_m. Due to occlusion and imperfect image feature extraction algorithms, not all object points will be detected as features in an image of that object. Let the fraction of detected object features be

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

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 object features.

We terminate the search for better solutions when the current solution is such that the number of object points that match any image point is greater than or equal to the threshold $t_m = \rho p_d M$, where ρ determines what percent of the detected object points must be matched $(0 < \rho \leq 1)$, and M is the total number of object points, so that $p_d M$ is the number of detected object points. ρ accounts for measurement noise that typically prevents some detected object 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 object points is high. It has been observed, however, that these situations are relatively uncommon.

We note that Grimson and Huttenlocher [Grimson 1991] have derived an expression for a threshold on the number of matched object points necessary to accept a local optimum; their expression is a function of the numbers of image and object points and of the sensor noise, and guarantees with a specified probability that the globally optimal solution has been found.

5.2 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, overlaid on top of it, the projected object 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 iteration or restart. At the i^{th} step of the SoftPOSIT iteration, the match matrix $m^i = \{m^i_{j,k}\}$ is used to predict the final correspondences of object to image points. Upon convergence of SoftPOSIT, one would expect image point j to correspond to object 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 step i as $r_i = n_i/(p_d K)$ where p_d is the fraction of detected object features as defined above.

The early termination test centers around this match ratio measure. This measure is commonly used [Grimson 1991] 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} step of the 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 step 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} step of the 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 step 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 object points 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 .

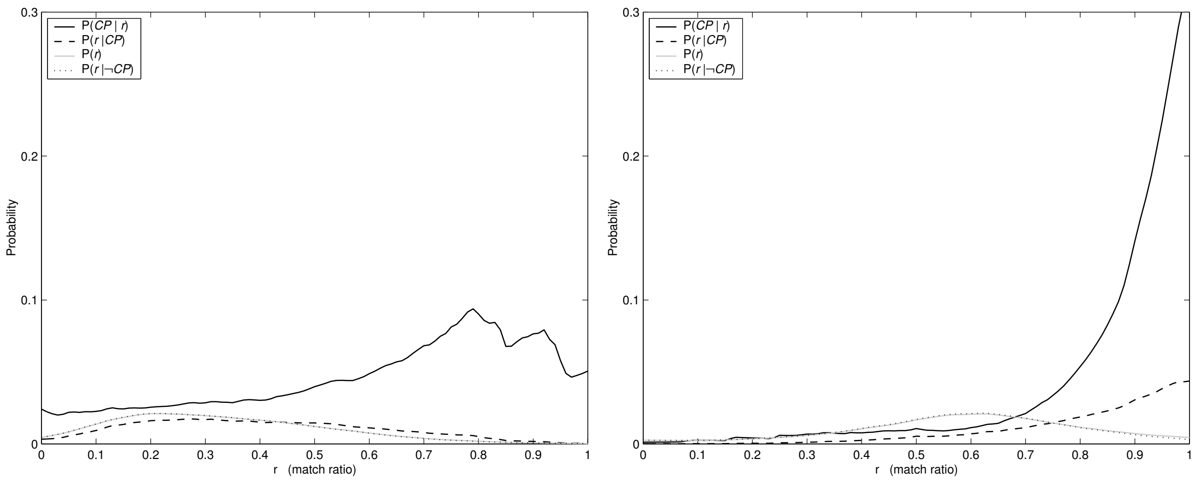


Figure 5: Probability functions estimated for (a) the first iteration, and (b) the 31st iteration, of the SoftPOSIT algorithm.

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, M, 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. M is the number of points (vertices) in a 3D object. p_d is the probability that the image of any particular object point will be detected as a feature point in the image. p_d takes into account occlusion of the 3D object 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 object 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 $n_t = 100$, $M \in \{20, 30, 40, 50, 60, 70, 80\}$, $p_d \in \{0.4, 0.6, 0.8\}$, $p_c \in \{0.2, 0.4, 0.6\}$, and $\sigma \in \{0.5, 1.0, 2.5\}$. (Because corner detection algorithms typically claim accuracies of $1/10^{\text{th}}$ of a pixel [Brand 1994], these values of σ are conservative.) With these parameters, 18,900 independent trials were performed.

For each trial, a 3D object is created in which the M object vertices are randomly located in a sphere centered at the object's origin. Because this algorithm works with points, not with line segments, it is only the object vertices that are important in the current tests. However, to make the images produced by the algorithm easier to understand, we draw each object vertex as connected by an edge to the two closest of the remaining object vertices. These connecting edges are not used by the SoftPOSIT algorithm. The object is then rotated into some arbitrary orientation, and translated to some random point in the field of view of the camera. Next, the object is projected into the image plane of the camera; each projected object 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, $Mp_dp_c/(1-p_c)$ 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 object point, whether or not the point was detected. Figure 6 shows a few examples of cluttered images of random objects that are typical of those used in our experiments.

In our experiments, we consider a pose to be *good* when it allows t_m (defined in Section 5.1) or more of the M object points to be matched to some image point. The number of random starts (random initial pose guesses) for each trial was limited to 10,000. Thus, if a good pose is not found after 10,000 starts, the algorithm gives up. (As discussed below, far fewer starts are typically required for success.) Figures 7 and 8 show a number of examples of poses found by SoftPOSIT when random 6-vectors are used as the initial guesses for pose.

