ABSTRACT

Title of dissertation: DETERMINISTIC ANNEALING FOR CORRESPONDENCE, POSE, AND RECOGNITION

Philip J. David, Doctor of Philosophy, 2006

Dissertation directed by: Dr. Daniel DeMenthon
Department of Computer Science

The problem of determining the pose - the position and orientation - of an object given a model and an image of that object is a fundamental problem in computer vision. Applications include object recognition, object tracking, site inspection and updating, and autonomous navigation when scene models are available. The pose of an object is readily determined given a few correspondences between features in the image and features in the model. Conversely, corresponding model and image features can easily be determined if the pose of the object is known. However, when neither the pose nor the correspondences are known, the problem of determining either is difficult due to the fact that a small change in an object's pose can result in a large change in its appearance. Most existing techniques approach this as a combinatorial optimization problem in which the space of model-to-image feature correspondence is searched in order to find object poses that are supported by large numbers of image features. These approaches, however, are only practical when the level of clutter and occlusion in the image is small, which is often not the case in real-world environments.

This dissertation presents new algorithms that simultaneously determine the pose and feature correspondences of 2D and 3D objects from images containing large amounts of clutter and occlusion. Objects are modeled as sets of 2D or 3D points or line segments, and image features consist of either points or line segments. In each of the algorithms presented, deterministic annealing is used to convert a discrete combinatorial optimization problem into a continuous one that is indexed by a control parameter. 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. These algorithms perform well in experiments involving highly cluttered synthetic and real imagery.

DETERMINISTIC ANNEALING FOR CORRESPONDENCE, POSE, AND RECOGNITION

by

Philip J. David

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2006

Advisory Committee:
Professor Larry Davis, Chair
Professor Rama Chellappa
Dr. Daniel DeMenthon, Advisor
Professor Ramani Duraiswami
Professor David Jacobs
Professor Benjamin Kedem

ACKNOWLEDGMENTS

I owe special thanks to Daniel DeMenthon, my Ph.D. advisor. His guidance and patience over the years have made this work possible. He provided encouragement, many good ideas, and was a pleasure to work with.

I have been supported financially by the Army Research Laboratory. It is here that I have been employed for the duration of my Ph.D. research. I thank Larry Tokarcik, Barbara Broom, Thomas Mills, Jay Gowens, and all of Army Research Laboratory management for believing in me and allowing me time away from my normal job to perform this research.

I thank my wife, Andrea, and our three children, Collin, Brian, and Michelle, for their constant love and encouragement which I have relied on throughout this time. It is to them that I dedicate this work. Finally, I am forever indebted to my parents for their love, patience, and encouragement.

Various parts of this dissertation have been published elsewhere [28, 29, 30, 31, 32, 33, 34, 39].

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ALGORITHMS

Chapter 1

Introduction

1.1 Introduction

A central problem in computer vision, and one that has been studied for at least 40 years [126], is that of determining the position (translation) and orientation (rotation) of a 3D object given a single image of that object. The position and orientation of the object with respect to the camera's coordinate system is known as the object's *pose*. A 3D model of the object is assumed to be available for use in this task. The model consists of a set of precisely known 3D features which, in the work described here, are 3D points and 3D line segments. It is also assumed that the projections of some of these 3D features can be located in an image of the object. When the correspondences between the model features and image features are known, algorithms for solving the pose estimation problem are relatively straightforward and efficient. The problem of determining an object's pose when correspondences between model features and image features are not known is much more difficult, and efficient solutions remain elusive. This later problem is also known as the *model-to-image registration problem* or the *simultaneous pose and correspondence problem*. This dissertation presents new algorithms for solving the simultaneous pose and correspondence problem, and evaluates these algorithms using real and simulated imagery.

There are many vision applications requiring automatic model-to-image registration. Some of these applications include the following. *Object recognition* [142] is the process of identifying the image of an object with a particular model from a database of models and determining the location in space of the object. This is a basic capability needed by systems to intelligently interact with their environments. Such systems may be found assembling parts in a factory, assisting visually impaired people function in a world where much information is transmitted visually, or improving the safety and effectiveness of soldiers by automatically detecting targets on the battlefield. In *object tracking* [104], the 3D position of an object is tracked throughout a sequence of images, allowing dynamic interaction with the object. Applications requiring object tracking include automotive parts assembly, autonomous docking of vehicles, and augmented reality in which a participant's head must be tracked in order to provide a virtual environment that moves in sync with the participant's motion. In *autonomous navigation* [47, 143], a robot is required to move about an environment using a map of that environment to guide its movements. Accurate localization is required for safe movement of the robot, and having an accurate map is required to obtain accurate localization. *Site inspection and map building* [43] is the process of comparing images of an environment to an existing map of that environment, and then updating the map based on these images. Determining the correspondences between model and image features, and the poses of objects, is a key step in many approaches for solving the above problems.

