Simultaneous Pose and Correspondence Determination using Line Features *

Philip David[1,2], Daniel DeMenthon[3], Ramani Duraiswami[3], and Hanan Samet[1]

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

Abstract

We present a new robust line matching algorithm for solving the model-to-image registration problem. Given a model consisting of 3D lines and a cluttered perspective image of this model, the algorithm simultaneously estimates the pose of the model and the correspondences of model lines to image lines. The algorithm combines softassign for determining correspondences and POSIT for determining pose. Integrating these algorithms into a deterministic annealing procedure allows the correspondence and pose to evolve from initially uncertain values to a joint local optimum. This research extends to line features the SoftPOSIT algorithm proposed recently for point features. Lines detected in images are typically more stable than points and are less likely to be produced by clutter and noise, especially in man-made environments. Experiments on synthetic and real imagery with high levels of clutter, occlusion, and noise demonstrate the robustness of the algorithm.

1. Introduction

This paper presents an algorithm for solving the *model-to-image registration* problem using *line features*. This 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 features and a single 2D image of these features. We assume that no additional information is available with which to constrain the pose of the object or to constrain the correspondence of model to image features. This is also known as the *simultaneous pose and correspondence problem.*

Automatic registration of 3D models to images is a fundamental and open problem in computer vision. 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* 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 [5].

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 according to the model features that project sufficiently close to an image feature.

The classic approach to solving these coupled problems is the hypothesize-and-test approach. 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 [6] for the case that no information is available to constrain the correspondences of model to image points.

Many investigators approximate the nonlinear perspective projection via linear affine approximations. This is accurate when the relative depth of object features is small compared to the distance of the object from the camera. Among the researchers that have addressed the full perspective problem, Wunsch and Hirzinger [11] 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. The method of Beveridge and Riseman [1] uses a random-start local search with a hybrid pose estimation algorithm em-

*Partial support of NSF awards 0086162, 9905844, and 9987944 is gratefully acknowledged.

ploying both full-perspective and weak-perspective camera models.

David et al. [4] recently proposed the *SoftPOSIT* algorithm for simultaneous pose and correspondence determination for the case of a 3D point model and its perspective image. This algorithm integrates an iterative pose technique called POSIT (Pose from Orthography and Scaling with ITerations) [5], and an iterative correspondence assignment technique called *softassign* [9] 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.

We extend the SoftPOSIT algorithm from matching point features to the case of matching line features: 3D model lines are matched to image lines in 2D perspective images. Lines detected in images are typically more stable than points and are less likely to be produced by clutter and noise, especially in man-made environments. Also, line features are more robust to partial occlusion of the model. Our current algorithm uses the SoftPOSIT algorithm for points to determine the pose and correspondences for a set of image and model lines. An iteration is performed where at each step the given 2D to 3D line correspondence problem is mapped to a new 2D to 3D point correspondence problem which depends on the current estimate of the camera pose. SoftPOSIT is then applied to improve the estimate of the camera pose. This process is repeated until the pose and correspondences converge.

In the following sections, we examine each step of the method. We first review the SoftPOSIT algorithm for computing pose from noncorresponding 2D image and 3D model points. We then describe how this is used to solve for pose when only line correspondences are available. Finally, some experiments with simulated and real images are shown.

2. Camera Models

Let $\mathbf{P}$ be a 3D point in a world coordinate frame with origin $\mathbf{O}$ (figure 1). If a camera placed in this world frame is used to view $\mathbf{P}$, then the coordinates of this point in the camera frame may be written as $R\mathbf{P} + \mathbf{T}$. Here, R is a 3×3 rotation matrix representing the orientation of the camera frame with respect to the world frame, and the translation $\mathbf{T}$ is the vector from the camera center $\mathbf{C}$ to $\mathbf{O}$, expressed in the camera frame. Let the i^{th} row of R be denoted by $\mathbf{R}_i$ and let the translation be $\mathbf{T} = (T_x, T_y, T_z)^{\mathsf{T}}$.