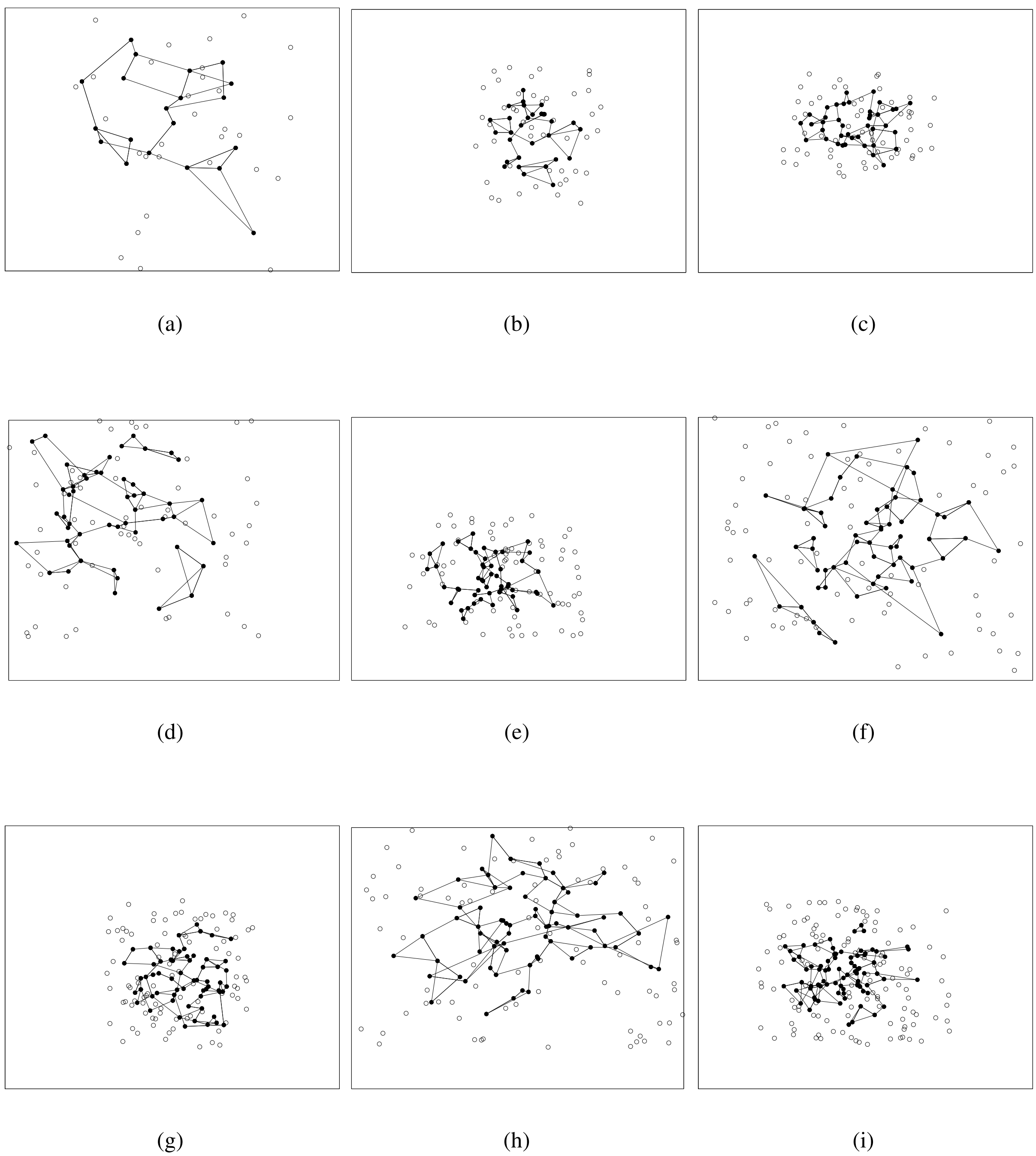
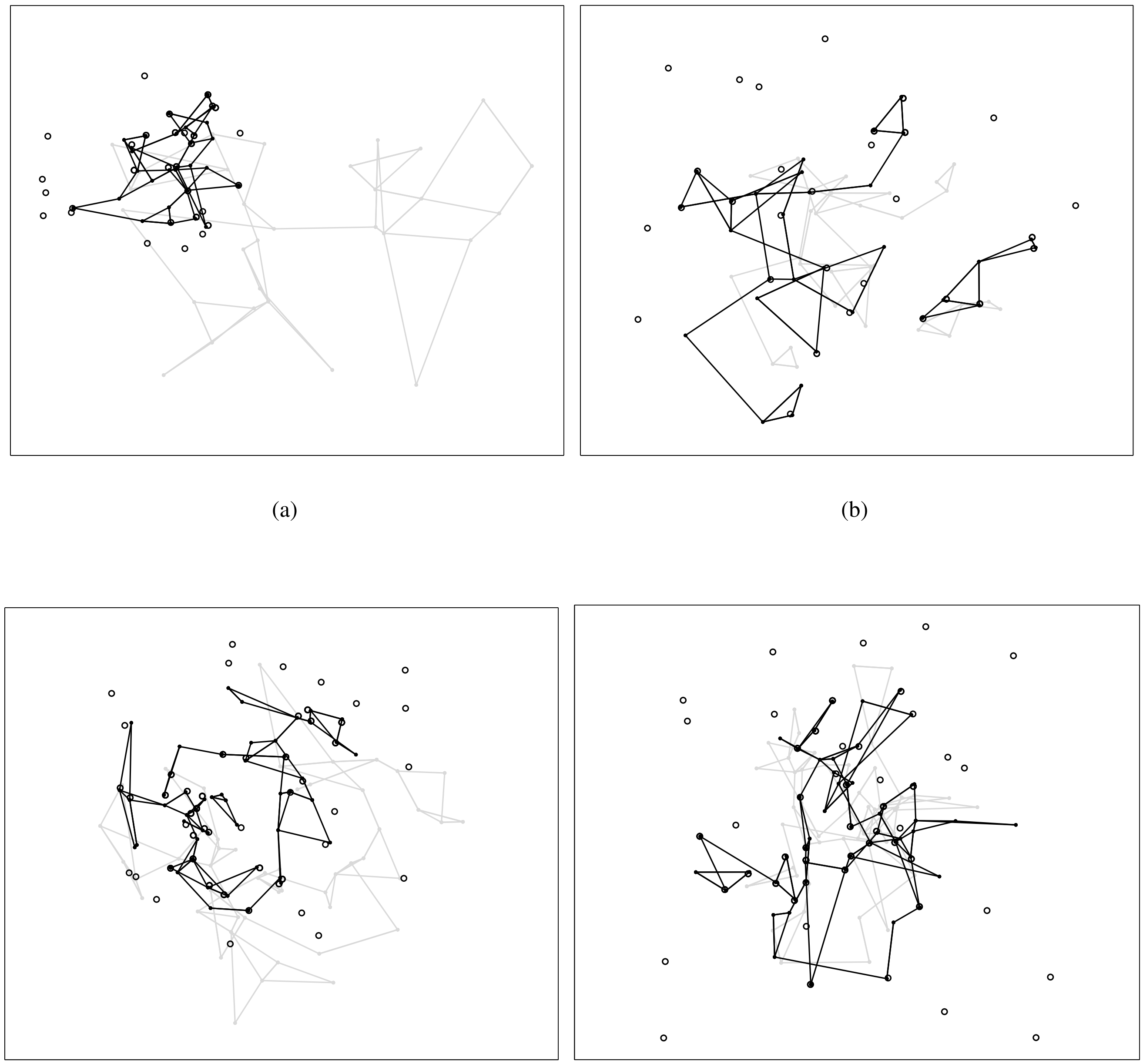


Figure 6: Typical images of randomly generated objects and images. The black points are projected object points and the white points (circles) are clutter points. The black lines, which connect the object points, are included in these pictures to assist the reader in understanding the pictures; they are not used by the algorithm. The number of points in the objects 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.