A model of an object is *registered* to an image of that object by determining the pose of the model that allows the projections of the model features to align

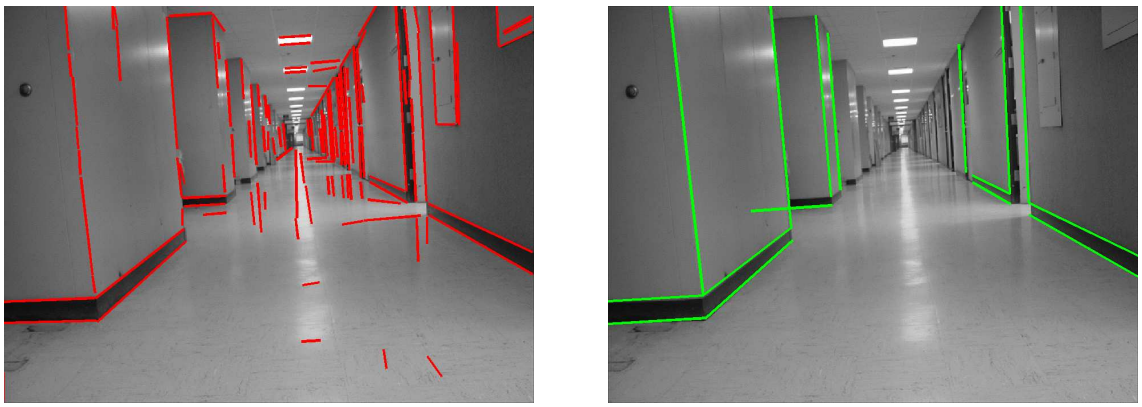


Figure 1.1: The left image shows straight lines automatically detected in an image of a hallway by image processing algorithms. The right image shows a partial model of the hallway projected into the image using a pose that aligns the image features with projected model features.

with features detected in the image. Model features are projected into an image using a mathematical model of the camera that produced the image along with an estimate of the camera's position relative to the scene object. Features in the image are located with some accuracy that depends on the type of image feature and on the image processing algorithms used to detect them. Not every model feature will have a corresponding feature in the image: some model features may be occluded by the object itself, some may be occluded by other objects in the scene, and some model features, although they may be visible to the camera, may not be detected in the image due to inadequate image processing algorithms. These same image processing algorithms may also produce "clutter" features, which are image features that don't correspond to any model features. Figure 1.1 shows an image of a hallway, the features detected in that image, and a 3D model projected onto the image using a pose that aligns the corresponding model and image features. A robot using vision to navigate this hallway would have to deal with data similar to

that shown in this figure. Although the presence of clutter and occlusion greatly complicates the model-to-image registration problem, the ability to deal with it is critical to any registration algorithm designed to work in the real world.

When one uses simple model and image features, such as points and lines, many image features may appear to match equally well to many object features, and some form of search will be required to determine the correct correspondences. One advantage of using simple features is that they generally can be found even in images of environments with little texture, such as the hallway image shown in Figure 1.1. The use of more descriptive image and model features, such as local color and texture features, can simplify the matching problem. However, these features are not useful in certain environments. Consequently, we assume throughout this work that our object models are 2D and 3D polyhedrals with features being 2D and 3D points and line segments, and that the image features consist of 2D points and line segments. 3D polyhedrals can be used to model many real-world objects of interest, however curves must be approximated using multiple lines.

As mentioned above, object recognition is the process of identifying (or labeling) the parts of an image with labels of particular models from a database of models, and determining the location in space of the identified models. A single image may contain multiple instances of one or more models from the database. In the work described here, we sometimes assume that there is exactly one model that we want to locate in the image, and that the identity of this model is known. An algorithm that can identify a single object in an image could be used for general purpose object recognition by cycling through all possible objects that may appear

in the image until one is recognized.

Automatic registration of 3D models to images 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:

The correspondence problem. The correspondence problem is a fundamental problem in computer vision for which there is as yet no reliable solution. Solving the correspondence problem consists of finding matching (corresponding) image features and model features, subject to certain constraints. The most common constraint is that sets of corresponding features must come from the same object. 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. This approach is typically used for *pose verification*, which attempts to determine how good a hypothesized pose is [61]. For models consisting of point or line features, the image of any model feature will look like the image of any other model feature; thus, when the object's pose is unknown, any model feature may match to any image feature. The correspondence problem is even more complicated, however, due to uncertainties associated with the image features. These uncertainties include errors in the locations of detected features (caused by noise in the image and imperfect image processing), occlusion of model features, and spurious features produced by scene clutter.