We assume that the camera is calibrated, so that pixel coordinates can be replaced by normalized image coordinates.

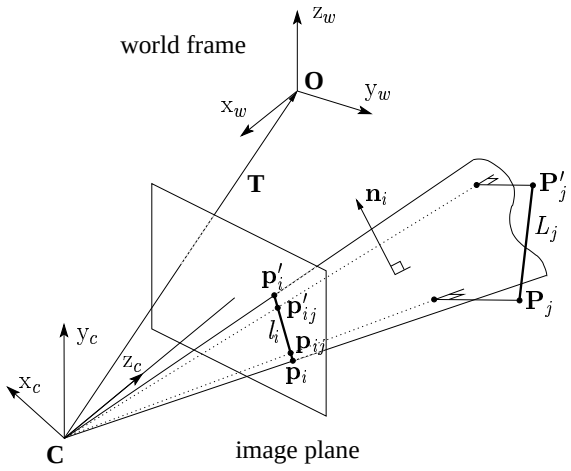

Figure 1: The geometry of line correspondences.

Then, the perspective image of a 3D point $\mathbf{P}$ in the world frame is (x, y) where

$$x = \frac{\mathbf{R}_1\mathbf{P}+T_x}{\mathbf{R}_3\mathbf{P}+T_z}, \quad y = \frac{\mathbf{R}_2\mathbf{P}+T_y}{\mathbf{R}_3\mathbf{P}+T_z}. \tag{1}$$

We will also need to use the *weak perspective* (also known as *scaled orthographic*) projection model, which makes the assumption that the depth of an object is small compared to the distance of the object from the camera, and that visible scene points are close to the optical axis. The weak perspective model will be used iteratively in the process of computing the full perspective pose. Under the weak perspective assumption, $\mathbf{R}_3 \cdot \mathbf{P} \approx 0$ since $\mathbf{R}_3$ is a unit vector in the world coordinate frame that is parallel to the camera's optic axis. The weak perspective image of a 3D point $\mathbf{P}$ in the world frame is (x^w, y^w) where

$$x^w = \frac{\mathbf{R}_1\mathbf{P}+T_x}{T_z}, \quad y^w = \frac{\mathbf{R}_2\mathbf{P}+T_y}{T_z}. \tag{2}$$

3. Pose from Point Correspondences

Our new line matching algorithm builds on the SoftPOSIT algorithm [4], which itself builds on the POSIT algorithm [5]. This section of the paper gives an overview of these two algorithms.

3.1. The POSIT Algorithm

The POSIT algorithm [5] computes an object's pose given a set of corresponding 2D image and 3D object points. The perspective image (x_i, y_i) of a 3D world point $\mathbf{P}_i$ is related

to the image (x_i^w, y_i^w) produced by a scaled orthographic camera according to

$$\begin{aligned} x_i^w &= w_i x_i, \\ y_i^w &= w_i y_i. \end{aligned} \tag{3}$$

Equation (3) is obtained by combining equations (1) and (2). The term w_i can be determined only if the camera pose is known:

$$w_i = \mathbf{R}_3 \cdot \mathbf{OP}_i / T_z + 1, \tag{4}$$

where $\mathbf{OP}_i$ is the vector in the camera coordinate frame from the world origin to $\mathbf{P}_i$. When the depth range of the object along the optical axis is small compared to the object distance, $\mathbf{R}_3 \cdot \mathbf{OP}_i$ is small compared to T_z, and $w_i \approx 1$. This is exactly the assumption made when a perspective camera is approximated by a scaled orthographic camera.

The POSIT algorithm starts by assuming that the perspective image points are identical to the scaled orthographic image points, so that $w_i = 1$ for all i. Under this assumption, the camera pose can be determined by solving a simple linear system of equations. This solution is only approximate since $w_i = 1$ is only approximate. However, given a more accurate estimate of the object's pose, the accuracy of the w_i terms can be improved by reestimating these terms using equation (4). This process is repeated until the pose converges.

