LAMP-TR-123 JULY 2005
CAR-TR-1010
CS-TR-4731
UMIACS-TR-2005-38

OBJECT RECOGNITION BY DETERMINISTIC ANNEALING OF RANKED AFFINE POSE HYPOTHESES

Philip David and Daniel DeMenthon

Language and Media Processing Laboratory
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742-3275
phild@cs.umd.edu, daniel@cfar.umd.edu

Abstract

We present an object recognition algorithm that uses model and image line features to locate complex objects in high clutter environments. Finding correspondences between model and image features is the main challenge in most object recognition systems. In our approach, corresponding line features are determined by a three-stage process. The first stage generates a large number of approximate pose hypotheses from correspondences of one or two lines in the model and image. Next, the pose hypotheses from the previous stage are quickly evaluated and ranked by comparing local image neighborhoods to the corresponding local model neighborhoods. Fast nearest neighbor and range search algorithms are used to implement a distance measure that is unaffected by clutter and partial occlusion. The ranking of pose hypotheses is invariant to changes in image scale, orientation, and partially invariant to affine distortion. Finally, a robust pose estimation algorithm is applied for refinement and verification, starting from the few best approximate poses produced by the previous stages. Experiments on real images demonstrate robust recognition of partially occluded objects in very high clutter environments.

Keywords: object recognition, line matching, feature correspondence, geometric pattern recognition

1 Introduction

Object recognition in cluttered environments is a difficult problem with widespread applications. Most approaches to object recognition, including the one presented here, rely on first finding correspondences between model features and image features, then computing a hypothesized model pose, and finally searching for additional image features that support this pose. The most challenging part of this process is the identification of corresponding features when the images are affected by clutter, partial object occlusion, changes in illumination, and changes in viewpoint. In fact, once the feature correspondence problem is solved, object recognition becomes almost trivial. A wide variety of features have been employed by object recognition systems, including points, edges, and textured regions. There are advantages and disadvantages to each type of feature, and each is suitable for different applications.

The surfaces of many objects consist of regions of uniform color or texture. Most of the information available for object recognition is at the boundaries (edges) of these regions; a line drawing representation of these objects provides a nearly complete description of these objects. Approaches to object recognition that rely on variations in texture internal to these regions are likely to perform poorly. Furthermore, objects that are composed of thin, stick-like components, such as bicycles, chairs, and ladders, are especially difficult for texture-based approaches because background clutter will be present within a few pixels of any object pixel, thus corrupting local texture templates [4]. Methods that rely on the boundary shapes are better suited to these types of objects. This paper presents a simple, effective, and fast method for recognizing partially occluded 2D objects in cluttered environments, where the object models and their images are each described by sets of line segments. A fair amount of perspective distortion is tolerated by the algorithm, so the algorithm is also applicable to 3D objects that are represented by sets of viewpoint-dependent 2D models. Because of the close ties between object recognition and feature correspondence, this paper is about a new feature correspondence algorithm as much as it is about a new object recognition algorithm.

Our approach assumes that at least *one* model line is detected as an unfragmented line in the image. By *unfragmented*, we mean that the corresponding image line is extracted from the image as a single continuous segment between the two endpoints of the projected model line.

This necessarily requires that at least one model line be unoccluded. Additional model lines must be present in the image for verification, but these may be partially occluded or fragmented. A potential difficulty with this approach is that line detection algorithms often fragment lines due to difficulties in parameter selection, and they usually don't extract lines completely at the intersections with other lines. The issue of fragmentation resulting from poor parameter selection can be ameliorated through post-processing steps that combine nearby collinear lines. However, this has not been necessary in any of our experiments. The issue of line detection algorithms being unable to accurately locate the endpoints of lines at the intersections with other lines does not cause a problem because a few missing pixels at the ends of a line does not significantly affect the computed model transformations (except in the case that the object's image is so small as to make recognition difficult regardless of how well the object's edges are detected). We show below that our line detector is able to detect a large number of object lines with very little relative error in their length when compared to the corresponding projected model lines.

A three-stage process is used to locate objects. In the first stage, a list of approximate model pose hypotheses is generated. Every pairing of a model line to an image line first contributes a pose hypothesis consisting of a similarity transformation. When both the model line and the corresponding image line form corner-like structures with other nearby lines, and the angles of the corners are similar (within 45°), a pose hypothesis consisting of an affine transformation is added to the hypothesis list, one for each such compatible corner correspondence. Typically, each model-to-image line correspondence contributes a small number of poses (one to six) to the hypothesis list.

We make use of information inherent in a single line correspondence (position, orientation, and scale) to reduce the number of correspondences that must be examined in order to find an approximately correct pose. For m model lines and n image lines, we generate $\mathcal{O}(mn)$ approximate pose hypotheses. Compare this to traditional algorithms that generate precise poses from three pairs of correspondences, where there are up to $\mathcal{O}(m^3n^3)$ pose hypotheses. An approach such as RANSAC [9], which examines a very small fraction of these hypotheses, still has to examine $\mathcal{O}(n^3)$ poses to ensure with probability 0.99 that a correct precise pose will be found [6]. By starting with an approximate pose instead of a precise pose, we are able to

greatly reduce the number of poses that need to be examined, and still find a correct precise pose in the end.

Most of the pose hypotheses will be inaccurate because most of the generating correspondences are incorrect. The second stage of our approach ranks each pose hypothesis based on the similarity of the corresponding local neighborhoods of lines in the model and image. The new similarity measure is largely unaffected by image clutter, partial occlusion, and fragmentation of lines. Nearest-neighbor search is used in order to compute the similarity measure quickly for many pose hypotheses. Because this similarity measure is computed as a function of approximate pose, the ranking of the pose hypotheses is invariant to image translation, scaling, rotation, and partially invariant to affine distortion of the image. By combining the process of pose hypothesis generation from assumed unfragmented image lines with the neighborhood similarity measure, we are able to quickly generate a ranked list of approximate model poses which is likely to include a number of highly ranked poses that are close to the correct model pose.

The final stage of the approach applies a more time-consuming but also more accurate pose refinement and verification algorithm to a few of the most highly ranked approximate poses. Gold's graduated assignment algorithm [10, 11], modified for line correspondences, is used for this purpose because it is efficient, tolerant of clutter and occlusion, and doesn't make hard correspondence decisions until an optimal pose is found.

Our three-stage approach allows CPU resources to be quickly focused on the highest payoff pose hypotheses, which in turn results in a large reduction in the amount of time needed to perform object recognition. An outline of the algorithm is shown in Figure 1. In the following sections, we first describe related work, and then describe each step of our algorithm in more detail. Although any line detection algorithm may be used, Section 3 briefly discusses the line detection algorithm that we use and presents an evaluation of its ability to extract unfragmented lines from an image. Section 4 then shows how approximate pose hypotheses are generated from a minimal number of line correspondences. Next, in Sections 5 and 6, we present our method for efficiently comparing local neighborhoods of model lines to local neighborhoods of image lines. Section 7 describes the pose refinement and verification algorithm that we use. Experiments with real imagery containing high levels of clutter and occlusion (see Figure 2,

```
Create a data structure for nearest neighbor and range searches of image lines.
Using a range search, identify corners in each model and in the image.
for each model do
  H = Ø.                                   // Initialize hypotheses list to empty.
  for each pair of model line l and image line l', do
    C = Pose hypotheses generated from l, l', and nearby corners.
    H = H ∪ C.
    Evaluate the similarity of model and image neighborhoods for poses C.
  end for
  P = Sort H based on neighborhood similarity measure.
  for i = 1 to N do
    Apply the graduated assignment algorithm starting from pose P(i).
    if a sufficient number of line correspondences are found then
      An object has been recognized.
    end if
  end for
end for
```

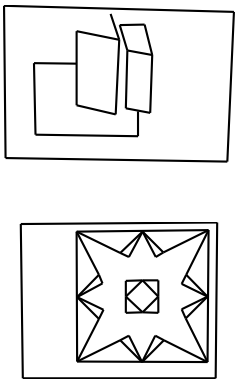
Figure 1: Outline of the new object recognition algorithm. The constant N is the number of pose refinements performed; as discussed in Section 8, good performance is obtained with $N = 4$.

for example) are discussed in Section 8 and demonstrate the effectiveness of the algorithm; this section also gives the run-time complexity of the algorithm. We see that our algorithm is faster and able to handle greater amounts of clutter than previous approaches that use line features. The approach is able to recognize planar objects that are rotated by up to 60° away from their modeled viewpoint, and recognize 3D objects from 2D models that are rotated by up to 30° from their modeled viewpoint. The paper ends with conclusions in Section 9.

2 Related Work

Automatic registration of 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.

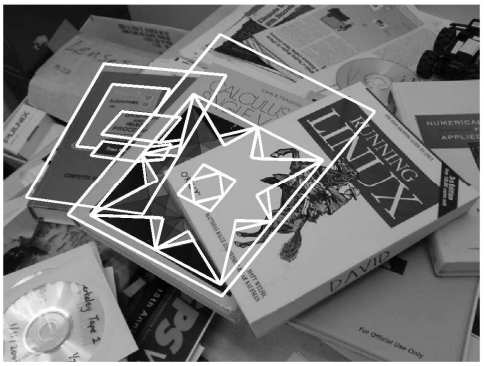
A wide variety of approaches to object recognition have been proposed since Robert's ground-breaking work on recognizing 3D polyhedral objects from 2D perspective images [17]. Among the pioneering contributions are Fischler and Bolles' RANSAC method [9], Baird's



(a) Models.

(b) Test image.

(c) 519 detected lines.



(d) Recognized books.

Figure 2: Recognizing books in a pile. The two models were generated from frontal images of the books.

tree-pruning method [2], and Ullman's alignment method [20]. These approaches, which hypothesize poses from small sets of correspondences and reject or accept those poses based on the presence of supporting correspondences, become intractable when the number of model and image features becomes large, especially when the image contains significant clutter.