The pose problem. Solving the pose (or *exterior orientation* [73]) problem consists of finding the rotation and translation of the object with respect to

a camera coordinate system. Given matching model and image features, one can easily determine the pose that best aligns those matches. For three to five point matches, the pose can be found in closed-form by solving sets of polynomial equations [49, 66, 72, 154]. For six or more point matches, linear and approximate nonlinear methods are generally used [38, 48, 68, 73, 98].

The classic approach to solving these coupled problems is the hypothesize-and-test approach [59]. 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 [49] for the case that no information is available to constrain the correspondences of model to image points. When three correspondences are used to determine a pose, a high probability of success can be achieved by the RANSAC algorithm in $O(n^4m)$ time when there are n image points and m model points (see Appendix A for details).

Another reason that the model-to-image registration problem is difficult is that the dimension of the search space is very large. In general, the model-to-image correspondence function is a many-to-many mapping: one model feature can match to zero or more image features, and one image feature can match to zero or more model features. Both of these cases are illustrated in Figure 1.2 for the case of line features. When point features are used, one image point can match to multiple model points (which all lie on a single line of sight), but each model point

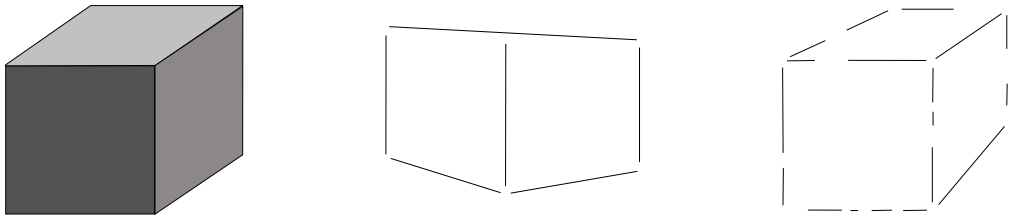


Figure 1.2: A model of a cube (left) and two possible images, (center and right), where lines have been extracted. The center image was produced by a pose that coincidentally aligned the images of two of the model lines; this illustrates the case of many-to-one model-to-image line correspondences. The right image was produced by a line detection algorithm that fragmented many of the image lines; this illustrates the case of one-to-many model-to-image correspondences.

can match at most one image point. So, in the case of point features, the model-to-image correspondence function is many-to-one. For a many-to-many correspondence function when m model features are to be matched to n image features, there are mn binary correspondence variables (with domain "match" or "no match") that must be determined. Although finite, the size of this correspondence space is still enormous for large m and n: 2^{mn}. 3D pose space has 6 degrees of freedom (3 for rotation and 3 for translation), but it is continuous and infinite. Thus, the dimension of the full search space is $mn+6$. Simplifying assumptions are often made to reduce the size of this search space. For example, by assuming that coincidental alignments of image lines, as shown in Figure 1.2, do not occur, the model-to-image correspondence function becomes one-to-many, and the dimension of the search space is reduce to $n+6$. Assuming that the correspondence function is one-to-one does not further reduce the dimension of the search space.

Most algorithms don't explicitly search this $mn+6$ or $n+6$ dimensional space, but instead assume that pose is determined by correspondences, or that correspon-

dences are determined by pose, and so search either an n-dimensional correspondence space (one-to-many model-to-image matches), an m-dimensional correspondences space (many-to-one model-to-image matches), or a 6-dimensional pose space. The correspondence space is m- or n-dimensional and discrete while the pose space is 6-dimensional and continuous. There are trade-offs associated with performing the search in ether of these spaces: search a large dimension finite space, or a smaller dimension continuous space. The previously best published runtime complexities for 3D model-to-image registration using geometric features are at least $\mathcal{O}(mn^3)$ [114] or $\mathcal{O}(m^2n^2)$ [11, 12]. With m and n large, these are still too complex for real-time operation on modern multi-giga-instruction per second processors.

More recently, the use of rich local feature descriptors has become popular as a way of reducing the number of correspondences that must be examined. Ideally, these features should be invariant to changes in illumination, scale, rotation in the image, and small changes in viewing direction. Researchers attempt to make these features highly distinctive, so that the number of possible correspondences to a database of features should be very small. The Harris corner detector [69] was one of the first distinctive image features used in pose and recognition algorithms; however, it is not stable to changes in image scale, so it performs poorly when models and images are of different scales. Schmid and Mohr [130] developed a rotationally invariant feature descriptor based on the Harris corner detector. Lowe [97] 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 later 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 building facades, 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. Furthermore, in some vision applications, such as those that use CAD models for navigation or recognition (see Figure 1.1, for example), the only model features available to match to images are geometric features. Thus, in many applications, only the relations between geometric features (such as points and edges) can be used for matching and object recognition, while the use of rich local feature descriptors will be of little value. The main goal of this research is therefore to develop efficient algorithms for model-to-image registration using geometric features.