3.2. The SoftPOSIT Algorithm

The SoftPOSIT algorithm [4] computes camera pose given a set of 2D image and 3D object points, where the correspondences between these two sets are not know a priori. The SoftPOSIT algorithm builds on the POSIT algorithm by integrating the *softassign* correspondence assignment algorithm [8, 9]. For M image points and N object points, the correspondences between the two sets is given by an $(M+1) \times (N+1)$ assignment matrix m where $0 \leq m_{ij} \leq 1$. Intuitively, the value of m_{ij} ($1 \leq i \leq M$,$1 \leq j \leq N$) specifies how well the i^{th} image point matches to the j^{th} object point. Initially, all m_{ij} have approximately the same value, indicating that correspondences are not known. Row $M+1$ and column $N+1$ of m are the *slack* row and column, respectively. The slack positions in m receive large values when an image point does not match any model point or a model point does not match any image point.

An object's pose can be parameterized by the two vectors $\mathbf{Q}_1 = (\mathbf{R}_1, T_x)/T_z$ and $\mathbf{Q}_2 = (\mathbf{R}_2, T_y)/T_z$; given $\mathbf{Q}_1$ and $\mathbf{Q}_2$, R and T can easily be determined. The homogeneous object points are written as $\mathbf{S}_j = (\mathbf{OP}_j, 1)$. Then, given the assignment weights m_{ij} between the image and object points, the error function

$$E = \sum_{i=1}^{M} \sum_{j=1}^{N} m_{ij} ((\mathbf{Q}_1 \cdot \mathbf{S}_j - w_j x_i)^2 + (\mathbf{Q}_2 \cdot \mathbf{S}_j - w_j y_i)^2) \tag{5}$$

gives the sum of the squared distances between scaled orthographic image points (approximated using the perspective image points as in equation (3)) and the corresponding (weighted by the m_{ij}) scaled orthographic images of the 3D object points (which depend on the object's estimated pose, $\mathbf{Q}_1$ and $\mathbf{Q}_2$) [4]. The solution to the simultaneous pose and correspondence problem consists of the m_{ij}, $\mathbf{Q}_1$, and $\mathbf{Q}_2$ which minimize E. The function E is minimized iteratively as follows:

1. Compute the correspondence variables m_{ij} assuming that pose is fixed.
2. Compute the pose vectors $\mathbf{Q}_1$ and $\mathbf{Q}_2$ assuming that correspondences are fixed.
3. Compute the scaled orthographic corrections w_i using equation (4) and the new pose vectors.

Steps (1) and (2) are described in more detail below.

Computing Pose. By assuming that the m_{ij} are fixed, the object pose which minimizes this error function is found by solving $\partial E / \partial \mathbf{Q}_1 = 0$ and $\partial E / \mathbf{Q}_2 = 0$. The solution is

$$\mathbf{Q}_1 = (\sum_{j=1}^{N} m'_j \mathbf{S}_j \mathbf{S}_j^{\mathsf{T}})^{-1} (\sum_{i=1}^{M} \sum_{j=1}^{N} m_{ij} w_j x_i \mathbf{S}_j), \tag{6}$$

$$\mathbf{Q}_2 = (\sum_{j=1}^{N} m'_j \mathbf{S}_j \mathbf{S}_j^{\mathsf{T}})^{-1} (\sum_{i=1}^{M} \sum_{j=1}^{N} m_{ij} w_j y_i \mathbf{S}_j), \tag{7}$$

where $m'_j = \sum_{i=1}^{M} m_{ij}$ [4]. Computing $\mathbf{Q}_1$ and $\mathbf{Q}_2$ requires the inversion of the 4×4 matrix $\sum_{j=1}^{N} m'_j \mathbf{S}_j \mathbf{S}_j^{\mathsf{T}}$, which is inexpensive.