More recently, the use of rich feature descriptors has become popular as a way of reducing the number of feature correspondences that must be examined. The Harris corner detector [12] has seen widespread use for this purpose; however, it is not stable to changes in image scale, so it performs poorly when matching models and images of different scales. Schmid and

Mohr [18] have developed a rotationally invariant feature descriptor using the Harris corner detector. Lowe [14] extended this work to scale invariant and partially affine invariant features with his SIFT approach, which uses scale-space methods to determine the location, scale, and orientation of features, and then, relative to these parameters, a gradient orientation histogram describing the local texture. Excellent results have been obtained by approaches using these rich features when objects have significant distinctive texture. However, there are many common objects that possess too little distinctive texture for these methods to be successful. Examples include thin objects such as bicycles and ladders where background clutter will be present near all object boundaries, and uniformly textured objects such as upholstered furniture. In these cases, only the relations between geometric features (such as points and edges) can be used for matching and object recognition. Edges are sometimes preferedpreferred over points because they are easy to locate and are stable features on both textured and nontextured objects.

Our approach has some similarities to Ayache and Faugeras's HYPER system [1]. They use a tree-pruning algorithm to determine 2D similarity transformations that best align 2D object models with images, where both the models and images are represented by sets of line segments. The ten longest lines in the model are identified as "privileged" segments. The privileged segments are used for initial hypothesis generation because there are fewer of them (so fewer hypotheses have to be generated), and because the use of long segments results in more accurate pose estimates. The authors point out that the probability of having all privileged segments simultaneously occluded is very small, and only one privileged segment needs to be visible to identify a model. Although this is true, we believe that long model lines are just as likely as short model lines to be fragmented in an image, and therefore we treat all model lines identically and do not identify any as privileged. In the HYPER system, 2D pose hypotheses are generated by matching each privileged model line to every compatible image line, where compatibility is defined in terms of the difference in the angles of corners formed with neighbors, and the difference in scale from an *a priori* scale. The hypotheses are ranked based on the degree of compatibility of the matched segments, and then the best hypotheses are refined using a tree search to locate additional matching model and image segments that are compatible with the initial pose estimate. During this tree search, a pose hypothesis is

augmented with additional matches when the difference in orientation of the two segments, the Euclidean distance between their midpoints, and the relative difference between their lengths, are all small. In contrast, our pose hypotheses are based on affine transformations instead of similarity transformations, we use a dissimilarity measure (see Section 6) to rank hypotheses that is less affected by line fragmentation because it does not depend on the lengths of lines nor on unique reference points on the lines, and we use the more robust and efficient graduated assignment algorithm [10] for pose refinement.

A number of more recent works [16, 4] have also used edges for object recognition of poorly textured objects. Mikolajczyk et al. [16] generalize Lowe's SIFT descriptors to edge images, where the position and orientation of edges are used to create local shape descriptors that are orientation and scale invariant. Carmichael's approach [4] uses a cascade of classifiers of increasing aperture size, trained to recognize local edge configurations, to discriminate between object edges and clutter edges; this method requires many training images to learn object shapes, and it is not invariant to changes in image rotation or scale.

Gold and Rangarajan [11] simultaneously compute pose and 2D-to-2D or 3D-to-3D point correspondences using deterministic annealing to minimize a global objective function. We previously used this method [5] for matching 3D model lines to 2D image lines, and we use it here for the pose refinement stage of our algorithm. Beveridge [3] matches points and lines using a random start local search algorithm. Denton and Beveridge [7] extended this work by replacing random starts with a heuristic that is used to select which initial correspondence sets to apply the local search algorithm. Although we use line features instead of point features, Denton's approach is conceptually similar to ours in a number of ways. Both approaches first hypothesize poses using small sets of local correspondences, then sort the hypotheses based on a local match error, and finally apply a pose refinement and verification algorithm to a small number of the best hypotheses. Significant differences between the two approaches are that ours uses lines instead of points, and also zero or one neighboring features, instead of four, to generate pose hypotheses; so our approach will have many fewer hypotheses to consider, and each hypothesis is much less likely to be corrupted by spurious features (clutter).

3 Line Detection

Line segments in models are matched to line segments in images. Each line segment in a model or image is represented by its two endpoints. Generation of model lines may be performed manually by the user, or automatically by applying image processing to images of the objects to be recognized. We currently use publicly available software [13] to automatically locate model and image lines. Briefly, this software operates as follows. The Canny edge detector is first applied. Next, contiguous edge pixels are linked together into contours and very short contours are discarded. Each contour is then partitioned into line segments by breaking the contour at edge pixels until no edge pixel is more than a specified distance from the line connecting the two endpoints of its subcontour; this is done by finding the longest subcontour (starting at the first edge point) whose maximum distance from the line connecting the endpoints of the subcontour is less than a threshold. This subcontour is replaced by the line segment, and then the process is repeated for the remainder of the contour.

In our experience, for images with dense edges, this approach to line detection performs better than the Hough Transform approach [8]. The high-connectivity of the edges produced by the Canny edge detector greatly simplifies the process of fitting lines to those contours when the contour partitioning approach is used. Line fitting using the Hough Transform, on the other hand, is easily confounded by spurious peaks generated by coincidental alignment of physically separated edge points.

A requirement of our approach, as stated in Section 1, is to detect at least one unfragmented image line segment. An evaluation of the accuracy of our line detector shows that this requirement is easily satisfied for the types of scenes described in this paper. Using six different images of books and office objects (as typified by images shown throughout this paper), we manually measured the length of 250 projected model lines and the lengths of the corresponding automatically detected line segments. All model edges that were partially or fully visible were measured. If a visible model edge was not detected by our software, then the "corresponding line segment" was assigned a length of zero. For each model edge, the relative error in the length of the corresponding detected line segment is calculated as $|(l_m - l_i)/l_m|$ where l_m is the length of the projected model line and l_i is the length of the detected line segment.

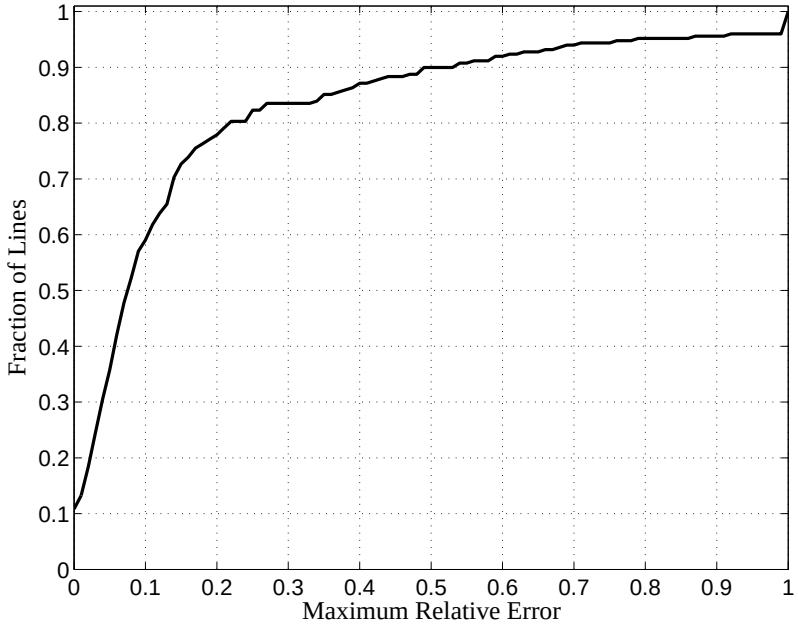


Figure 3: The accuracy of our line detector is depicted in this graph, which plots the relative error versus the fraction of the model lines detected with relative error no greater than that amount.

Figure 3 shows a plot of the relative error versus the fraction of the 250 model lines that are detected with relative error no greater than that amount. One can see that 11% of all partially and fully visible model lines are detected in the images with less than one pixel error in the positions of their endpoints. Furthermore, 35% of all model lines are detected as image segments where the sum of the errors in the endpoint positions is no more than 5% of the length of the corresponding projected model lines; That is, 35% of the visible model lines are detected with relative error in length that is less than 5%. We find that 5% relative error is small enough to obtain a good coarse pose hypothesis, and that with 35% of the model lines having relative errors no larger than this, there will be many such good hypotheses that will allow the pose refinement stage of the algorithm to recognize an object.

4 Generating Pose Hypotheses

We wish to generate a small set of approximate poses that, with high certainty, includes at least one pose that is close to the true pose of the object. The smaller the number of correspondences

used in estimating a pose, the less likely the estimated pose will be corrupted by spurious correspondences. But at the same time, using fewer correspondences will produce a less accurate pose when all correspondences used by the estimation are correct. From a single correspondence of a model line to an image line, where the image line may be fragmented (only partially detected due to partial occlusion or faulty line detection), we can compute the 2D orientation of the model as well as a one-dimensional constraint on its position, but the scale and translation of the model cannot be determined; this does not provide sufficient geometric constraints to evaluate the similarity of a local region of the model with a local region of the image.

On the other hand, if we assume that a particular image line is unfragmented, then from a single correspondence of a model line to this image line we can compute a 2D similarity transformation of the model. This is possible because the two endpoints of the unfragmented image line must correspond to the two endpoints of the model line, and two corresponding points are sufficient to compute a similarity transformation. A similarity transformation will be accurate when the viewing direction used to generate the 2D model is close to the viewing direction of the object. However, even when there is some perspective distortion present, approximate similarity transformations from correct correspondences are often highly ranked by the next stage of our approach. Generating hypothesized poses that are highly ranked in the next stage is the main goal of this first stage since the pose refinement algorithm used in the final stage has a fairly large region of convergence.