(a) (b)

(c) (d)

Figure 7: Projected objects and cluttered images for which SoftPOSIT was successful. The small circles are the image points (including projected object and clutter) to which the objects must be matched. The light gray points and lines show the projections of the objects in the initial poses (random guesses) which lead to good poses being found. The black points and lines show the projections of the objects in the good poses that are found. The black points that are not near any circle are occluded object 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 $p_d = 0.6$, $p_c = 0.4$, $\sigma = 2.5$, $M = 30$ for (a) and (b), $M = 50$ for (c) and (d).

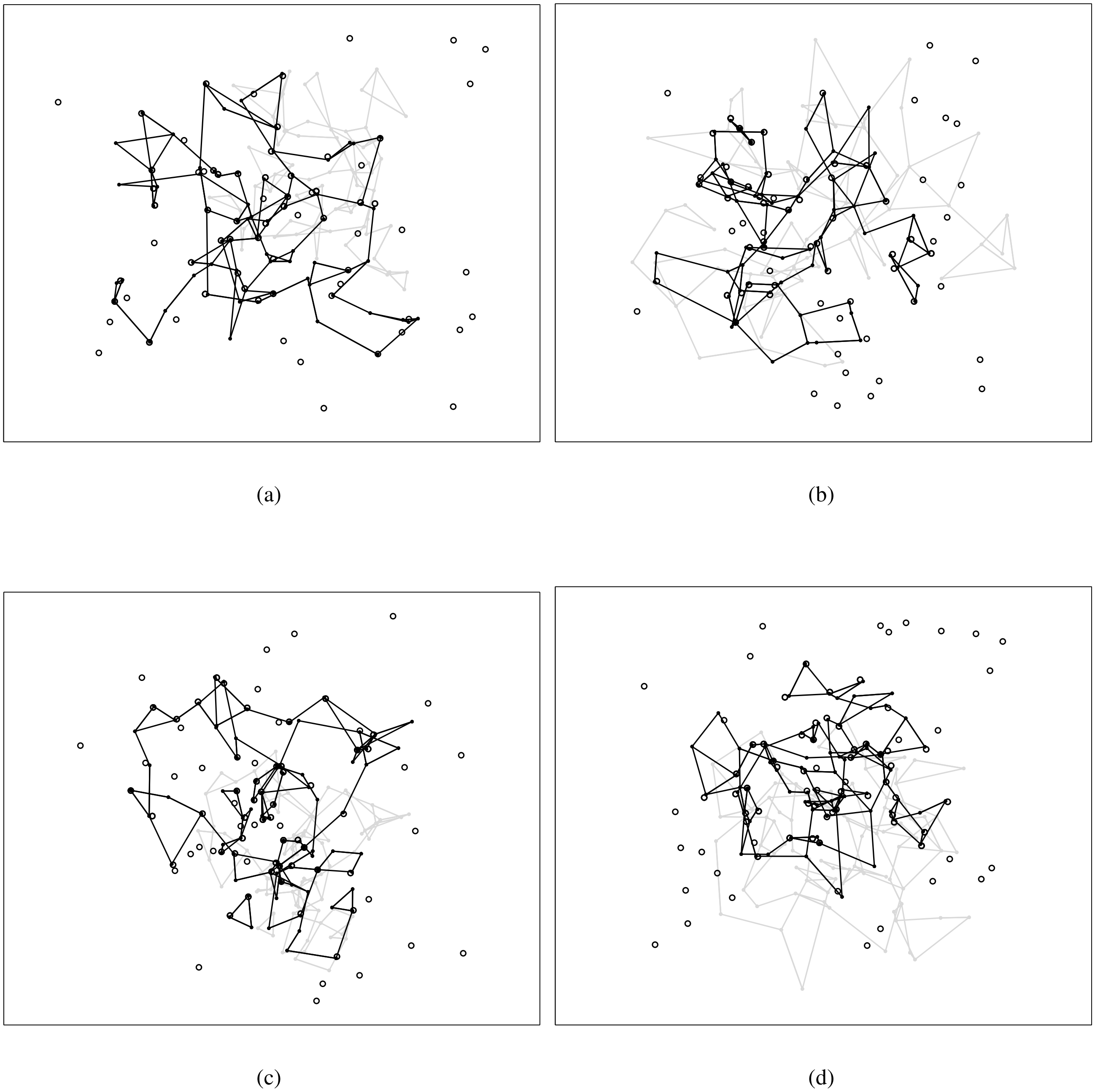


Figure 8: More complex objects and cluttered images for which SoftPOSIT was successful. The Monte Carlo parameters for these tests are $p_d = 0.6$, $p_c = 0.4$, $\sigma = 2.5$ and $M = 70$ for (a) and (b), $M = 80$ for (c) and (d).

Figure 9 shows the success rate of the algorithm (percent of trials for which a good pose was found