Computing Correspondences. The correspondence variables m_{ij} are optimized assuming the pose of the object is known and fixed. The goal is to find a zero-one assignment matrix, $m = \{m_{ij}\}$, that explicitly specifies the matches between a set of M image points and a set of N object points, and that minimizes the objective function E. The assignment matrix must satisfy the constraint that each image point match at most one object point, and vice versa (i.e., $\Sigma_k m_{ik} = \Sigma_k m_{kj} = 1$ for all i and j). The objective function E will be minimized if the assignment matrix matches image and object points with the smallest distances d_{ij}. (Section 5 describes how these distances are computed.) This problem can be solved by the iterative softassign technique [8, 9]. The iteration for the assignment matrix begins with a matrix m^0 in which element m_{ij}^0 is initialized to $\exp(-\beta(d_{ij}^2 - \alpha))$, 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 large the

distance between two points must be before we consider them unmatchable. The continuous assignment matrix converges toward a discrete matrix due to two mechanisms that are used concurrently:

1. First, a technique due to Sinkhorn [10] 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 one. When the matrix is not square, the sums of the rows and columns will be close to, but not exactly equal to one.

2. The term β is increased as the iteration proceeds. As β increases and each row or column of m is renormalized, the terms m_{ij} corresponding to the smallest d_{ij}^2 tend to converge to one, while the other terms tend to converge to zero. This is a deterministic annealing process [7] known as *softmax* [2]. 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 [8, 9]. The matrix m resulting from an iteration loop that comprises these two substeps is the assignment that minimizes the global objective function E. These two substeps are interleaved in an iteration loop along with the substeps that optimize the pose.

4. Pose from Unknown Line Correspondences

4.1. Geometry of Line Correspondences

Each 3D line in an object is represented by the two 3D endpoints of that line whose coordinates are expressed in the world frame: $L_j = \{\mathbf{P}_j, \mathbf{P}'_j\}$. See figure 1. A line l_i in the image is defined by the two 2D endpoints, $\mathbf{p}_i$ and $\mathbf{p}'_i$, of the line and is represented by the plane of sight that passes through l_i and the camera center $\mathbf{C}$. The normal to this plane is $\mathbf{n}_i = (\mathbf{p}_i, 1) \times (\mathbf{p}'_i, 1)$, and 3D points $\mathbf{P}$ in the camera frame lying on this plane satisfy $\mathbf{n}_i^{\mathsf{T}}\mathbf{P} = 0$.

Let us assume that image line l_i corresponds to object line L_j. If the object has pose given by R and $\mathbf{T}$, then $\mathbf{S}_j = R\mathbf{P}_j + \mathbf{T}$ and $\mathbf{S}'_j = R\mathbf{P}'_j + \mathbf{T}$ lie on the plane of sight through l_i. When R and $\mathbf{T}$ are erroneous and only approximate the true pose, the closest points to $\mathbf{S}_j$ and $\mathbf{S}'_j$ which satisfy this incidence constraint are the orthogonal projections of $\mathbf{S}_j$ and $\mathbf{S}'_j$ onto the plane of sight of l_i:

$$\begin{aligned} \widetilde{\mathbf{S}}_{ij} &= R\mathbf{P}_j + \mathbf{T} - (R\mathbf{P}_j + \mathbf{T}) \cdot \mathbf{n}_i, \\ \widetilde{\mathbf{S}}'_{ij} &= R\mathbf{P}'_j + \mathbf{T} - (R\mathbf{P}'_j + \mathbf{T}) \cdot \mathbf{n}_i, \end{aligned} \tag{8}$$

where we have assumed that $\mathbf{n}_i$ has been normalized to a unit vector. Under the approximate pose R and $\mathbf{T}$, the image points corresponding to object points $\mathbf{P}_j$ and $\mathbf{P}'_j$ can be approximated as the images of $\widetilde{\mathbf{S}}_{ij}$ and $\widetilde{\mathbf{S}}'_{ij}$:

$$\mathbf{p}_{ij} = \frac{(\widetilde{\mathbf{S}}_{ij_x}, \widetilde{\mathbf{S}}_{ij_y})}{\widetilde{\mathbf{S}}_{ij_z}}, \quad \mathbf{p}'_{ij} = \frac{(\widetilde{\mathbf{S}}'_{ij_x}, \widetilde{\mathbf{S}}'_{ij_y})}{\widetilde{\mathbf{S}}'_{ij_z}}. \tag{9}$$

4.2. Computing Pose and Correspondences

The pose and correspondence algorithm for *points* (SoftPOSIT) involves iteratively refining estimates of the pose and correspondences for the given 2D and 3D point sets. The new algorithm for *lines* builds on this approach by additionally refining in the iteration a set of estimated images of the endpoints of the 3D object lines. With this estimated image point set, and the set of object line endpoints, SoftPOSIT is used on each iteration to compute a refined estimate of the object's pose.

On any iteration of the line algorithm, the images of the 3D object lines endpoints are estimated by the point set

$$P_{\text{img}}(R, \mathbf{T}) = \{\mathbf{p}_{ij}, \mathbf{p}'_{ij}, 1 \le i \le M, 1 \le j \le N\}, \tag{10}$$

which is computed using equations (8) and (9). For every 3D endpoint of an object line, there are M possible images of that point, one for each image line. This set of $2MN$ image points depends on the current estimate of the object's pose, and thus changes from iteration to iteration. The object points used by SoftPOSIT are fixed and is the set of $2N$ object line endpoints: $P_{\text{obj}} = \{\mathbf{P}_j, \mathbf{P}'_j, 1 \le j \le N\}$.

We now have a set of $2MN$ image points and a set of $2N$ object points. To use SoftPOSIT, an assignment matrix between the two sets is needed. The initial assignment matrix for point sets P_{img} and P_{obj} is computed from the distances between the image and model lines as discussed in section 5. If l_i and L_j have distance d_{ij}, then all points in P_{img} and P_{obj} derived from l_i and L_j will also have distance d_{ij} . Although the size of this assignment matrix is $(2MN+1) \times (2N+1)$, only $4MN$ of it values are nonzero (not counting the slack row and column). Thus, with a careful implementation, the current algorithm for line features will have the same run-time complexity as the SoftPOSIT algorithm for point features, which was empirically determined to be $O(M^2N)$ [4].

The following is high-level pseudocode for the line-based SoftPOSIT algorithm.

1. Initialize: R, $\mathbf{T}$, β, P_{obj} .

2. Project the model lines into the image using the current pose estimate. Compute the distances d_{ij} between the true image lines and the projected model lines.

3. Initialize the assignment matrix as $m_{ij}^0 = \exp(-\beta(d_{ij}^2 - \alpha))$ and then compute m by normalizing with Sinkhorn's algorithm.

4. Compute $P_{\text{img}}(R, \mathbf{T})$ (equation (10)).

5. Solve for $\mathbf{Q}_1$ and $\mathbf{Q}_2$ (equations (6) and (7)) using m and the point sets P_{obj} and P_{img}^k , and then compute R and $\mathbf{T}$ from $\mathbf{Q}_1$ and $\mathbf{Q}_2$.

6. Stop if R and $\mathbf{T}$ have converged; otherwise, set $\beta = \beta_{\text{update}} \cdot \beta$ and go to step (2).

The algorithm described above performs a deterministic annealing search starting from an initial guess for the object's pose. However, it provides only a local optimum. A common way of searching for a global optimum, and the one taken here, is to run the algorithm starting from a number of different initial guesses, and keep the first solution that meets a specified termination criteria. Our initial guesses range over $[-\pi, \pi]$ for the three Euler angles, and over a 3D space of translations containing the true translation. We use a random number generator to generate these initial guesses. See [4] for details.