Because we don't know which endpoint of the model line corresponds to which endpoint of the image line, we consider both possibilities and generate a similarity transformation for each. For $\mathbf{p}_1$ and $\mathbf{p}_2$ model line endpoints corresponding to image line endpoints $\mathbf{q}_1$ and $\mathbf{q}_2$, respectively, the similarity transformation mapping the model to the image is $\mathbf{q}_i = A\mathbf{p}_i + \mathbf{t}$ where $A = sR$ and s, R, and $\mathbf{t}$ are the scaling, rotation, and translation, respectively, defined by

$$\begin{aligned} s &= \|\mathbf{q}_1 - \mathbf{q}_2\| \,/\, \|\mathbf{p}_1 - \mathbf{p}_2\|, \\ R &= \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \\ \mathbf{t} &= \mathbf{q}_1 - A\mathbf{p}_1, \end{aligned}$$

and where θ is the rotation angle (in the range $-\pi$ to π, clockwise being positive) from $\mathbf{p}_1 - \mathbf{p}_2$

to $\mathbf{q}_1 - \mathbf{q}_2$.

We can obtain more accurate approximate poses with little additional work when the model line and the unfragmented image line (called the *base lines* below) form corner-like structures with other lines: corners in the model should correspond to corners in the image. Corners in the model are formed by pairs of model lines that terminate at a common point, while corners in the image are formed by pairs of image lines that terminate within a few pixels of each other. By looking at corners, we expand our search to correspondences of two line pairs. However, because we restrict the search for corner structures in the image to lines that terminate within a few pixels of an endpoint of a base image line, the number of corners examined for any base image line is usually quite small. As before, we assume only that the base image line is unfragmented; other image lines may be fragmented. If a base model line forms a corner with another model line, which is usually the case for objects described by straight edges, and if the base image line is unfragmented, then all model lines that share an endpoint with the base model line should be unoccluded around that endpoint in the image, and therefore there is a good chance that these other models lines will appear in the image near the corresponding endpoint of the base image line. Thus, looking at corners formed with the base image lines provides a way of finding additional line correspondences with a low outlier rate.

The model and image lines which participate in corner structures are efficiently located using a range search algorithm [15]. The endpoints of all image lines are first inserted into a search tree data structure. Then, for each endpoint of each image line, a range search is performed to locate nearby endpoints and their associated lines. A similar process is performed for the model lines. This preprocessing step is done once for each model and image. To generate pose hypotheses for a particular base correspondence, the angles of corners formed with the base model line are compared to the angles of corners formed with the base image line. An affine pose hypothesis is generated for any pair of corner angles that are within $45°$. As before, this is repeated for each of the two ways that the base model line can correspond to the base image line. Note that these affine pose hypotheses are generated in addition to the similarity pose hypotheses describe above. The similarity pose hypotheses are kept even though they may be less accurate because the affine pose hypotheses are more susceptible to being corrupted by spurious correspondences.

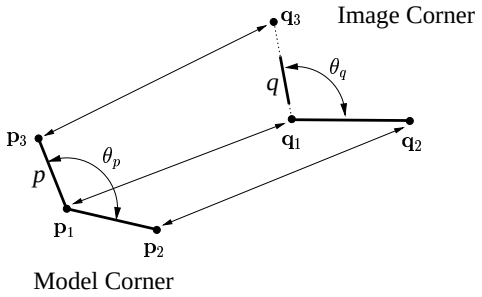


Figure 4: Geometry for calculation of the approximate affine transformation.

An affine pose hypothesis is generated as follows. Let $\mathbf{p}_1$ and $\mathbf{p}_2$ be the endpoints of the base model line, and $\mathbf{q}_1$ and $\mathbf{q}_2$ be the corresponding endpoints of the base image line. See Figure 4. Assume that a pair of corners is formed with the base lines by model line p and image line q that terminate near endpoints $\mathbf{p}_1$ and $\mathbf{q}_1$, and have angles θ_p and θ_q, respectively. We have two pairs of corresponding points and one pair of corresponding angles. Since a 2D affine transformation has 6 degrees of freedom but we have only 5 constraints (two for each point correspondence, and one for the angle correspondence), we impose the additional constraint that the affine transformation must scale the length of line p in the same way as it does the length of the base model line $\mathbf{p}_1\mathbf{p}_2$. This, defines a third pair of corresponding points $\mathbf{p}_3$ and $\mathbf{q}_3$, on p and q, respectively, as shown in Figure 4. $\mathbf{p}_3$ is the second endpoint of p, and $\mathbf{q}_3$ is the point collinear with q such that

$$\frac{\|\mathbf{p}_2 - \mathbf{p}_1\|}{\|\mathbf{q}_2 - \mathbf{q}_1\|} = \frac{\|\mathbf{p}_3 - \mathbf{p}_1\|}{\|\mathbf{q}_3 - \mathbf{q}_1\|}.$$

$\mathbf{q}_3$ is found to be

$$\mathbf{q}_1 + \frac{\|\mathbf{p}_3 - \mathbf{p}_1\| \, \|\mathbf{q}_2 - \mathbf{q}_1\|}{\|\mathbf{p}_2 - \mathbf{p}_1\|^2} \begin{bmatrix} \cos\theta_q & -\sin\theta_q \\ \sin\theta_q & \cos\theta_q \end{bmatrix} (\mathbf{p}_2 - \mathbf{p}_1) .$$

The affine transformation mapping a model point $\mathbf{p}_i = \begin{bmatrix} p_{i_x} & p_{i_y} \end{bmatrix}^\mathsf{T}$ to the image point $\mathbf{q}_i =$

$\begin{bmatrix} q_{i_x} & q_{i_y} \end{bmatrix}^{\mathsf{T}}$ is

$$\begin{bmatrix} q_{i_x} \\ q_{i_y} \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} p_{i_x} \\ p_{i_y} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}. \quad (1)$$

For each correspondence $\mathbf{p}_i \leftrightarrow \mathbf{q}_i$ we have two linear equations in the 6 unknowns a_1, a_2, a_3, a_4, t_x, and t_y. From the three corresponding points, we can solve for the parameters of the affine transformation. Figure 5 shows the pose hypotheses generated for a particular correct base correspondence.

5 Similarity of Line Neighborhoods

The second stage of the recognition algorithm ranks all hypothesized approximate model poses in the order that the pose refinement and verification algorithm should examine them; the goal is to rank highly those poses that are most likely to lead the refinement and verification algorithm to a correct precise pose. This way, the final stage can examine the smallest number of approximate poses needed to ensure that a correct pose will be found if an object is present. For this purpose, a geometric measure of the similarity between the model (transformed by an approximate pose) and the image is computed. To ensure that this similarity measure can be computed quickly, for any base model line generating a hypothesized pose, only a local region of model lines surrounding the base line (called the base model line's *neighborhood*) is compared to the image lines. Let $\mathcal{M}$ be the set of lines for a single model and $\mathcal{I}$ be the set of image lines. We define the *neighborhood radius* of a line l to be the smallest distance, denoted $r(l)$, such that the two endpoints of at least N_{nbr} lines (excluding l) are within distance $r(l)$ of l. In all of our experiments, the value of N_{nbr} is fixed at 10 lines ($N_{\text{nbr}} \leq |\mathcal{M}|$). The neighborhood of a model line l is the set of N_{nbr} model lines, $\mathcal{N}(l)$, whose endpoints are within distance $r(l)$ of l. Figure 6 illustrates a line and its neighbors.

For a hypothesized approximate model pose $\{A, \mathbf{t}\}$ generated for a base model line l, let $\mathcal{T}(\mathcal{N}(l), A, \mathbf{t})$ denote the neighbors of l transformed by the pose $\{A, \mathbf{t}\}$, and let $d(l', l'')$ denote the distance (defined in Section 6) between two lines l' and l'' in the image. Then, the geometric similarity between a model neighborhood $\mathcal{N}$ transformed by the pose $\{A, \mathbf{t}\}$ and the