1.2 Overview

This dissertation develops a number of new algorithms for solving the model-to-image registration problem. Point and line features are used for both the model and the image; the model features are 3D and the image features, which are assumed to have been generated by a perspective camera, are 2D. Our first new algorithm, which we call SoftPOSIT, integrates an iterative pose estimation technique called POSIT, developed by DeMenthon and Davis [38], and an iterative correspondence assignment technique called the graduated assignment algorithm, developed by Gold

and Rangarajan [55, 56], 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. Each of the components of the SoftPOSIT algorithm are briefly described below.

The POSIT (Pose from Orthography and Scaling with ITerations) algorithm computes an object's pose given a set of *corresponding* 2D image and 3D object points. The algorithm operates by assuming that the given perspective image points are close to the images that would be produced by a scaled orthographic camera, which is an approximation to a perspective camera. Under this assumption, the camera's pose can be determined by solving a simple system of linear equations. Because this assumption is only approximate, the computed pose will also be only approximate. However, by reestimating the scaled orthographic image points using the newly estimated pose and repeating the process, the accuracy of the pose can be improved. This process is repeated until the pose converges. The POSIT algorithm is fast and, as will be shown below, can be adapted to handle the case of *noncorresponding* image and object points.

The graduated assignment algorithm was originally applied to the problem of graph matching [55], where one seeks a *match matrix* that defines the correspondences between the nodes in two graphs. These two graphs are called the input and model graphs; the input graph is a possibly corrupted (with missing and spurious nodes and links) version of the model graph. The optimal match matrix is required

to minimizes an energy function that measures the compatibility of the corresponding nodes and links in the two graphs. This algorithm was later adapted to the problem of matching two 2D images or two 3D objects, where, in addition to a match matrix, a *linear* geometric transformation between the input and model was also sought [56]. SoftPOSIT is an extension of this algorithm to the more difficult problem of registration between a 3D object and its perspective image, that [56] did not address.

For an input graph with n nodes and a model graph with m nodes, the graduated assignment algorithm uses an $(n+1) \times (m+1)$ binary-valued *match matrix* M to represent the matches between nodes in these two graphs. A value of 1 at M_{ij} represents a match between input node i and model node j. The $(n+1)^{\text{st}}$ row and $(m+1)^{\text{st}}$ column of M are the *slack* row and column, respectively, which are used to account for missing and spurious nodes in the input graph. The process of matching the input to the model is based on three ideas: graduated nonconvexity, softassign, and sparsity.

The method of *graduated nonconvexity* (also known as *deterministic annealing* or the *continuation method*) transforms the discrete search space (for a binary match matrix) into a continuous search space. The continuous analog of the match matrix is called the *assignment matrix*. The continuous space is indexed by a control parameter that determines the fuzziness of the optimal assignment matrix, and hence the amount of smoothing implicitly applied to the energy function. The assignment matrix minimizing the energy function is tracked as this control parameter is slowly adjusted to force the continuous assignment matrix closer and closer to a binary

match matrix. The result is that many poor local minima can be avoided.

Softassign is the process that maps an approximate assignment matrix into a doubly stochastic assignment matrix. All rows and columns of a *doubly stochastic matrix* sum to one, so that the matrix represents a probability function for the correspondences between input and model nodes. At the end of the graduated nonconvexity process, when the assignment matrix must be a binary match matrix, this constraint forces each input node to match at most one model node, and each model node to match at most one input node. The Sinkhorn algorithm [135], which consists of alternating row and column normalizations, is used to ensure that the doubly stochastic constraint is satisfied. Previously, Rangarajan and Mjolsness [124] satisfied this constraint in a deterministic annealing framework by integrating the constraint equations into the energy function using Lagrange multipliers and then applying a gradient descent optimization algorithm. The Sinkhorn algorithm is a much more efficient alternative to using Lagrange multipliers, and as shown in [56], the two methods give identical assignment matrices.

The use of the graduated assignment algorithm for matching problems is motivated by the following. First, the softassign algorithm efficiently enforces the doubly stochastic constraint on the assignment matrix. Second, the deterministic annealing process allows poor local minima to be avoided. Third, the algorithm handles spurious and missing data through the use of slack rows and columns in the assignment matrix. Finally, the approach can be adapted to simultaneously solve for certain geometric transformations between the data and model.