5. Distance Measures

The sizes of the regions of convergence to the true pose is affected by the distance measure employed in the correspondence optimization phase of the algorithm. The line-based SoftPOSIT algorithm applies SoftPOSIT to point features where the distances associated with these point features are calculated from the line features. The two main distinguishing features between the different distance measures are (1) whether distances are measured in 3-space or in the image plane, and (2) whether lines are treated as having finite or infinite length. The different distance measures that we experimented with are described below.

The first distance measure that we tried measures distances in the image plane, but implicitly assumes that both image and projected model lines have infinite length. This metric applies a type of Hough transform to all lines (image and projected model) and then measures the distance in this transformed space. The transform that is applied maps an infinite line l to the 2D point $h_k(l)$ on that line which is closest to some fixed reference point r_k. The distance between an image line l_i and the projection l_j of object line L_j with respect to reference point r_k is then $d_{ij}^k = \|h_k(l_i) - h_k(l_j)\|$. Because this Hough line distance is biased with respect to the reference point r_k, for each pair of image and projected object line, we sum the distances computed using five different reference points, one at each corner of the image and one at the image center: $d_{ij} = \sum_{k=1}^{5} \|h_k(l_i) - h_k(l_j)\|$.

The second distance measure that we tried measures distances in the image plane between finite length line segments. The distance between image line l_i and the projection l_j of object line L_j is $d_{ij} = \Delta\theta(l_i, l_j) + \rho d(l_i, l_j)$ where $\Delta\theta(l_i, l_j)$ measures the difference in the orientation of the lines, $d(l_i, l_j)$ measures the difference in the location of the lines, and ρ is a scale factor that determines the relative importance of orientation and location. $\Delta\theta(l_i, l_j) = 1 - |\cos(\angle l_i l_j)|$ where $\angle l_i l_j$ denotes the angle between the lines. Because lines detected in an image are usually fragmented, corresponding only to pieces of object lines, $d(l_i, l_j)$ is the sum of the distance of each endpoint of l_i to the closest point on the finite line segment l_j. So, for a correct pose, $d(l_i, l_j) = 0$ even when l_i is only a partial detection of L_j. This distance measure has produced better performance than the previous measure, resulting in larger regions of convergence and fewer number of iterations to converge.

6. Experiments

6.1. Simulated Images

Our initial evaluation of the algorithm is with simulated data. Random 3D line models are generated by selecting a number of random points in the unit sphere and then connecting each of these points to a small number of the closest remaining points. An image of the model is generated by the following procedure:

1. **Projection:** Project all 3D model lines into the image plane.

2. **Noise:** Perturb with normally distributed noise the locations of the endpoints of each line.

3. **Occlusion:** Delete randomly selected image lines.

4. **Missing ends:** Chop off a small random length of the end of each line. This step simulates the difficulty of detecting lines all the way into junctions.

5. **Clutter:** Add a number of lines of random length to random locations in the image. The clutter lines will not intersect any other line.

Figure 2 shows our algorithm determining the pose and correspondence of a random 3D model with 30 lines, from a simulated image with 40% occlusion of the model, 40% of the image lines being clutter, and normally distributed noise with standard deviation 0.15% of the image dimension (about 1 pixel for a 640×480 image). As seen in this figure, the initial and final projections of the model differ greatly, and so it would be difficult for a person to determine the correspondence of image lines to model lines from the initial projection of the model into the image. Our algorithm, however, is often successful at finding the true pose