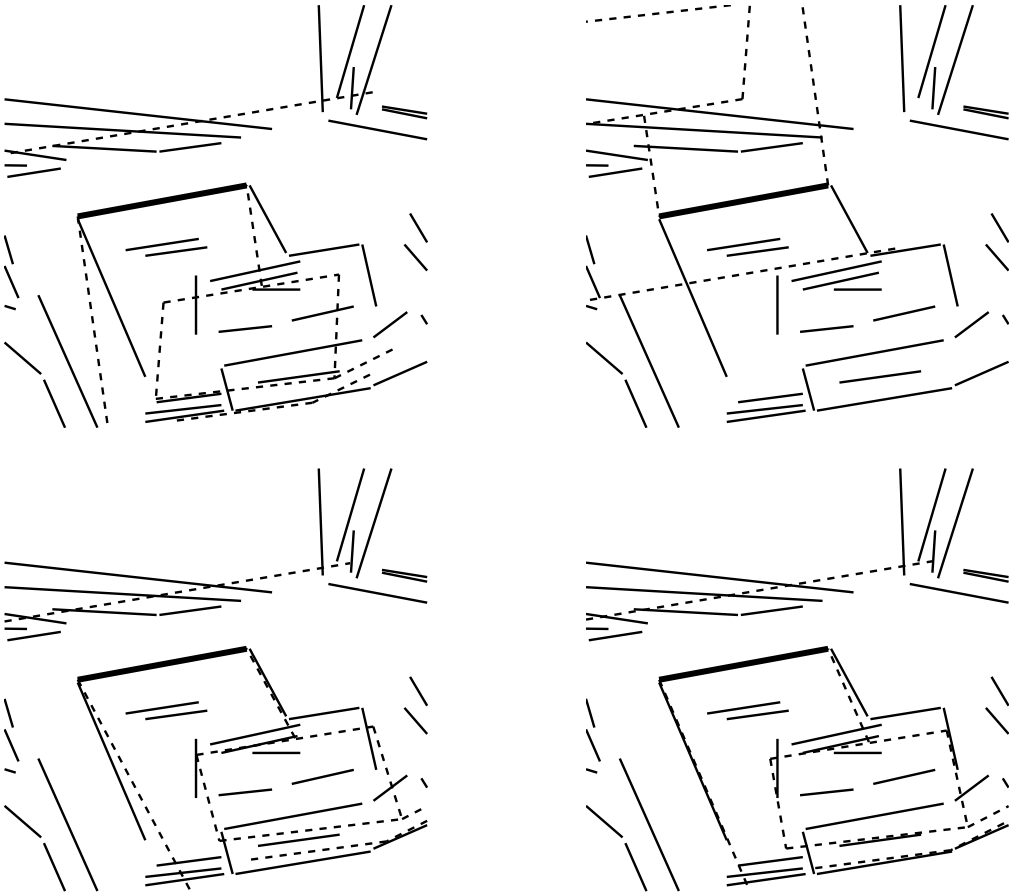


Figure 5: Pose hypotheses generated for a correct correspondence of a real model and image line are shown. The model lines (dashed lines and thick solid line) are shown overlaid on the image lines (thin lines). The one thick solid line in each image shows the base correspondence: a model line perfectly aligned with an image line. The top row shows the two similarity transformations, one for each possible alignment of the base lines. The bottom row shows the two affine transformations, one for each possible corner correspondence of the base lines. These are the complete set of transformations hypothesized for this base correspondence. Notice the better alignment in the images of the bottom row, resulting from the use of corner angle correspondences, compared to the upper left image.

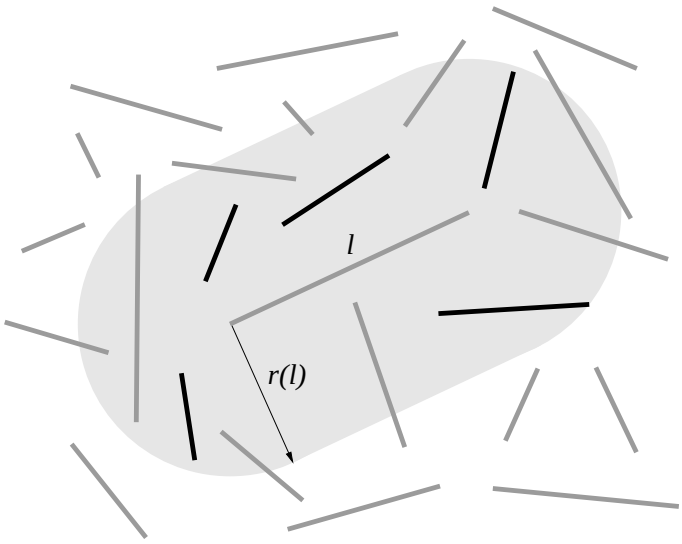


Figure 6: The neighborhood radius of line l, in the center of the image, is the minimum distance $r(l)$ for which both endpoints of N_{nbr} lines are within distance $r(l)$ of l. Here, $N_{\text{nbr}} = 5$, but in actual experiments, we take $N_{\text{nbr}} = 10$.

set of image lines $\mathcal{I}$ is

$$S\left(\mathcal{N}, \mathcal{I}, A, \mathbf{t}\right) = \sum_{l' \in \mathcal{T}(\mathcal{N}, A, \mathbf{t})} \min\left\{ S_{\text{max}}, \min_{l'' \in \mathcal{I}} d\left(l', l''\right) \right\}. \tag{2}$$

The smaller the value of $S\left(\mathcal{N}, \mathcal{I}, A, \mathbf{t}\right)$, the more "similar" a model neighborhood $\mathcal{N}$ is to the image $\mathcal{I}$ under the transformation $\{A, \mathbf{t}\}$. The parameter S_{max} ensures that "good" poses are not penalized too severely when a line in the model is fully occluded in the image. This parameter is easily set by observing the values of $S\left(\mathcal{N}, \mathcal{I}, A, \mathbf{t}\right)$ that are generated for poor poses (that should be avoided), and then setting S_{max} to this value divided by N_{nbr}.

As explained in Section 6, the distance between a single model neighbor and the closest image line can be found in time $\mathcal{O}\left(\log n\right)$ when there are n image lines. Since $|\mathcal{N}| = N_{\text{nbr}}$, the time to compute $S\left(\mathcal{N}, \mathcal{I}, A, \mathbf{t}\right)$ is $\mathcal{O}\left(\log n\right)$.

6 Distance Between Lines

For any image line l' (which is typically a transformed model line), we wish to efficiently find the line $l'' \in \mathcal{I}$ that minimizes $d(l', l'')$ in Equation 2 that expresses the similarity between a model and image neighborhood. This search can be performed efficiently when each line is represented by a point in an *N*-dimensional space and the distance between two lines is the Euclidean distance between the corresponding points in this *N*-dimensional space. Assuming that we have a suitable line representation, a tree data structure storing these *N*-dimensional points can be created in time $\mathcal{O}(n \log n)$ and the closest image line can be found in time $\mathcal{O}(\log n)$. This tree structure need only be created once for each image, and is independent of the model lines.

Thus, we want to represent each line as a point in an *N*-dimensional space such that the Euclidean distance between two lines is small when the two lines are superposed. We would also like the distance function to be invariant to partial occlusion and fragmentation of lines. Representing a line by its 2D midpoint is insufficient because two lines can have an identical midpoint but different orientations. We could use the midpoint and orientation of a line, but a short line superposed on a longer line (think of the short line as a partially occluded version of the longer line) could be assigned a large distance because their midpoints may be far. Further, there is problem associated with line orientation because a line with an orientation of θ should produce the same distance as when its orientation is given as $\theta \pm 2k\pi$ for $k = 1, 2, \ldots$. For example, two lines with identical midpoints but orientations 179° and -179° should produce the same distance as if the orientations of the two lines were 1° and -1°. It is not possible with a Euclidean distance function to map both of these pairs of angles to the same distance. A solution to these occlusion and orientation problems is to generate multiple representations of each line.

Let l be a line with orientation θ (relative to the horizontal, $0 \leq \theta \leq \pi$) and endpoints $[x_1, y_1]$ and $[x_2, y_2]$. When l is a line in the image ($l \in \mathcal{I}$), l is represented by the two 3D points

$$\left[\frac{\theta}{r_\theta}, \frac{x_{\text{mid}}}{r_m}, \frac{y_{\text{mid}}}{r_m}\right] \quad \text{and} \quad \left[\frac{\theta - \pi}{r_\theta}, \frac{x_{\text{mid}}}{r_m}, \frac{y_{\text{mid}}}{r_m}\right] \tag{3}$$

where $[x_{mid}, y_{mid}] = [x_1 + x_2, y_1 + y_2]/2$ is the midpoint of the line, and r_θ and r_m are con-

stant scale factors (described below); these are the 3D points used to create the data structure that's used in the nearest neighbor searches.

When l is a transformed model line (as in l' above), l is represented by the set of 3D points

$$\left\{ \left[\frac{\theta}{r_\theta}, \frac{\bar{x}_i}{r_m}, \frac{\bar{y}_i}{r_m} \right], \left[\frac{\theta - \pi}{r_\theta}, \frac{\bar{x}_i}{r_m}, \frac{\bar{y}_i}{r_m} \right], i = 1, 2, \ldots, N_{\text{pts}} \right\} \tag{4}$$

where

$$N_{\text{pts}} = \left\lceil \frac{\| [x_2 - x_1, y_2 - y_1] \|}{w} \right\rceil + 1, \tag{5}$$

$$\Delta = \frac{[x_2 - x_1, y_2 - y_1]}{N_{\text{pts}} - 1},$$

$$[\bar{x}_i, \bar{y}_i] = [x_1, y_1] + (i - 1)\, \Delta.$$

In words, two orientations are used for each transformed model line, but the position of the line is represented by a series of N_{pts} points that uniformly sample the length of the line at an interval w. The reason multiple sample points are required to represent the position of transformed model lines but not the image lines is that when $\{A, \mathbf{t}\}$ is a correct pose for the model, the image line, which may be partially occluded or otherwise fragmented, will in general be shorter than the transformed model line. In this case, the midpoint of the image line will lie somewhere along the transformed model line, but the midpoint of the transformed model line may lie off of the image line. The occlusion problem of the representation is only truly eliminated by a uniform distribution of sample points along transformed model lines when there is a sample point $[\bar{x}_i, \bar{y}_i]$ for every pixel on a transformed model line. However, we have found that placing a sample point at approximately every 10^{th} pixel ($w = 10$ in Equation 5) along each transformed model line is sufficient to solve this problem. Then, when a transformed model line l' is correctly aligned with a possibly partially occluded image line l'', we will have $d(l', l'') \leq w/r_m$. Figure 7 illustrates how model and image lines are sampled in the process of generating the points in Equations 3 and 4.