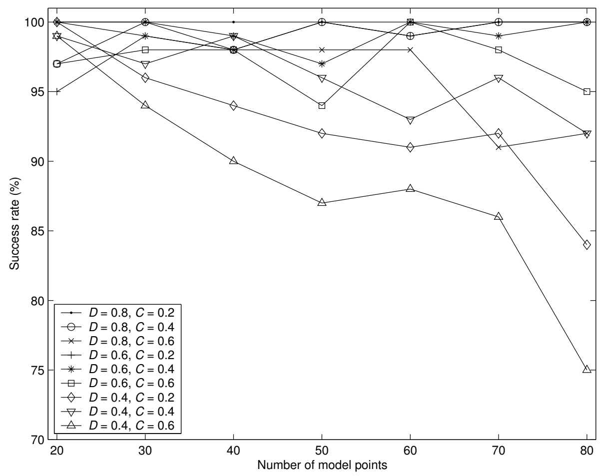
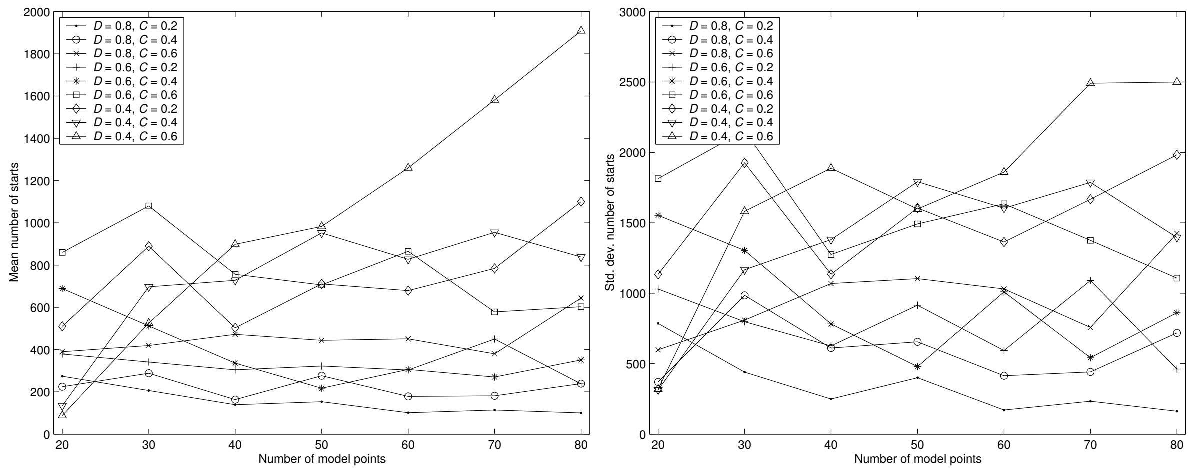


Figure 9: Success rate as a function of the number of object points for fixed values of p_d and p_c. (Note that p_d and p_c are denoted by D and C, respectively, in the legend of this figure and in the next few figures.)

in 10,000 starts, given no knowledge of the correct pose) as a function of the number of object 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 object points decreases. This is due to the algorithm's ability to more easily match a small number of object points to clutter than a large number of object 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 object 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 object points. Also, there does not appear to be any significant increase in the standard deviation of the number of random starts as the



(a) (b)

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

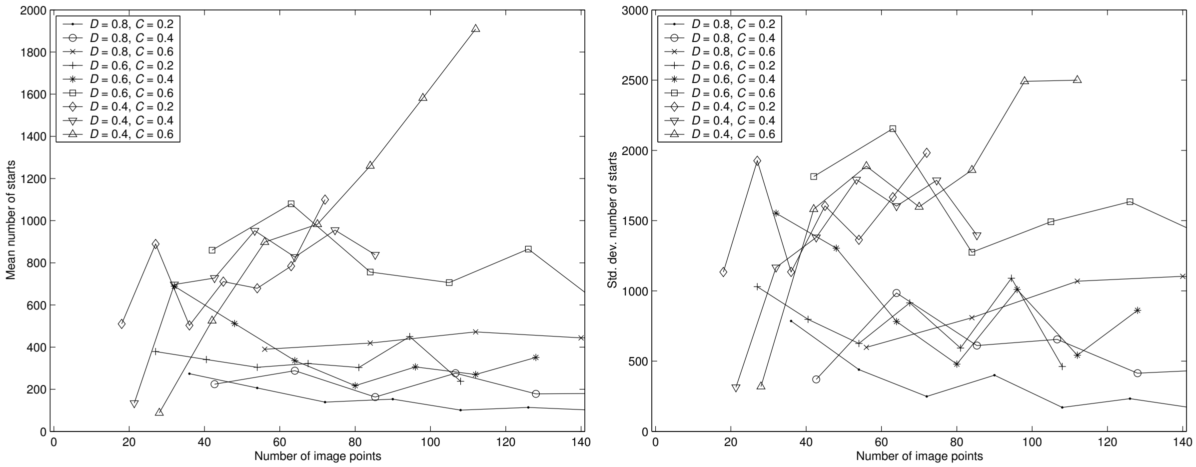
number of object points increases. The mean number of starts over all of the tests is approximately 500; the mean exceeds 1100 starts only in the single hardest case. Figure 11 shows the same data but plotted as a function of the number of image points. Again, except for the two highest occlusion and clutter cases, the mean number of starts is about constant or increases very slowly as the number of image points increases.

6.2 Run Time Comparison

The RANSAC algorithm [Fischler 1981] is the best known algorithm to compute object pose given noncorresponding 3D object and 2D image points. In this section, we compare the expected run time[1] of SoftPOSIT to that of RANSAC for each of the simulated data sets discussed in Section 6.1.

The mean run time of SoftPOSIT on each of these data sets was recorded during the Monte Carlo experiments. As will be seen below, to have run RANSAC on each of these data sets would have required a prohibitive amount of time. This was not necessary, however, since we can accurately estimate the

[1] All algorithms and experiments were implemented in Matlab on a 2.4 GHz Pentium 4 processor running the Linux operating system.



(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.

number of random samples of the data that RANSAC will examine when solving any particular problem. The expected run time of RANSAC is then the product of that number of samples with the expected run time on one sample of that data.

The computational complexity of a pose problem depends on the three parameters M, p_d, and p_c defined in Section 6.1. For each combination of these three parameters, we need to determine the expected run time of RANSAC for a single sample of three object points and three image points[2] from that data. This was accomplished by running RANSAC on many random samples generated using the same set of three complexity parameters. The time per sample for a given problem complexity is estimated as the total time used by RANSAC to process those samples (excluding time for initialization) divided by the number of samples processed.

We now estimate how many samples RANSAC will examine for problems of a particular complexity. In Appendix A, we compute the probability, p, as a function of M, p_d, and p_c, that a random sample of three object points and three image points consists of three correct correspondences. Then, the number

[2]Three correspondences between object and image points is the minimum necessary to constrain the object to a finite number of poses.

of random samples of correspondence triples that must be examined by RANSAC in order to ensure with probability z that at least one correct correspondence triple will be examined is

$$s_1(z, p) = \frac{\log(1 - z)}{\log(1 - p)}.$$

Some implementations of RANSAC will halt as soon as the first good sample is observed, thus reducing the run time of the algorithm. In this case, the expected number of random samples that will be examined in order to observe the first good sample is

$$s_2(p) = \frac{1}{p}.$$

Note that for all values of M, p_d, and p_c that we consider here, and for $z \geq 0.75$ (the smallest observed success rate for SoftPOSIT), $s_2(p) < s_1(z, p)$. A RANSAC algorithm using s_2 will always be faster than one using s_1, but it will not be as robust since robustness increases with the number of samples examined. In the following, the run times of SoftPOSIT and RANSAC are compared using both s_1 and s_2 to determine the number of samples that RANSAC examines.