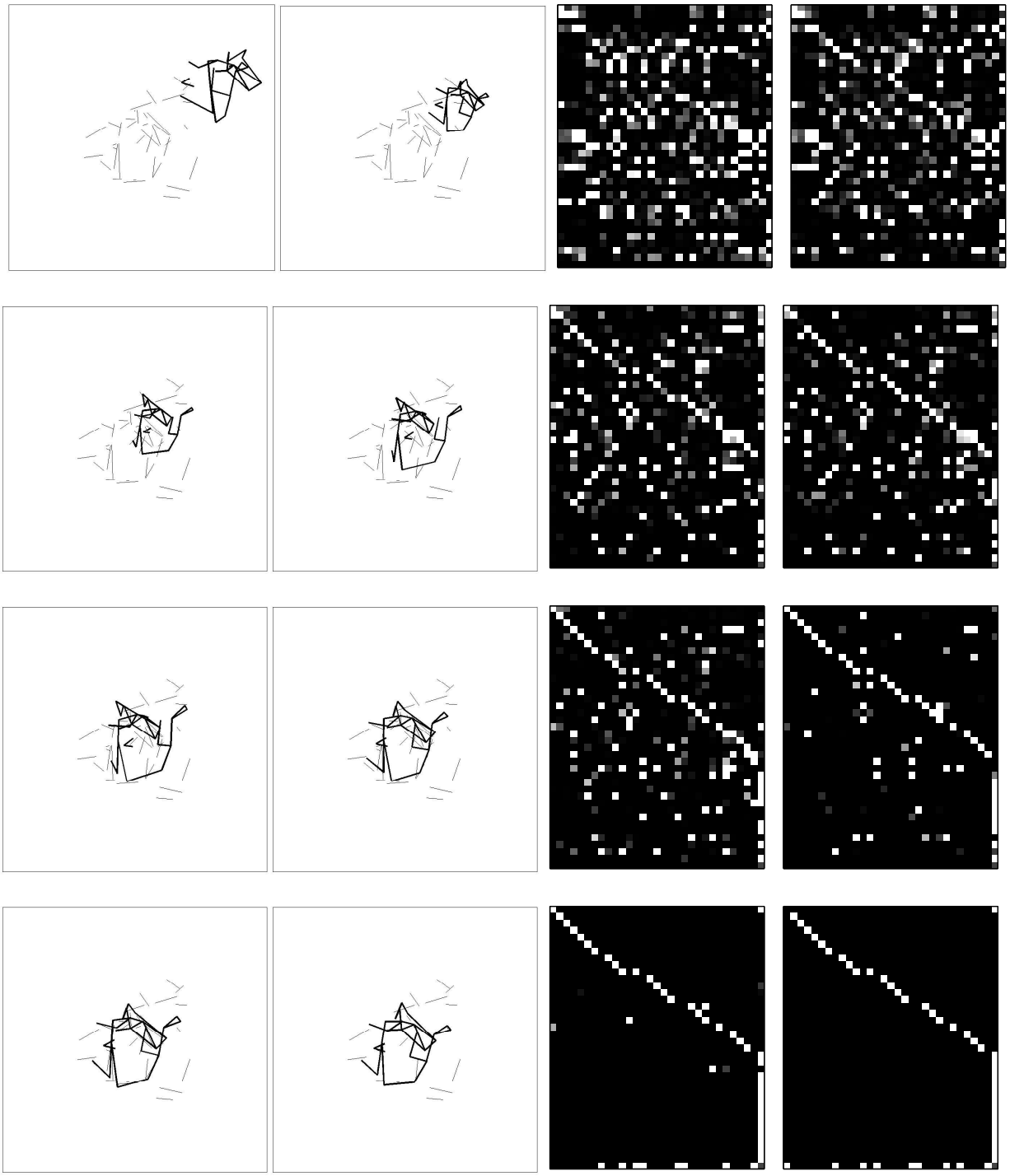

Figure 2: Example application of our algorithm to a cluttered image. The eight frames on the left show the estimated pose at initialization (upper left) and at steps 1, 3, 5, 12, 20, 27, and 35 of the iteration. The thin lines are the image lines and the bold lines are the projection of the model at the current step of the iteration. The correct pose has been found by iteration step 35. The right side of this figure shows the evolution of the assignment matrix at the corresponding steps of the iteration. Because of the way the simulated data was generated, the correct assignments lie near the main diagonal of the assignment matrix. Image lines are indexed along the vertical axis, and model lines along the horizontal axis. Brighter pixels in these figures correspond to greater weight in the assignment matrix. The correct assignments have been found by iteration step 35. Unmatched image and object points are apparent by the large values in the last row and column of the assignment matrix.