The scale factors r_θ and r_m are chosen to normalize the contribution of the orientation and position components to the distance measure. Given a model line l with neighborhood radius $r(l)$ that is mapped to an image line l', the radius around l' in which lines corresponding to

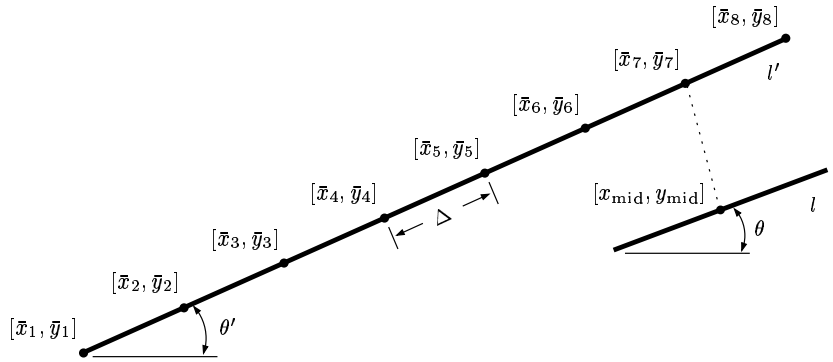


Figure 7: Sampling the position of model and image lines. Image line l is represented using its midpoint $[x_{\text{mid}}, y_{\text{mid}}]$ and its orientation θ. Projected model line l' is represented using the points $\{[\bar{x}_i, \bar{y}_i], i = 1, \ldots, N_{\text{pts}}\}$ and its orientation θ'. When the pose of a model is accurate and a model line and image line correspond, the midpoint of that image line will be close to some sample point on the projected model line, and the orientation of the two lines will be similar. This is true even when the image line is fragmented. In this example, $[x_{\text{mid}}, y_{\text{mid}}]$ is closest to $[\bar{x}_7, \bar{y}_7]$.

neighbors of l are expected to be found is defined to be $r'(l, l') = (\|l'\| r(l)) / \|l\|$, which is just the neighborhood radius of l scaled by the same amount that l is itself scaled by the mapping. Assume a transformed model line and an image line are represented by the points $[\theta_1, x_1, y_1]$ and $[\theta_2, x_2, y_2]$, respectively, as described by Equations 3 and 4. When the orientation of the lines differ by $\pi/2$ radians, we want $|\theta_1 - \theta_2| = 1$; when the horizontal distance between the sample points equals the neighborhood radius of the image line, we want $|x_1 - x_2| = 1$; and, when the vertical distance between the sample points equals the neighborhood radius of the image line, we want $|y_1 - y_2| = 1$. The value $r_\theta = \pi/2$ satisfies the first normalization constraint. However, because image lines will have different neighborhood radii depending on which model line they correspond to, the later normalization constraints can't be satisfied by a constant scale factor, but they are satisfied for model and image lines of average length by

$$r_m = \frac{\left(\sum_{l \in \mathcal{I}} \|l\|\right) \left(\sum_{l \in \mathcal{M}} r(l)\right)}{|\mathcal{I}| \sum_{l \in \mathcal{M}} \|l\|}. \tag{6}$$

The terms in Equation 6 that sum over model lines represent sums over all lines in all models. Using this value for r_m has worked well in practice.

Finally, for any transformed model line l', to find the image line l'' that minimizes $d(l', l'')$

we simply query the nearest neighbor data structure (generated using points from Equation 3) with all of the points listed in Equation 4 and then use the distance of the closest one. Because the complexity of the nearest neighbor search is $\mathcal{O}(\log n)$, the use of multiple points to represent lines does not significantly slow down the algorithm.

7 Graduated Assignment for Lines

The final stage of the object recognition algorithm is to apply a pose refinement and verification algorithm to the few "best" approximate poses. We use the graduated assignment algorithm [11] for this purpose because it is efficient ($\mathcal{O}(mn)$ complexity for m model lines and n image lines), robust to occlusion and clutter, and doesn't make hard correspondence decisions until a locally optimal pose is found.

Given an approximate initial pose $T_0 = \{A_0, \mathbf{t}_0\}$, we wish to find a 2D affine transformation $T = \{A, \mathbf{t}\}$ that maximizes the number of matches between model and image lines such that the distance between matched lines does not exceed a threshold δ_{final}. For a transformation T and a line l, let us denote by $T(l)$ the transformation of l by T. We assume that our initial pose T_0 is accurate enough so that the pose refinement algorithm does not have to consider all possible correspondences between model lines l and image lines l' but only those correspondences where $d(T(l), l') \leq \delta_0$; here, $d()$ is the distance function defined in Section 6 and δ_0 is an initial distance threshold that allows any reasonably close correspondence. Limiting the correspondences in this way results in a significant increase in the speed of this step without affecting the final outcome. Let $\mathcal{I}' = \{l' \in \mathcal{I} \mid \exists l \in \mathcal{M} \wedge d(T_0(l), l') \leq \delta_0\}$ be the subset of image lines that are initially reasonably close to any transformed model line.

Given m model lines $\mathcal{M} = \{l_j, j = 1, \ldots, m\}$, n image lines $\mathcal{I}' = \{l'_k, k = 1, \ldots, n\}$, and an approximate model pose $T_0 = \{A_0, \mathbf{t}_0\}$, we wish to find the 2D affine transformation T and the $(m+1) \times (n+1)$ *match matrix* M that minimizes the objective function

$$E = \sum_{j=1}^{m} \sum_{k=1}^{n} M_{jk} \left(d(T(l_j), l'_k)^2 - \delta^2 \right). \tag{7}$$

M defines the correspondences between model lines and image lines; it has one row for each of the m model lines and one column for each of the n image lines. This matrix must satisfy

the constraint that each model line match at most one image line, and vice versa. By adding an extra row and column to M, *slack row* $m+1$ and *slack column* $n+1$, these constraints can be expresses as $M_{jk} \in \{0,1\}$ for $1 \leq j \leq m+1$ and $1 \leq k \leq n+1$, $\sum_{i=1}^{n+1} M_{ji} = 1$ for $1 \leq j \leq m$, and $\sum_{i=1}^{m+1} M_{ik} = 1$ for $1 \leq k \leq n$. A value of 1 in the slack column $n+1$ at row j indicates that the j^{th} model line does not match any image line. A value of 1 in the slack row $m+1$ at column k indicates that the k^{th} image line does not match any model line. The objective function E in Equation 7 is minimized by maximizing the number of correspondences $l_j \leftrightarrow l'_k$ where $d\left(T\left(l_j\right), l'_k\right) < \delta_{\text{final}}$.

Optimizing the objective function in Equation 7 as a function of M and T is difficult because it requires a minimization subject to the constraint that the match matrix be a zero-one matrix whose rows and columns each sum to one. A typical nonlinear constrained optimization problem minimizes an objective function on a feasible region that is defined by equality and inequality constraints. The zero-one constraint on the match matrix is impossible to express using equality and inequality constraints. The graduated assignment algorithm developed by Gold and Rangarajan ([10, 11]), however, can efficiently optimize our objective function subject to these constraints. This algorithm uses deterministic annealing to convert a discrete problem (for a binary match matrix) into a continuous one that is indexed by the control parameter β. The parameter β ($\beta > 0$) determines the uncertainty of the match matrix, and hence the amount of smoothing implicitly applied to the objective function. The match matrix minimizing the objective function is tracked as this control parameter is slowly adjusted to force the continuous match matrix closer and closer to a binary match matrix. 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.

The objective function is minimized by first computing the variables M_{jk} that minimizes E assuming that the transformation T is fixed, and then computing the transformation T that minimizes E assuming that the M_{jk} are fixed. This process is repeated until these estimates

converge. For a fixed transformation T, the continuous match matrix M is initialized by

$$M_{jk}^0 = \begin{cases} 1 & \text{if } j = m+1 \text{ or } k = n+1 \\ \exp\left(-\beta\left(d\left(T\left(l_j\right), l'_k\right)^2 - \delta^2\right)\right) & \text{otherwise,} \end{cases} \tag{8}$$

where δ varies between δ_0 at the start of the optimization and δ_{final} at the end. Note that δ determines how distant two lines can be before the correspondence becomes undesirable:

$$\begin{aligned} M_{jk}^0 &< 1 \text{ when } d\left(T\left(l_j\right), l'_k\right)^2 > \delta^2, \\ M_{jk}^0 &= 1 \text{ when } d\left(T\left(l_j\right), l'_k\right)^2 = \delta^2, \\ M_{jk}^0 &> 1 \text{ when } d\left(T\left(l_j\right), l'_k\right)^2 < \delta^2. \end{aligned}$$

So, for example, when $d\left(T\left(l_j\right), l'_k\right)^2 > \delta^2$, M_{jk}^0 will be given a value less than the initial slack values of 1 for row j and column k, thus initially making assignment to slack preferred over the assignment of model line j to image line k. Next, the match constraints are enforced by applying to M^0 the Sinkhorn algorithm [19] of repeated row and column normalizations:

repeat

$M_{jk}^{i+1} = M_{jk}^i / \Sigma_{s=1}^{n+1} M_{js}^i$, $1 \le j \le m$, $1 \le k \le n+1$.

$M_{jk}^{i+1} = M_{jk}^{i+1} / \Sigma_{s=1}^{m+1} M_{sk}^i$, $1 \le j \le m+1$, $1 \le k \le n$.

until $\left\| M^{i+1} - M^i \right\|$ **small**

Sinkhorn showed that when each row and column of a square matrix is normalized several times by the sum of the elements of that row or column, respectively (alternating between row and column normalizations), the resulting matrix converges to one that has positive elements with all rows and columns summing to 1, in other words, a probability distribution. However, this is only approximate for a non-square matrix such as ours: either the rows or the columns will sum to one, but not both. When β is small, *all* elements of M^0 will be close to the neutral value of 1; this represents a high degree of uncertainty in the correspondences. As β increases (and presumably the accuracy of the pose as well), the uncertainty in the correspondences decreases and the elements of M^0 move towards the values of 0 or ∞. Thus, the match matrix starts off approximating a continuous probability distribution when β is small, and ends up as a binary correspondence matrix when β is large. Appendix A describes changes that we have

made to the Sinkhorn algorithm as described above that often result in improved convergence of the graduated assignment algorithm to the local optima.