For a data set with complexity given by M, p_d, and p_c, SoftPOSIT has a given observed success rate which we denote by $z_{\text{softPOSIT}}(M, p_d, p_c)$ (see Figure 9). Since we did not run RANSAC on this data, we can't compare the success rates of SoftPOSIT and RANSAC for a given fixed amount of run time. However, we can compare the run time required by both to achieve the same rate of success on problems of the same complexity by estimating the run time of RANSAC when its required probability of success is $z = z_{\text{softPOSIT}}(M, p_d, p_c)$. These run times are shown in Figures 12 and 13 . From these figures, it can be seen that the RANSAC algorithm requires one to three orders of magnitude more run time than SoftPOSIT for problems with the same level of complexity in order to achieve the same level of success. Furthermore, for the majority of the complexity cases, run time as a function of input size increase at a faster rate for the RANSAC algorithms than for the SoftPOSIT algorithm. The totality of Monte Carlo experiments described in Section 6.1 required about 30 days for SoftPOSIT to complete. From this analysis it can be estimated that a RANSAC algorithm which examines s_1 samples would require about

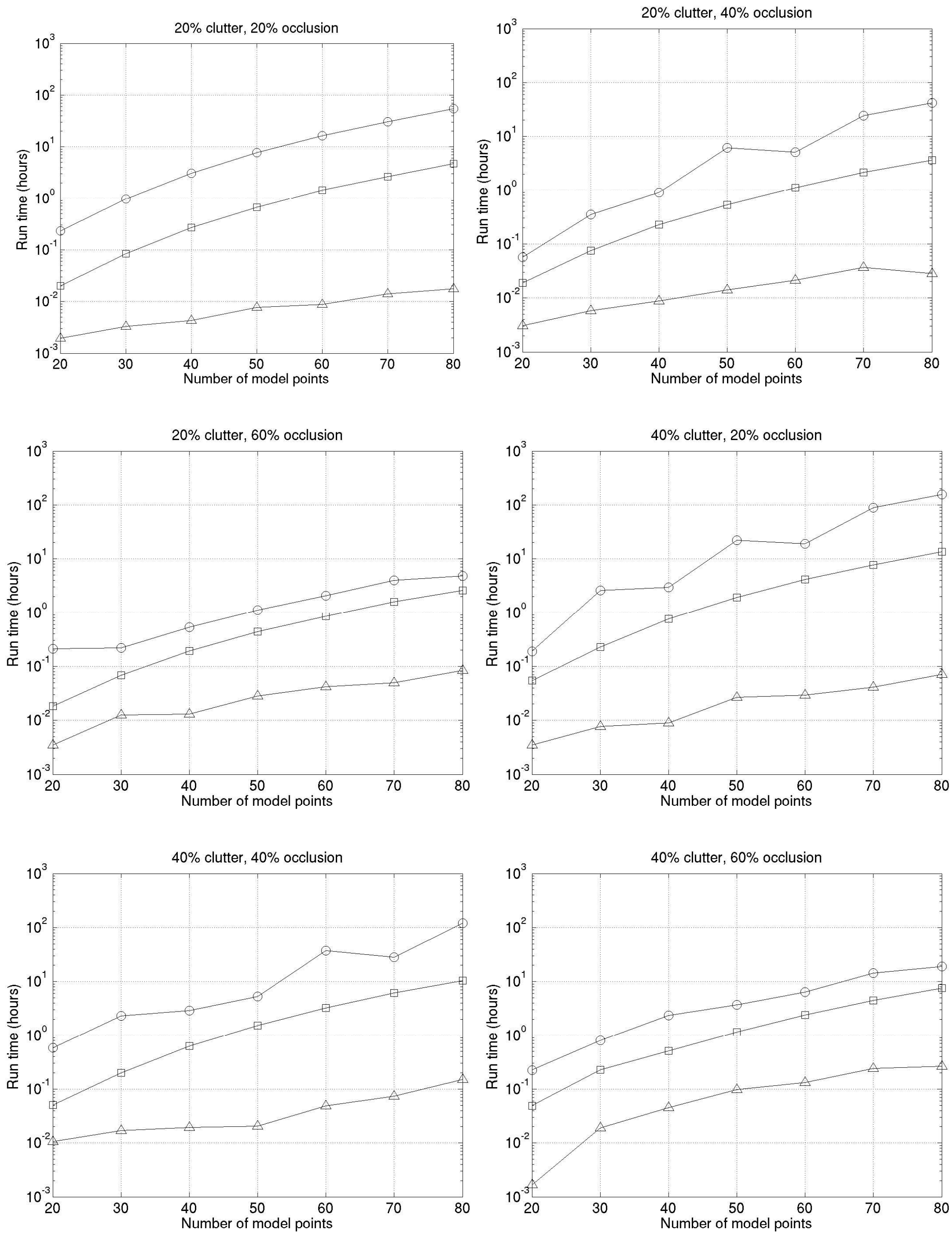


Figure 12: Comparison of the run times of SoftPOSIT to those of RANSAC for problems with 20-40% clutter and 20-60% object occlusion. The SoftPOSIT run times are marked with triangles. The RANSAC run times are marked with circles for the case that the number of samples is determined by s_1 , and with squares for the case that the number of samples is determined by s_2.