from such initial guesses. Although we have not yet done a quantitative evaluation of the algorithm, anecdotal evidence suggests that under 50% occlusion and 50% clutter, the algorithm finds the true pose in about 50% of trials when the initial guess for the pose differs from the true pose by no more than about 30° of rotation about each of the x, y, and z axis. (The initial rotation of the model shown in figure 2 differs from that of the true pose by 28° about each of the coordinate axis.)

6.2. Real Images

Figure 3 shows the results of applying our algorithm to the problem of a robotic vehicle using imagery and a 3D CAD model of a building to navigate through the building. A Canny edge detector is first applied to an image to produce a binary edge image. This is followed by a Hough transform and edge tracking to generate a list of straight lines present in the image. This process generates many more lines than are needed to determine a model's pose, so only a small subset are used by the algorithm in computing pose and correspondence. Also, the CAD model of the building is culled to include only those 3D lines near the camera's estimated position.

7. Conclusions

The simultaneous determination of model pose and model-to-image feature correspondence is very difficult in the presence of model occlusion and image clutter. Experiments with the line-based SoftPOSIT algorithm show that it is capable of quickly solving high-clutter, high-occlusion problems, even when the initial guess for the model pose is far from the true pose. The algorithm solves problems for which a person viewing the image and initial model projection have no idea how to improve the model's pose or how to assign feature correspondences.

We are interested in determining the complexity of the algorithm when no information is available to constrain the model's pose, except for the fact that the model is visible in the image. This will allow us to compare the efficiency of line-based SoftPOSIT to other algorithms. The key parameter that needs to be determined is the number of initial guesses required to find a good pose, as a function of clutter, occlusion, and noise. We expect that the line-based algorithm will require many fewer initial guesses than point-based SoftPOSIT algorithm.

Figure 3: Determining the pose of a CAD model from a real image. Straight lines are automatically detected in the image (top). The initial guess for the pose of the hallway model differed from the true pose by about 14° about each coordinate axis (middle). The projection of the model after finding its pose with our algorithm (bottom).

References

[1] J.R. Beveridge and E.M. Riseman, "Optimal Geometric Model Matching Under Full 3D Perspective," *Computer Vision and Image Understanding*, vol. 61, no. 3, pp. 351–364, May 1995.

[2] Bridle, J.S. “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, 1990.

[3] S. Christy and R. Horaud. “Iterative Pose Computation from Line Correspondences,” *Computer Vision and Image Understanding,* Vol. 73, No. 1, pp. 137-144 (January 1999).

[4] P. David, D.F. DeMenthon, R. Duraiswami, and H. Samet. "SoftPOSIT: Simultaneous Pose and Correspondence Determination." European Conference on Computer Vision (ECCV), Copenhagen, Denmark, May 2002, pp. 698-714.

[5] D.F. DeMenthon and L.S. Davis. “Model-Based Object Pose in 25 Lines of Code,” *International Journal of Computer Vision,* Vol. 15, pp. 123-141, June 1995.

[6] 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, no. 6, pp. 381–395, June 1981.

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

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

[9] S. Gold, A. Rangarajan, C.P. Lu, S. Pappu, E. Mjolsness. “New Algorithms for 2D and 3D Point Matching: Pose Estimation and Correspondence,” *Pattern Recognition,* Vol. 31, No. 8, August 1998, pp. 1019-1031.

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

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