We also need to compute the affine transformation T that minimizes the objective function E assuming that the continuous-valued match matrix M is held constant. This is difficult to do directly because of the complex nonlinear relation between T and the nearest neighbor distance function d. Instead, we replace d with a new distance, d', whose square is the sum of the squared distances of the endpoints of an image line to the infinitely extended model line. For a model line l_j and an image line l'_k, the new squared distance between $T(l_j)$ and l'_k is

$$d'\left(T\left(l_j\right), l'_k\right)^2 = \left[T\left(\mathbf{n}_j\right)^{\mathsf{T}}\left(p'_{k1} - T\left(p_{j1}\right)\right)\right]^2 + \left[T\left(\mathbf{n}_j\right)^{\mathsf{T}}\left(p'_{k2} - T\left(p_{j1}\right)\right)\right]^2$$

where p'_{k1} and p'_{k2} are the two endpoints of l'_k, and where $T\left(\mathbf{n}_j\right)$, $T\left(p_{j1}\right)$, and $T\left(p_{j2}\right)$ denote the normal and two endpoints of $T\left(l_j\right)$, respectively. The new objective function is

$$E' = \sum_{j=1}^{m}\sum_{k=1}^{n} M_{jk}\left(d'\left(T\left(l_j\right), l'_k\right)^2 - \delta^2\right).$$

In general, the transformation T that minimizes E' is not guaranteed to minimize E. In practice, however, because three line correspondences define a 2D affine transformation, one would expect E and E' to have approximately the same minimizers whenever the model has three or more lines in a non-degenerate configuration. Since the expression for $d'\left(T\left(l_j\right), l'_k\right)^2$ involves rotating a vector and transforming a point, it is actually easier to reverse the roles of l_j and l'_k and minimize E' by computing the inverse transformation $T' = \{A', \mathbf{t}'\}$ that maps image lines into the frame of reference of the model. This way, the model normal vectors are constants and T' is applied only to image points. Then, T is computed as the inverse of T'. The objective function that is minimized to determine T' is

$$E'' = \sum_{j=1}^{m}\sum_{k=1}^{n} M_{jk}\sum_{i=1}^{2}\left(\left(\mathbf{n}_j^{\mathsf{T}}\left(A'p'_{ki} + \mathbf{t}' - p_{j1}\right)\right)^2 - \delta^2\right) \tag{9}$$

where $\mathbf{n}_j = \left[x_{n_j}, y_{n_j}\right]$ is the unit normal vector of model line l_j, $p_{j1} = [x_{j1}, y_{j1}]$ is one endpoint of model line l_j, and $p'_{ki} = [x'_{ki}, y'_{ki}]$ is the i^{th} endpoint $(i = 1, 2)$ of image line l'_k. The

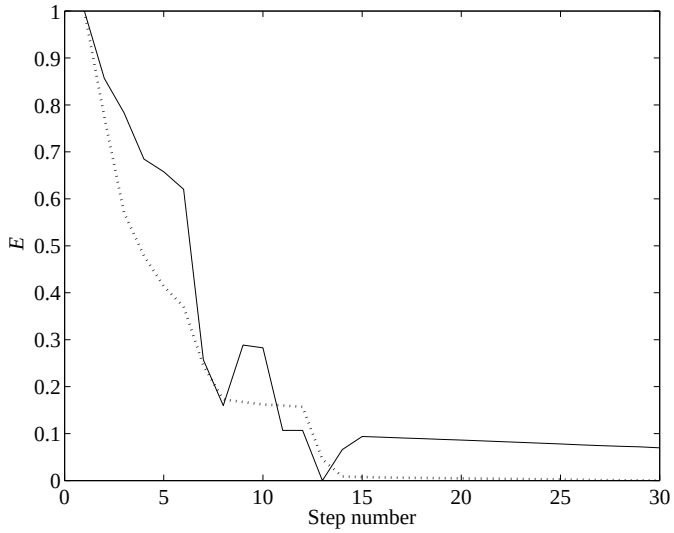


Figure 8: Comparison of the two objective functions for a typical minimization by the graduated assignment algorithm. The solid line is E, which uses the Euclidean distances to the nearest neighbor, and the dotted line is E', which uses the sum of the distances of the image line endpoints to the infinitely extended model lines. The values of E and E' generally decrease during the optimization process, but they can also rise due to changes in the assignment variables M_{jk}.

transformation

$$T' = \{A', \mathbf{t}'\} = \left\{ \begin{bmatrix} a' & b' \\ c' & d' \end{bmatrix}, \begin{bmatrix} t'_x \\ t'_y \end{bmatrix} \right\}$$

that minimizes Equation 9 can be found by solving the system of six equations

$$\begin{aligned} &\partial E''/\partial a' = 0, \quad \partial E''/\partial b' = 0, \quad \partial E''/\partial c' = 0, \\ &\partial E''/\partial d' = 0, \quad \partial E''/\partial t'_x = 0, \quad \partial E''/\partial t'_y = 0 \end{aligned} \tag{10}$$

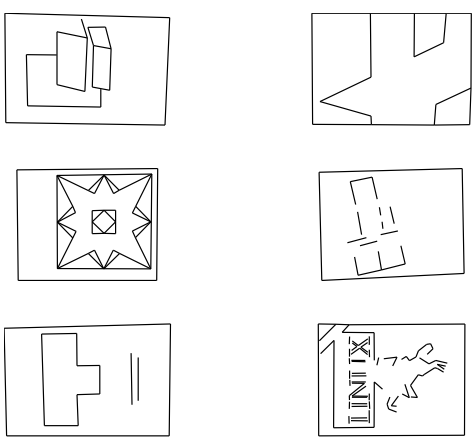
for a', b', c', d', t'_x, and t'_y. The solution to this system of equations is given in Appendix B.

Figure 8 compares the values of E and E' over a typical application of the graduated assignment algorithm. Pseudocode for the pose refinement and verification algorithm is shown in Figure 9. Figure 10 shows an example of how graduated assignment transforms an initial approximate pose into a more accurate pose.

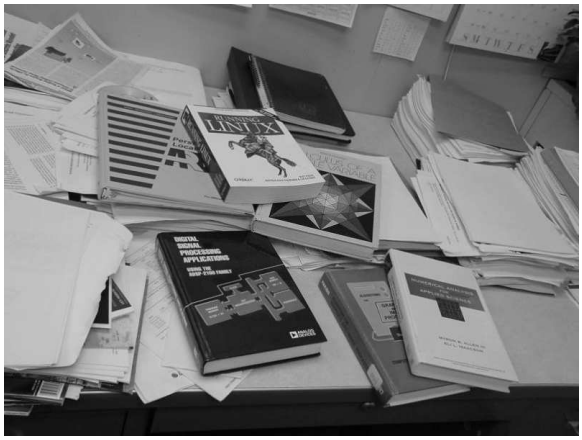
initialize: $T = T_0$, $\beta = \beta_0$, $\delta = \delta_0$, $\epsilon = \infty$, $k = 0$, *maxsteps* = 30.
while $k <$ *maxsteps* **and** $\epsilon > \epsilon_{\textbf{halt}}$, **do**
  Initialize M^0 according to Equation 8.
  Apply Sinkhorn's algorithm to M^0 to produce M.
  Compute T'_k via Equation 10.
  Compute T_k as the inverse of T'_k.
  $\epsilon = \max |T_k - T_{k-1}|$.
  $k = k + 1$.
  $\delta = \delta - (\delta_{\text{final}} - \delta_0) / maxsteps$.
  $\beta = \beta_{\text{update}} \times \beta$.
end while

Figure 9: The pose refinement algorithm.

Figure 10: Pose refinement using the graduated assignment algorithm: initial pose (left) and final pose after applying the algorithm (right).



(a) Models of books.



(b) Test image.

(c) 731 detected lines.

(d) Recognized books.

Figure 11: Five books, some partially occluded, are recognized in a cluttered environment.

8 Experiments

To validate our approach, we recognized partially occluded 2D and 3D objects in cluttered environments under a wide variety of viewpoints. All images were acquired at a resolution of 800×600 pixels. 400 to 800 lines were typically detected in an image, and each model had between 20 and 80 lines. First, we used books to test the recognition of planar objects. Figure 11 illustrates recognition results when our algorithm is applied to an image of a pile of books. For all but one of the five recognized books, the pose hypothesis leading to correct recognition was found in the top 10 hypotheses of the sorted hypothesis list. One book

("Linux", shown in the lower right of Figure 11a) was not found until the 24^{th} pose hypothesis. This book might be more difficult for our approach to recognize because a large part of its cover depicts a horse with curved borders, for which detected lines were inconsistent in images acquired from different viewpoints.

The performance of our algorithm depends on how reliably it can move to the top of the sorted hypothesis list those pose hypotheses associated with correct correspondences. To evaluate this, we estimate $P_\theta(k)$, the probability that one of the first k sorted pose hypotheses for a model leads to a correct recognition when the viewpoint of the recognized object and the viewpoint used to generate its model differ by an angle of θ, assuming that an instance of the model does in fact appear in the image. Knowing $P_\theta(k)$ allows one to determine how many pose hypotheses should be examined by the pose refinement process before restarting with a new model, either of a new object or of the same object but from a different viewpoint. Because $P_\theta(k)$ is highly dependent on the amount and type of clutter and occlusion in an image, and because the level of clutter and occlusion present in our test was held fixed, $P_\theta(k)$ should be interpreted loosely. The six books shown in Figure 11a were used to perform this experiment. All six books were placed flat on a table along with a number of other objects for clutter. Each book in turn was moved to the center of the table and then rotated on the plane of the table to 8 different orientations, where each orientation was separated by approximately $45°$. For each of these orientations, the camera was positioned at angles $0°$, $10°$, $20°$, $30°$, $40°$, $50°$, $60°$, and $70°$ relative to the normal of the table ($0°$ is directly overhead) and at a fixed distance from the center book, and then an image was acquired. The center books were unoccluded in these experiments. A separate image was also acquired of each book in an arbitrary orientation with the camera in the $0°$ position; these later images were used to generate the book models. Figure 12 shows an image of the books on this table for the camera in the $50°$ position. We then applied our algorithm to each model and image pair and determined the position in the sorted hypothesis list of the first hypothesis that allowed the object to be recognized. Up to 100 hypotheses were examined for each model and image pair. The estimated values of $P_\theta(k)$ are shown in Figure 13. From this we see that, for planar objects whose orientations differ by up to $60°$ from the modeled orientation, a probability of correct recognition of 0.8 can be achieved by examining the first 30 pose hypotheses. By examining just the top four pose hypotheses, we

Figure 12: Image of a table of books taken by a camera tilted 50° from vertical.