The main advantage of the SoftPOSIT algorithm over other algorithms that

register perspective images to 3D models is its efficiency – it combines two algorithms that each have low computational complexity. Its other advantage is its use of deterministic annealing to avoid many local optima. Getting trapped in local optima is a problem that plagues all optimization algorithms. SoftPOSIT avoids many local optima 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.

Figure 1.3 shows an example computation of SoftPOSIT for a model with 15 points. Notice that it would be impossible to make hard correspondence decisions for the initial pose (frame 1) where the model image does not match the actual image at all. The deterministic annealing mechanism keeps all the options open until the two images are almost aligned.

The SoftPOSIT algorithm described above matches 2D image points and line segments to 3D model points and line segments. Theses simple geometric features provide no means to prune correspondences from a search because, when viewed in isolation, an image point or line segment can be a perfect image of any model point or line segment, respectively. However, when groups of nearby geometric features are viewed as a single entity, their appearances become much more distinctive. Shape contexts are feature vectors that describe the shapes of geometric entities in local regions of the image [107]; these can be generated even for sets of simple geometric point and line features. We apply shape contexts to line features for correspondence recovery and shape-based 2D object recognition. In doing so, the number of correspondences that are examined is drastically reduced, and a correspondence of a

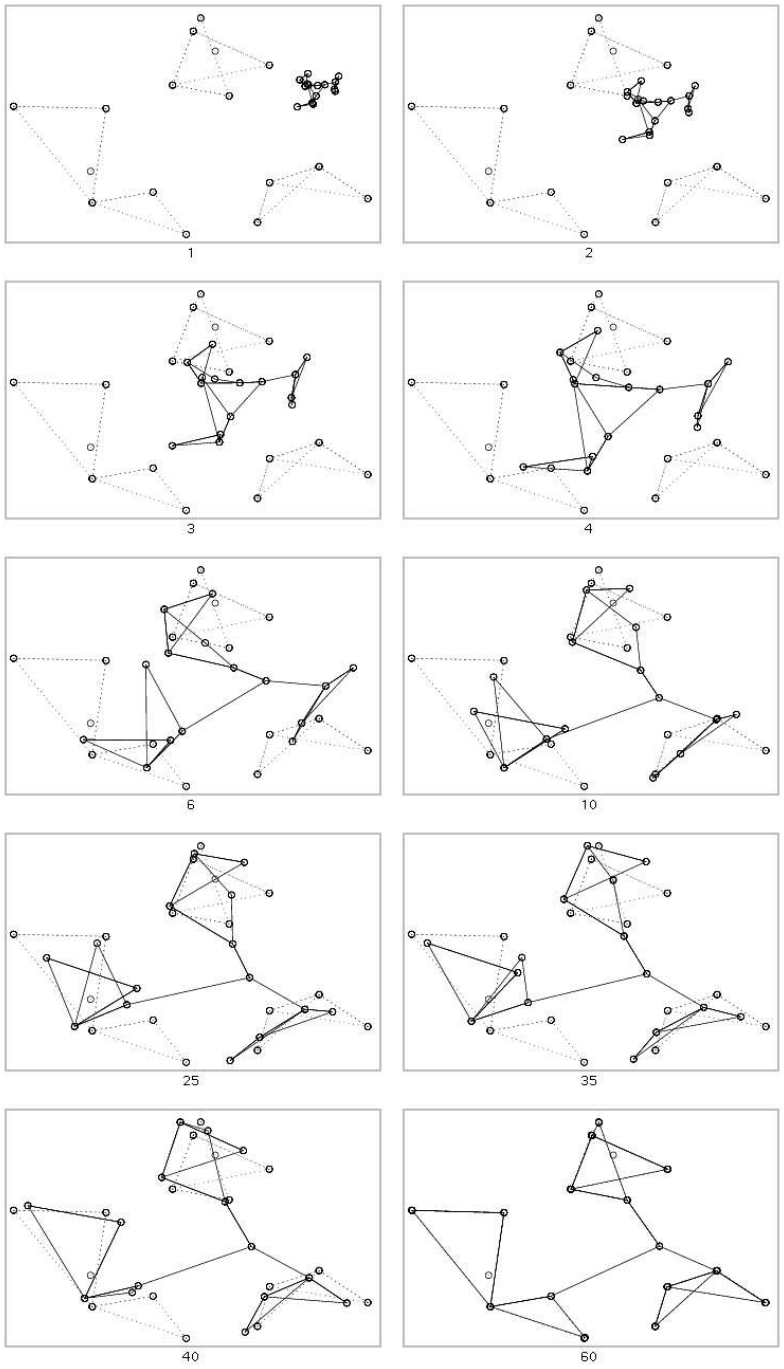


Figure 1.3: Evolution of perspective projections for a 15-point object (solid lines) being aligned by the SoftPOSIT algorithm to an image (dashed lines) with one occluded point and two clutter points. The iteration step of the algorithm is shown under each frame.