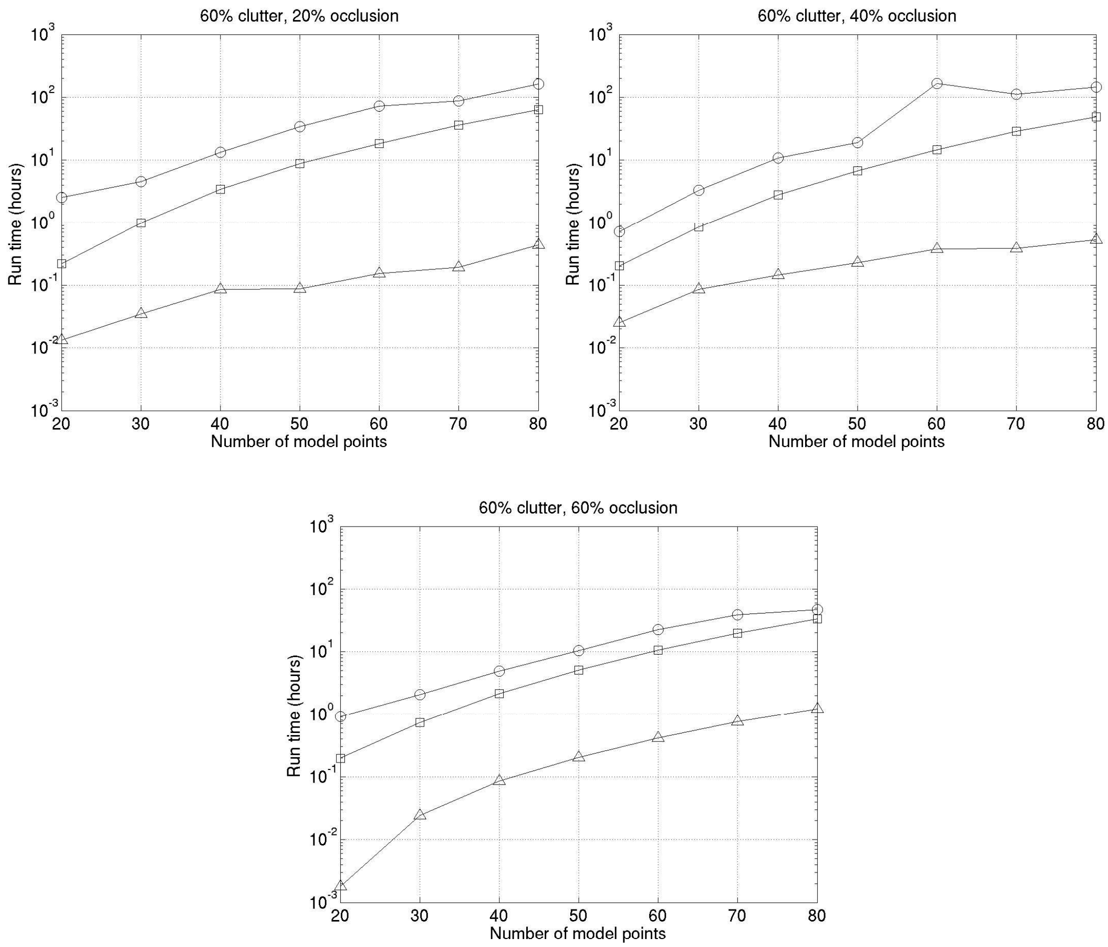


Figure 13: Comparison of the run times of SoftPOSIT to those of RANSAC for problems with 60% clutter and 20-60% object occlusion. The SoftPOSIT run times are marked with triangles. The RANSAC run times are marked with circles for the case that the number of samples is determined by s_1 , and with squares for the case that the number of samples is determined by s_2.

19.4 *years* to complete the same experiments, and a RANSAC algorithm which examines s_2 samples would require about 4.5 years. Clearly, it would not have been practical to run RANSAC on all of these experiments.

6.3 Algorithm Complexity

The run-time complexity of a single invocation of SoftPOSIT is $O(MN)$ where M is the number of object points and N is the number of image points; this is because the number of iterations on all of the loops in the pseudocode in Section **??** are bounded by a constant, and each line inside a loop is computed in time at most $O(MN)$. As shown in Figures 10 and 11, the mean number of random starts (invocations of SoftPOSIT) required to find a good pose in the worst (hardest) case, 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 $O(N)$, assuming that $M < N$, as is normally the case. Then the run-time complexity of SoftPOSIT with random starts is $O(MN^2)$. This is a factor of N better than the complexity of any published algorithm that solves the simultaneous pose and correspondence problem under a full perspective camera model.

6.4 Experiments with Images

6.4.1 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 14 shows an image generated by this system and a object 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 object 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 object

(a)

(b)

Figure 14: Registration of a 3D city model to an image generated by a virtual reality system. Using the initial guess for the model's pose, the 3D model vertices that project near the detected skyline in the image are selected to be matched to image points along this skyline. (a) Original image from the virtual reality system. (b) Selected model lines and points (white) projected into this image using the pose computed by SoftPOSIT.

points that don't project into the estimated field of view. Then the object points that do fall into this estimated field are further culled by keeping only those that project near 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.

6.4.2 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 15 shows a small robot docking onto a larger robot. In order to accomplish this,

Figure 15: A small robot docking onto a larger robot.

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 16 shows the lines and corner points detected in one image of the large robot. In this test there are