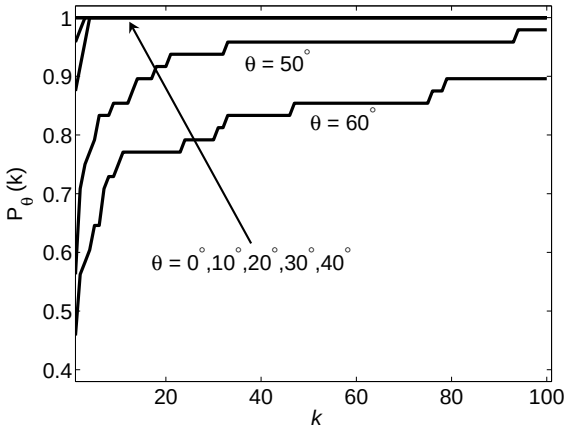


Figure 13: $P_\theta\,(k)$ is the probability that one of the first k sorted pose hypotheses for a model leads to a correct recognition for that model. θ is the difference in viewpoint elevation between the model and the object. For $\theta \leq 40^\circ$, one of the four highest ranked pose hypotheses always leads to correct recognition. The curves for $\theta = 0^\circ$ thru 40° are superposed for $k \geq 4$.

can achieve a probability of correct recognition of 1.0 for objects whose orientations differ by up to 40° from the modeled orientations. Thus, a good strategy would be to apply the algorithm using a set of models for each object generated for every 40° degree change in viewpoint; in this case, it would be sufficient to represent planar objects by five models in order to recognize all orientations of up to 80° from the normal.

Finally, we applied our algorithm to three 3D objects. We acquired 17 images of each object, where the objects were rotated by 2.5° between successive images. The first image of each object was used to represent the object, and from this a 2D model was generated by identifying the object edges in that image. Examining only the top 10 sorted pose hypotheses, all three objects were successfully recognized from all viewpoints that differed by up to 25° from the modeled viewpoint. Two of the objects (the monitor and hole punch) were also recognized at 30° away from the modeled viewpoints. Figure 14 shows the object models, the images, the detected lines, and the final poses of the recognized objects for the most distant viewpoints that each was recognized. The range of recognizable object orientations could have been be extended somewhat by examining more pose hypotheses, but at some point it becomes more cost effective to add a new model for a different viewpoint.

The minimum number of 2D models needed to represent a 2D or 3D object can be determined from the maximum difference between the object's orientation and a modeled orientation that still allows the object to be recognized from that 2D model. Let this maximum difference between object and model orientation be denoted by Θ. To be conservative, based on results described above, we take $\Theta = 40^\circ$ for 2D objects, and $\Theta = 20^\circ$ for 3D objects. The minimum number of 2D models needed to represent an object is then determined by counting the number of identical right-circular cones of angle Θ that are needed to fully enclose the upper hemisphere (for the case of 2D objects) or the entire sphere (for the case of 3D objects) when the vertices of the cones are placed at the sphere's center. (The angle of a right circular cone is the angle around the vertex of the cone between the cone's axis and the conic surface.) One can determine that six cones of angle 40° fully enclose the upper hemisphere while 44 cones of angle 20° fully enclose the entire sphere. Thus, recognizing 2D objects from any viewpoint above the plane of the object requires at most six 2D models, while recognizing 3D objects from any viewpoint (above or below the ground plane) requires at most 44 2D models.

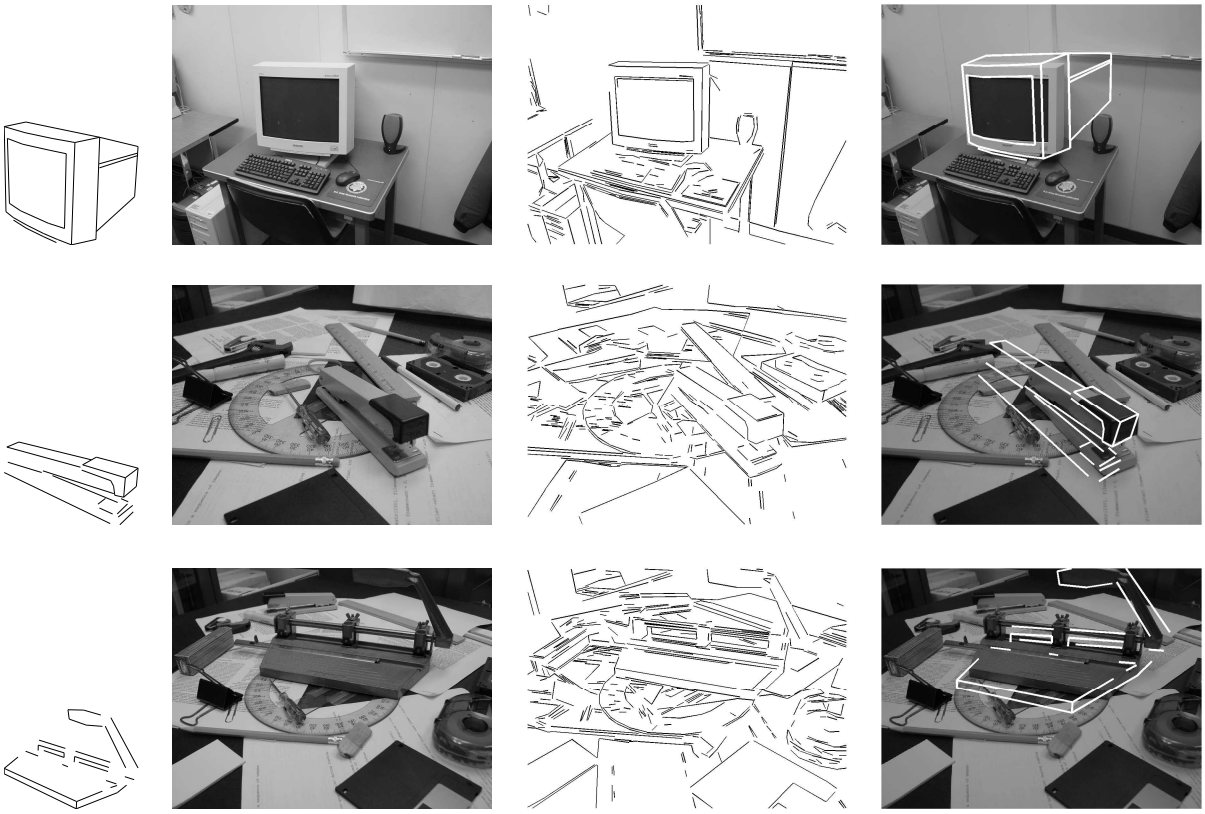


Figure 14: Recognition of 3D objects from viewpoint-dependent 2D models: computer monitor (top row), stapler (middle row), and hole punch (bottom row). Shown in each row, from left to right, is the 2D object model, original image, image lines, and model of recognized object overlaid on the original image. The modeled view of each object differs from the test view by 20° to 30°.

From the above experiments, it is apparent that only a small number of pose hypotheses need to be examined by the pose refinement algorithm in order to reliably recognize objects. We use this to determine the overall runtime complexity of our algorithm. Assume that we have q models, each containing m lines, and that the image contains n lines. Initialization of the nearest neighbor data structure and identification of corners can be performed in $\mathcal{O}(n \log n)$ time. For each model, we generate $\mathcal{O}(mn)$ pose hypotheses. The neighborhood similarity of each of these can be evaluated in $\mathcal{O}(\log n)$ time. The pose hypotheses can be sorted in time $\mathcal{O}(mn \log(mn))$. The pose refinement algorithm requires $\mathcal{O}(mn)$ time. Thus, the overall runtime complexity of our algorithm is $\mathcal{O}(qmn \log(mn))$.

9 Conclusions

We have presented an efficient approach to recognizing partially occluded objects in cluttered environments. Our approach improves on previous approaches by making use of information available in one or two line correspondences to compute approximate object poses. Only a few model lines need to be unfragmented in an image in order for our approach to be successful; this condition is easily satisfied in most environments. The use of one or two line correspondences to compute an objects pose allows for a large reduction in the dimensionality of the space that must be searched in order to find a correct pose. We then developed an efficiently computed measure of the similarity of two line neighborhoods that is largely unaffected by clutter and occlusion. This provides a way to sort the approximate model poses so that only a small number need to be examined by more time consuming algorithms. Experiments show that a single view of an object is sufficient to build a model that will allow recognition of that object over a wide range of viewpoints.

A Modifications to the Sinkhorn Algorithm

Sinkhorn's original algorithm [19] treats all rows and columns identically. The modified Sinkhorn algorithm discussed in Section 7 must treat the slack row and column differently from other rows and columns: the slack values are not normalized with respect to other slack values, only with respect to the nonslack values. This is necessary in order to allow multi-

ple image lines to be identified as clutter and to allow multiple model lines to be identified as occluded. A problem with this modified algorithm is the following. Suppose that a nonslack value is a maximum in both its row and column. After normalizing that row, it is possible that this previously maximal value is now less than the slack value for that column. The same sort of thing can happen when columns are normalized. Intuitively, this behavior is undesirable: nonslack values that start off maximal *in both their row and column* should remain maximal in their row and column throughout the Sinkhorn iteration. The purpose of Sinkhorn normalization is not to shift assignment weights around, but only to normalize the assignments so that they approximate a probability distribution. A secondary problem with the Sinkhorn algorithm is that the order of normalization (row first or column first) can have a significant effect on the final normalization, especially when there is potential for "weight shifting" as describe above.