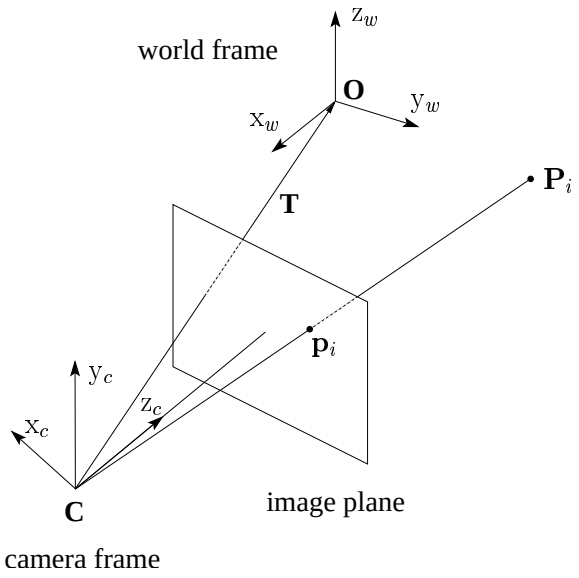


Figure 1.4: Using a pin-hole camera model, the perspective projection of world point $\mathbf{P}_i$ is the image point $\mathbf{p}_i$.

single image and model shape context provides a good initial estimate of the object's pose when the correspondence is correct. This initial 2D pose estimate is then refined using a deterministic annealing algorithm, and then this refined 2D pose is used to initialize a 3D pose refinement algorithm for full 3D object recognition. This multi-stage algorithm is described in Chapters 6 and 7

1.3 Camera Models

Let $\mathbf{P}_i$ be a 3D point in a world coordinate frame with origin $\mathbf{O}$ (see Figure 1.4). If a camera placed in this world frame is used to view $\mathbf{P}_i$, then the coordinates of this point in the camera frame may be written as $R\mathbf{P}_i + \mathbf{T}$. 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 k^{th} row of R be denoted by $\mathbf{R}_k$ and let the translation be $\mathbf{T} = (T_x, T_y, T_z)^{\mathsf{T}}$.

We use the standard pinhole camera model for perspective projection and assume that the camera is calibrated with unit focal length, so that pixel coordinates can be replaced by normalized image coordinates. Then, the *perspective* image of a 3D point $\mathbf{P}_i$ in the world frame is $\mathbf{p}_i = (x_i, y_i)$ where

$$x_i = \frac{\mathbf{R}_1\mathbf{P}_i + T_x}{\mathbf{R}_3\mathbf{P}_i + T_z}, \quad y_i = \frac{\mathbf{R}_2\mathbf{P}_i + T_y}{\mathbf{R}_3\mathbf{P}_i + T_z}. \tag{1.1}$$

The *weak perspective* (or *scaled orthographic*) projection model makes the assumptions that the depth of the viewed object is small compared to the distance of the object from the camera, and that the object is close to the optical axis. Because $\mathbf{R}_3$ is a unit vector in the world coordinate frame that is parallel to the camera's optical axis, under the weak perspective assumptions the distance of object points from the camera center, $\mathbf{C}$, may be approximated by a constant: $\mathbf{R}_3\mathbf{P}_i + T_z$ is approximately constant for all i. This implies that $\mathbf{R}_3\mathbf{P}_i$ is approximately constant for all i. In the following, to simplify the equations, we assume without loss of generality, that $\mathbf{R}_3\mathbf{P}_i \approx 0$ whenever the weak perspective assumptions are true. This is easily satisfied by placing the origin of the world coordinate at the centroid of the object points. The weak perspective image of a 3D point $\mathbf{P}_i$ in the world frame is then $\mathbf{p}_i^w = (x_i^w, y_i^w)$ where

$$x_i^w = \frac{\mathbf{R}_1\mathbf{P}_i + T_x}{T_z}, \quad y_i^w = \frac{\mathbf{R}_2\mathbf{P}_i + T_y}{T_z}. \tag{1.2}$$

Thus, the weak perspective image of any point is its orthographic projection divided by the scale factor T_z, hence the name "scaled orthographic projection." When the above assumptions are accurate, the weak perspective image of a scene point is an accurate approximation to its perspective image.

As seen in Equation (1.1), the perspective image and camera coordinates of a scene point are related by nonlinear equations. The weak perspective projection model has the advantage that the image and camera coordinates of a scene point are related by linear equations (Eq. (1.2)). For this reason, the weak perspective projection model is often used in order to simplify numerical algorithms. However, indiscriminate use of this approximation when it is not accurate can lead to poor pose estimates. We use the weak perspective camera model in the iterative SoftPOSIT algorithm in such a way that this model becomes more accurate as the algorithm converges to correct solutions.