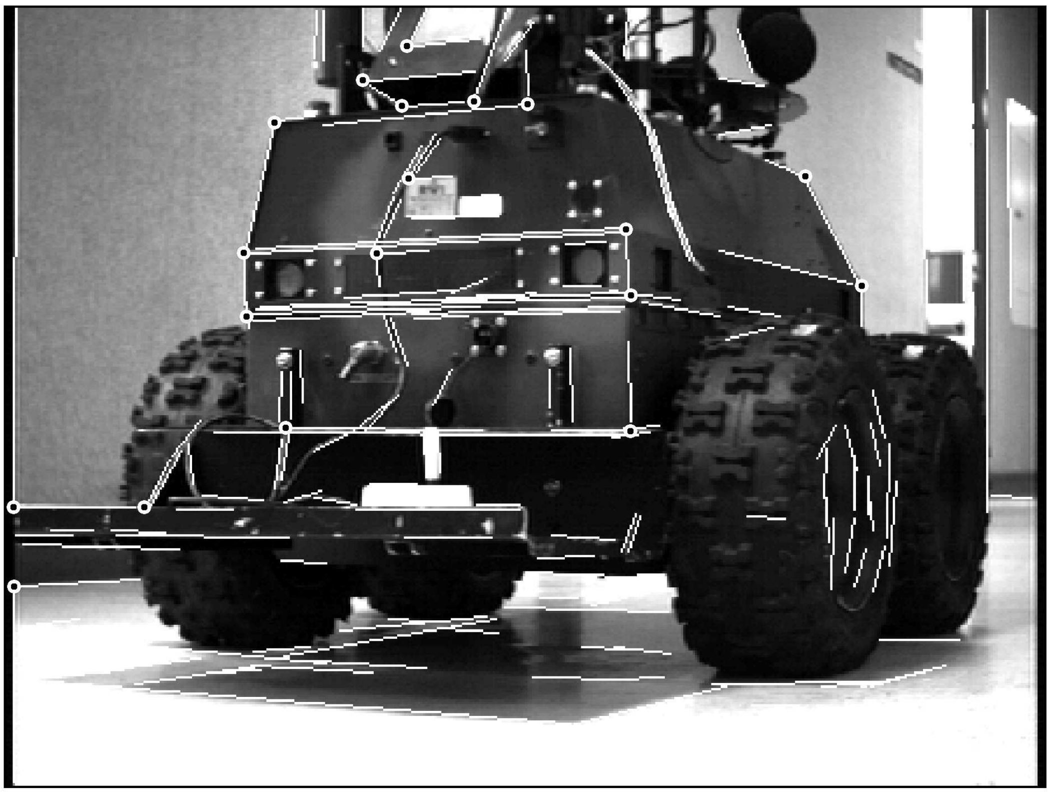
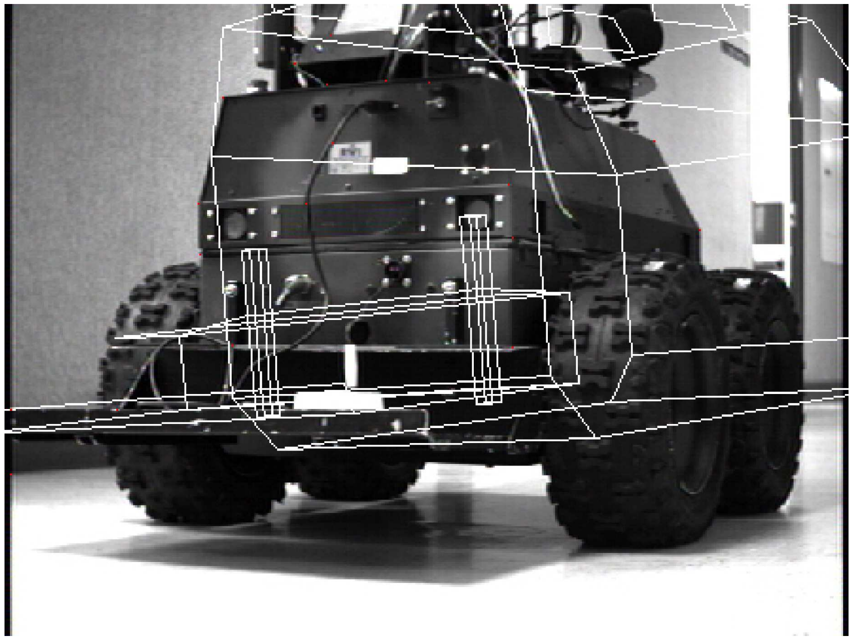
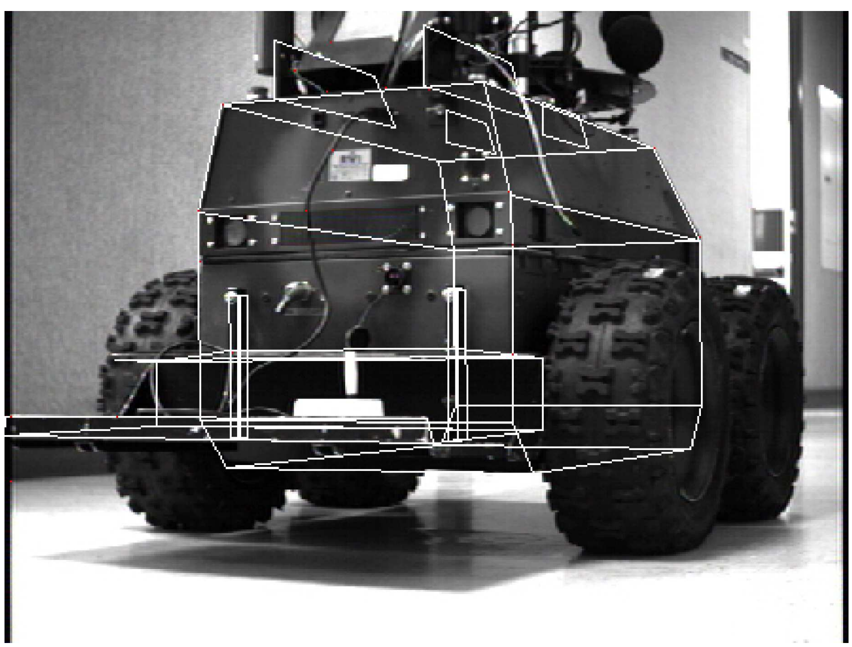


Figure 16: An image of the large robot as seen from the small robot's point of view. Long straight lines detected in the image are shown in white, and their intersections, which ideally should correspond to vertices in the 3D object, are shown in black.

70 points in the object; 89% of these are occluded (or not detected in the image), and 58% of the image points are clutter. Figure 17a shows the initial guess generated by SoftPOSIT which led to the correct pose being found, and Figure 17b shows this correct pose.



(a)



(b)

Figure 17: The initial guess at the robot's pose (a) that leads to the correct pose as shown in (b).

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 softassign algorithm for determining correspondences and the POSIT algorithm for determining pose. This 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 $O(MN^2)$. 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 run time complexity of the general hypothesize-and-test approach to model-to-image registration is derived in this appendix. We first define a few parameters:

M is the number of 3D object points,

N is the number of image points,

p_d is the fraction of object points that are present (non-occluded) in the image,

R is 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 [Fischler 1981] 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 M, N, p_d, and R for the hypothesize-and-test algorithm for large values of M and N.

Because we assume that the hypothesize-and-test algorithm has no *a priori* information about which correspondences are correct, correspondences are formed from randomly chosen object and image points. We assume that three correspondences are used to estimate the object's pose. Let $S = p_d M$ be the number of detected (non-occluded) object points in the image. For a correspondence to be correct, two conditions must be satisfied: the object point must be non-occluded and the image point must correspond to the object 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 the probability that these two conditions are satisfied, which is

$$\frac{S-n+1}{M-n+1} \cdot \frac{1}{N-n+1}.$$

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

$$\frac{S(S-1)(S-2)}{M(M-1)(M-2)N(N-1)(N-2)} \approx \frac{S^3}{M^3 N^3} = \left(\frac{p_d}{N}\right)^3.$$

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

$$\left(1 - \left(\frac{p_d}{N}\right)^3\right)^T.$$

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

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

Solving for T, we get

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

Noting that $(p_d/N)^3$ 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

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

Since each sample requires $O(M \log N)$ time for back-projection and verification (assuming an $\mathcal{O}(\log N)$ nearest neighbor algorithm [Arya 1998] is used to search for image points close to each projected object point), the complexity of the general hypothesize-and-test algorithm is

$$\left(\frac{N}{p_d}\right)^3 \log\left(\frac{1}{1-R}\right) \times \mathcal{O}(M \log N) = \mathcal{O}(MN^3 \log N).$$

Appendix B Scaled Orthographic Image Points

Here we give a geometric interpretation of the relation between perspective and scaled orthographic image points. Consider Figure 4. 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 (Eqs. (1, 2)):

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

with $w = \mathbf{R}_3 \cdot \widetilde{\mathbf{P}}/T_z + 1$. We can see that $\boldsymbol{cp'} = s(\mathbf{R}_1 \cdot \widetilde{\mathbf{P}} + T_x, \mathbf{R}_2 \cdot \widetilde{\mathbf{P}} + T_y)$. 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 produced by 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 $\mathbf{R}_3 \cdot \widetilde{\mathbf{P}} + T_z$, 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.