To minimize weight shifting, after performing row normalizations, the values in the slack row are set so that their ratio to the nonslack value in each column which was previously maximum is the same as this ratio was prior to row normalization. A similar thing is done after column normalizations. In addition, to eliminate the effect of normalization order, rows and columns are normalized independently on each step of the iteration and then the two normalized matrices are combined into one. The pseudocode for this new Sinkhorn algorithm is shown in Figure 15.

B Solving for the Affine Transformation

The affine transformation that minimizes Equation 9 is obtained by solving the system given in Equation 10. Expanding Equation 10, we obtain the linear system $A\mathbf{x} = \mathbf{b}$ where $\mathbf{x} = \left(a'\ b'\ c'\ d'\ t'_x\ t'_y\right)^{\mathsf{T}}$, A is the 6×6 symmetric matrix

$$A = \sum_{j=1}^{m} \sum_{k=1}^{n} M_{jk} \begin{pmatrix} x_{n_j}^2 \left(x_{k1}'^2 + x_{k2}'^2\right) & x_{n_j}^2 \left(x'_{k1} y'_{k1} + x'_{k2} y'_{k2}\right) & x_{n_j} y_{n_j} \left(x_{k1}'^2 + x_{k2}'^2\right) \\ x_{n_j}^2 \left(x'_{k1} y'_{k1} + x'_{k2} y'_{k2}\right) & x_{n_j}^2 \left(y_{k1}'^2 + y_{k2}'^2\right) & x_{n_j} y_{n_j} \left(x'_{k1} y'_{k1} + x'_{k2} y'_{k2}\right) \\ x_{n_j} y_{n_j} \left(x_{k1}'^2 + x_{k2}'^2\right) & x_{n_j} y_{n_j} \left(x'_{k1} y'_{k1} + x'_{k2} y'_{k2}\right) & y_{n_j}^2 \left(x_{k1}'^2 + x_{k2}'^2\right) \\ x_{n_j} y_{n_j} \left(x'_{k1} y'_{k1} + x'_{k2} y'_{k2}\right) & x_{n_j} y_{n_j} \left(y_{k1}'^2 + y_{k2}'^2\right) & y_{n_j}^2 \left(x'_{k1} y'_{k1} + x'_{k2} y'_{k2}\right) \\ x_{n_j}^2 \left(x'_{k1} + x'_{k2}\right) & x_{n_j}^2 \left(y'_{k1} + y'_{k2}\right) & x_{n_j} y_{n_j} \left(x'_{k1} + x'_{k2}\right) \\ x_{n_j} y_{n_j} \left(x'_{k1} + x'_{k2}\right) & x_{n_j} y_{n_j} \left(y'_{k1} + y'_{k2}\right) & y_{n_j}^2 \left(x'_{k1} + x'_{k2}\right) \end{pmatrix} \cdots$$

$i = 0$

$\mathcal{R} = \left\{ \left(r, c, \frac{M^0_{r,n+1}}{M^0_{rc}}, \frac{M^0_{m+1,c}}{M^0_{rc}} \right) \mid M^0_{rc} > M^0_{jc} \text{ and } M^0_{rc} > M^0_{rk} \text{ for all } j \neq r \text{ and } k \neq c \right\}$

repeat

$M'_{jk} = M^i_{jk} / \Sigma_{s=1}^{n+1} M^i_{js}$, $1 \leq j \leq m$, $1 \leq k \leq n+1$. *// Normalize rows*

for each $(r, c, \lambda, \mu) \in \mathcal{R}$ *// Adjust slack row*

$M'_{m+1,c} = \mu M'_{rc}$

end for

$M''_{jk} = M^i_{jk} / \Sigma_{s=1}^{m+1} M^i_{sk}$, $1 \leq j \leq m+1$, $1 \leq k \leq n$. *// Normalize columns*

for each $(r, c, \lambda, \mu) \in \mathcal{R}$ *// Adjust slack column*

$M''_{r,n+1} = \lambda M''_{rc}$

end for

$i = i + 1$

$M^i_{jk} = \left(M'_{jk} + M''_{jk} \right) / 2$, $1 \leq j \leq m+1$, $1 \leq k \leq n+1$

until $\left\| M^i - M^{i-1} \right\|$ **small**

Figure 15: The new Sinkhorn algorithm.

$$\begin{pmatrix} x_{n_j}y_{n_j}\left(x'_{k1}y'_{k1}+x'_{k2}y'_{k2}\right) & x^2_{n_j}\left(x'_{k1}+x'_{k2}\right) & x_{n_j}y_{n_j}\left(x'_{k1}+x'_{k2}\right) \\ x_{n_j}y_{n_j}\left(y'^2_{k1}+y'^2_{k2}\right) & x^2_{n_j}\left(y'_{k1}+y'_{k2}\right) & x_{n_j}y_{n_j}\left(y'_{k1}+y'_{k2}\right) \\ y^2_{n_j}\left(x'_{k1}y'_{k1}+x'_{k2}y'_{k2}\right) & x_{n_j}y_{n_j}\left(x'_{k1}+x'_{k2}\right) & y^2_{n_j}\left(x'_{k1}+x'_{k2}\right) \\ y^2_{n_j}\left(y'^2_{k1}+y'^2_{k2}\right) & x_{n_j}y_{n_j}\left(y'_{k1}+y'_{k2}\right) & y^2_{n_j}\left(y'_{k1}+y'_{k2}\right) \\ x_{n_j}y_{n_j}\left(y'_{k1}+y'_{k2}\right) & 2x'^2_{n_j} & 2x'_{n_j}y'_{n_j} \\ y^2_{n_j}\left(y'_{k1}+y'_{k2}\right) & 2x'_{n_j}y'_{n_j} & 2y'^2_{n_j} \end{pmatrix},$$

and $\mathbf{b}$ is the column 6-vector given by

$$\mathbf{b}=\sum_{j=1}^{m}\sum_{k=1}^{n}M_{jk}\left(x_{n_j}x_{j1}+y_{n_j}y_{j1}\right)\begin{pmatrix} x_{n_j}\left(x'_{k1}+x'_{k2}\right) \\ x_{n_j}\left(y'_{k1}+y'_{k2}\right) \\ y_{n_j}\left(x'_{k1}+x'_{k2}\right) \\ y_{n_j}\left(y'_{k1}+y'_{k2}\right) \\ 2x_{n_j} \\ 2y_{n_j} \end{pmatrix}.$$

The unknown $\mathbf{x}$ is easily found using standard linear methods.

References

[1] N. Ayache and O.D. Faugeras, "HYPER: A New Approach for the Recognition and Positioning of Two-Dimensional Objects," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 8, no. 1, pp. 44–54, January 1986.

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

[3] 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.

[4] O. Carmichael and M. Hebert, "Shape-Based Recognition of Wiry Objects," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26 , no. 12 , pp. 1537–1552, December 2004.

[5] P. David, D. DeMenthon, R. Duraiswami, and H. Samet, “Simultaneous Pose and Correspondence Determination using Line Features,” *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2, pp. II-424–II-431, June 2003.

[6] P. David, D. DeMenthon, R. Duraiswami, and H. Samet, “SoftPOSIT: Simultaneous Pose and Correspondence Determination,” *Int. Journal of Computer Vision*, vol. 49, no. 3, pp. 259–284, 2004.

[7] J. Denton and J.R. Beveridge, “Two Dimensional Projective Point Matching,” *5th IEEE Southwest Symposium on Image Analysis and Interpretation*, pp. 77–81, April 2002.

[8] R.O. Duda and P.E. Hart, “Use of the Hough Transform To Detect Lines and Curves in Pictures," *Communications of the ACM,* vol. 15, no. 1, pp. 11–15, 1972.

[9] 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.

[10] S. Gold and A. Rangarajan, “A Graduated Assignment Algorithm for Graph Matching,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, no. 4, pp. 377–388, 1996.

[11] 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, pp. 1019-1031, August 1998.

[12] C.G. Harris and M.J. Stephens, “A Combined Corner and Edge Detector,” *Proc. Fourth Alvey Vision Conference*, Manchester, pp. 147-151, 1988.

[13] P. D. Kovesi, “MATLAB Functions for Computer Vision and Image Analysis,” School of Computer Science & Software Engineering, The University of Western Australia. Available from: <http://www.csse.uwa.edu.au/~pk/research/matlabfns/>.

[14] D.G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *Int. Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[15] C. Merkwirth, U. Parlitz, I. Wedekind, and W. Lauterborn, “TSTOOL User Manual,” University of Göttingen, http://www.physik3.gwdg.de/tstool/index.html.

[16] K. Mikolajczyk, A. Zisserman, and C. Schmid, “Shape Recognition with Edge-Based Features,” *Proc. British Machine Vision Conference*, September 2003.

[17] L. Roberts, “Machine Perception of Three-Dimensional Solids,” *Optical and Electrooptical Information Processing*, J. T. Tipett, Ed., M.I.T. Press, Cambridge, MA, pp. 159-197, 1965.

[18] C. Schmid and R. Mohr, “Local Grayvalue Invariants for Image Retrieval,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19 , no. 5 , pp. 530–535, May 1997.

[19] R. Sinkhorn. “A Relationship between Arbitrary Positive Matrices and Doubly Stochastic Matrices,” *Annals Mathematical Statistics*, vol. 35, no. 2, pp. 876–879, 1964.

[20] S. Ullman, “Aligning Pictorial Descriptions: An Approach to Object Recognition,” *Cognition*, vol. 32, no. 3, pp. 193–254, 1989.