1.4 Outline

The rest of this dissertation is organized as follows. Chapter 2 describes previous work by other authors related to the model-to-image registration problem. Chapter 3 then presents the SoftPOSIT algorithm that uses deterministic annealing to simultaneously determine the pose of a 3D model of point features and their correspondences to feature points in an image. The algorithm combines the iterative graduated assignment algorithm [55, 56] for computing correspondences and the iterative POSIT algorithm [38] for computing object pose under a full-perspective

camera model. This algorithm, unlike most previous algorithms for pose determination, does not have to hypothesize small sets of matches and then verify the positions of the remaining image features. Instead, all possible matches are treated identically throughout the search for an optimal pose. The SoftPOSIT algorithm is then extended in Chapters 4 and 5 to two different algorithms for the case of matching 3D line segments in the model to 2D line segments in the image. Next, in Chapter 6, a multi-stage algorithm is presented that first uses shape contexts to rank affine pose hypotheses and then applies deterministic annealing to refine these poses to determine more accurate pose and correspondences of the 2D objects. This 2D algorithm is extended into a view-based object recognition algorithm for 3D objects in Chapter 7. Experimental results of applying these algorithms to highly cluttered synthetic and real imagery appear in the chapters where each algorithm is described; in general, good performance is obtained on some very difficult data. Additionally, Chapter 8 compares the performance of the three line-based recognition algorithms on identical data. Finally, the dissertation concludes in Chapter 9 with an summary of the results and contributions of this work.

Chapter 2

Related Work

A large body of previous work is related to the problem of determining the pose and correspondences of objects. This work dates back at least 40 years [126]. This chapter attempts to give an overview of this work. In order to perform feature-based pose and correspondence estimation, features must first be detected in images; this subject is discussed in Section 2.1. Next, Section 2.2 describes methods for estimating correspondences of point and line features in the absence of information regarding object geometry. Sections 2.3 and 2.4 discuss methods for computing the pose of an object (or equivalently, the camera) when it is assumed that the image-to-model feature correspondences (points and lines, respectively) are known; some of these techniques will be employed in algorithms described in later chapters of this dissertation. Methods for object recognition, that is, computing the pose and correspondences of objects when neither pose nor correspondences are known in advance, are discussed in Section 2.5. Finally, because the graduated assignment algorithm is a key component of the new algorithms presented in this dissertation, it is discussed in detail in Section 2.5.2.2.

2.1 Image Feature Detection

Image features may be global, describing an entire image, or local, describing only a small region of an image. For the application of pose estimation, local features are more appropriate as their correspondences to a set of model features allows an accurate calculation of an object's pose. A variety of local features may be used for this purpose, including points, lines, regions, as well as higher-dimensional features. Corners are generally defined as 2D point features where there is high greylevel variation in two orthogonal directions. In contrast, edges are locations that have high greylevel variation in one direction, with low variation in the orthogonal direction. The biggest problem with image feature detection is the lack of consistent detection of features from one image to the next.

2.1.1 Point features

Point features in images are generally located using local interest operators, which select pixels that are good candidates for matching between two images. These operators use gradient, color, texture, and other local properties. One of the most widely used operators was the Moravec interest operator [106], which measures how distinct a region of the image is from its surroundings. Interest operators may be useful in model-to-image registration when there is reason to believe that interest points in the image are likely to correspond to some features in models. This may be the case for corner detectors: corners in an image may correspond to junctions of model edges. Consequently, a number of corner detectors are reviewed below.

The Harris corner detector [69] is a popular method for locating corners in images. If an input image I is very noisy, it should first be smoothed by convolving it with a Gaussian. Then, at every point in I, the derivatives I_x and I_y are computed. The local structure matrix at a point is defined as

$$M = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}.$$

This matrix is positive semidefinite, so its two eigenvalues will be nonnegative. In general, the eigenvectors of this matrix encode the edge directions, and the eigenvalues encode the edge magnitudes. At a uniform region in the image, the two eigenvalues of M will be equal to zero. For a perfect step edge, one eigenvalue will be large (with the associated eigenvector being orthogonal to the edge), and the other eigenvalue will be equal to zero. For a corner, both eigenvalues will be large. Thus, corners can be identified as points in the image where the smaller of the two eigenvalues is greater than some threshold. Usually, nonmaximal eigenvalues are suppressed over local windows in the image, so that at most one corner is detected in any small window. The two eigenvalues don't actually need to be computed: their ratio can be determined from the determinant and trace of the matrix M. The corner response function is defined as $R = \det(M) - \kappa\,(\mathrm{trace}(M))^2$ where $\kappa \approx 0.04$. A corner is then detected at a pixel (x, y) when $R(x, y) \geq \tau$ for some threshold τ.