Acknowledgments

The support of NSF grants EAR-99-05844 and IIS-00-86116 is gratefully acknowledged. We would also like to thank the anonymous reviewers for their assistance in improving this paper.

References

[Arya 1998] Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., and Wu, A. 1998, An Optimal Algorithm for Approximate Nearest Neighbor Searching, Journal of the ACM, 45(6): 891–923.

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

[Beis 1999] Beis, J.S. and Lowe, D.G. 1999. Indexing Without Invariants in 3D Object Recognition. IEEE Trans. Pattern Analysis and Machine Intelligence, 21(10):1000–1015.

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

[Beveridge 1995] Beveridge, J.R. and Riseman, E.M. 1995. Optimal Geometric Model Matching Under Full 3D Perspective. Computer Vision and Image Understanding, 61(3):351–364.

[Brand 1994] Brand, P. and Mohr, R. 1994. Accuracy in Image Measure. In Proc. SPIE, Videometrics III, Boston, MA, pp. 218–228.

[Bridle 1990] Bridle, J.S. 1990. Training Stochastic Model Recognition as Networks can Lead to Maximum Mutual Information Estimation of Parameters. In Proc. Advances in Neural Information Processing Systems, Denver, CO, pp. 211–217.

[Burns 1993] Burns, J.B., Weiss, R.S. and Riseman, E.M. 1993. View Variation of Point-Set and Line-Segment Features. IEEE Trans. Pattern Analysis and Machine Intelligence, 15(1):51–68.

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

[Cass 1992] Cass, T.A. 1992. Polynomial-Time Object Recognition in the Presence of Clutter, Occlusion, and Uncertainty. In Proc. European Conf. on Computer Vision, Santa Margherita Ligure, Italy, pp. 834–842.

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

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

[DeMenthon 1993] DeMenthon, D. and Davis, L.S. 1993. Recognition and Tracking of 3D Objects by 1D Search. In Proc. DARPA Image Understanding Workshop, Washington, DC, pp. 653–659.

[DeMenthon 1995] DeMenthon, D. and Davis, L.S. 1995. Model-Based Object Pose in 25 Lines of Code. International Journal of Computer Vision, 15(1-2):123–141.

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

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

[Fiore 2001] Fiore, P.D. 2001. Efficient Linear Solution of Exterior Orientation. IEEE Trans. on Pattern Analysis and Machine Intelligence, 23(2):140–148.

[Fischler 1981] Fischler, M.A. and Bolles, R.C. 1981. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. Association for Computing Machinery, 24(6):381–395.

[Geiger 1991] Geiger, D. and Yuille, A.L. 1991. A Common Framework for Image Segmentation. International Journal of Computer Vision, 6(3):227–243.

[Gold 1996] Gold, S. and Rangarajan, A. 1996. A Graduated Assignment Algorithm for Graph Matching, IEEE Trans. on Pattern Analysis and Machine Intelligence, 18(4):377–388.

[Gold 1998] Gold, S., Rangarajan, A., Lu, C.-P., Pappu, S. and Mjolsness, E. 1998. New Algorithms for 2D and 3D Point Matching: Pose Estimation and Correspondence. Pattern Recognition, 31(8):1019–1031.

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

[Grimson 1991] Grimson, E. and Huttenlocher, D.P. 1991. On the Verification of Hypothesized Matches in Model-Based Recognition. IEEE Trans. on Pattern Analysis and Machine Intelligence, 13(12):1201–1213.

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

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

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

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

[Jacobs 1992] Jacobs, D.W. 1992. Space Efficient 3-D Model Indexing. In Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Champaign, IL, pp. 439–444.

[Jurie 1999] Jurie, F. 1999. Solution of the Simultaneous Pose and Correspondence Problem Using Gaussian Error Model. Computer Vision and Image Understanding, 73(3):357–373.

[Lamdan 1988] Lamdan, Y. and Wolfson, H.J. 1988. Geometric Hashing: A General and Efficient Model-Based Recognition Scheme. In Proc. IEEE Int. Conf. on Computer Vision, Tampa, FL, pp. 238–249.

[Lu 2000] Lu, C.-P., Hager, G.D. and Mjolsness, E. 2000. Fast and Globally Convergent Pose Estimation from Video Images. IEEE Trans. on Pattern Analysis and Machine Intelligence, 22(6):610–622.

[Moon 1996] Moon, T.K. 1996. The Expectation-Maximization Algorithm. IEEE Signal Processing Magazine, 13(6):47–60.

[Morokoff 1994] Morokoff, W.J. and Caflisch, R.E., 1994. Quasi-Random Sequences and their Discrepancies. SIAM Journal Scientific Computing, 15(6):1251–1279.

[Murase 1995] Murase, H. and Nayar, S.K. 1995. Visual Learning and Recognition of 3-D Objects from Appearance. Int. Journal of Computer Vision, 14(1):5–24.

[Olson 1997] Olson, C.F. 1997. Efficient Pose Clustering Using a Randomized Algorithm. Int. Journal of Computer Vision, 23(2):131–147.

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

[Sinkhorn 1964] Sinkhorn, R. 1964. A Relationship between Arbitrary Positive Matrices and Doubly Stochastic Matrices. Annals Mathematical Statistics, 35(2):876–879.

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

[Wunsch 1996] Wunsch, P. and Hirzinger, G. 1996. Registration of CAD Models to Images by Iterative Inverse Perspective Matching. In Proc. Int. Conf. on Pattern Recognition, vol. 1, Vienna, Austria, pp. 78–83.

[Yuan 1989] Yuan, J.-C. 1989. A General Photogrammetric Method for Determining Object Position and Orientation. IEEE Trans. on Robotics and Automation, 5(2):129-142.