The Harris corner detector is very sensitive to image scale, so it's not good for use in matching image features when the same feature may have to be located at

a different scale in each image. The Harris corner detector is closely related to the Kanade-Lucas-Tomasi (KLT) feature detector [145]. The main difference is that the KLT feature detector applies a threshold directly to the smallest eigenvalue, while the Harris corner detector applies a threshold to the corner response function.

The SUSAN detector [136] is an approach to edge and corner detection, and to structure preserving noise reduction. SUSAN stands for Smallest Univalue Segment Assimilating Nucleus. Unlike many other feature detectors, the SUSAN detectors are nonlinear operators. Each image pixel has associated with it a *local similarity region* of pixels (within some circular window) that have brightness values similar to that of the center pixel. From the area, centroid, and second moment of these local similarity regions, two dimensional features and edges are detected. This approach to feature detection does not require the computation of image derivatives or smoothing for noise reduction; for this reason, it performs well in the presence of noise provided the noise is below the similarity threshold.

The area of a local similarity region is at a maximum when the center lies in a flat region of the image surface. It falls to half of this maximum very near a straight edge and falls even further when inside a corner. This property of a local similarity region's area is the main indicator of the presence of the edges and two-dimensional features. An image that is processed to give as output 1/area has edges and corner features strongly enhanced with the corner features more strongly enhanced than edges. The authors claim that the strength of this feature detector is that its parameters are much simpler and less arbitrary, and therefore easier to automate, than those for other feature detection algorithms.

Lowe [97] developed the Scale-Invariant Feature Transform (SIFT) method to locate distinctive image features that are invariant to changes in scale and rotation in the image plane, and partially invariant to changes in 3D viewpoint. These may be used for 2D object recognition, or 3D object recognition when multiple views of the object are trained in advance.

The features are created as follows. First, a scale-space representation of the image is created as a pyramid of difference of Gaussians at different scales convolved with the input image. The difference of Gaussian is a good approximation to the Laplacian of Gaussian, which gives the second derivative of a Gaussian-smoothed image. Zero crossings in the Laplacian will occur where the gradient magnitude is maximal in the original image. This scale space is searched to find image features whose locations are invariant to changes in scale. These features will be at local maxima and minima in the 3D scale space (2D position and scale). The precise location and scale are found by fitting a 3D quadratic function to the 3D scale space. The eigenvalues of the Hessian matrix are used to eliminate features that have large derivatives in only one direction. This scale determines the scale at which all subsequent computations occur. The orientation of a feature is assigned the dominant orientation from the gradient orientations of all points in a local window around the feature. Multiple features are assigned to the same point if there are multiple dominant orientations at that point. All subsequent computations are performed relative to this orientation. This location, scale, and orientation forms a local coordinate system in which to describe the local image region, thus making the feature vector scale and orientation invariant.

The feature vector for each feature point is a 4×4 array of gradient orientation histograms, where each histogram is computed from a 4×4 region of pixels in the neighborhood of the feature point. Each histogram is quantized into 8 possible directions, and the magnitude of each directional bin is the sum of the magnitudes of points with the given direction weighted by the distance of the point from the center of the window. This produces a $4 \times 4 \times 8 = 128$ element feature vector. Finally, the feature vector is normalized to reduce the effects of illumination changes.

2.1.2 Line features

Edges may be defined as local discontinuities in intensity (or some other image feature such as texture), and *lines* may be defined as collections of contiguous edges. *Straight lines* are collinear and contiguous edges. Line detection is difficult for a number of reasons: usually, the extent of local operators are small relative to features they are designed to detect; image data usually deviates from an ideal model of the events being detected; and, the discrete and quantized nature of images causes aliasing. Line detectors are subject to the following types of errors: false lines detected on low-magnitude gradients, curved lines being approximated as straight lines, and multiple lines erroneously being grouped into a single line. The following approaches to line detection attempt to deal with these issues.

Canny [21] addresses the problem of detecting edges in noisy images. He gives an analytical derivation of an "optimal" edge detector. The main contribution of this paper is that it combines several useful methods into a practical algorithm. Canny

formalizes the edge detection problem as a constrained optimization problem. The constraints that he optimizes are the following:

1. Achieve low error rate: minimize the probability of type I error (missing a true edge point) and the probability of type II error (incorrectly marking a nonedge point as an edge). This can be accomplished by maximizing the signal-to-noise ratio of the detector output.

2. Achieve good localization: the detected point should be as close to the center of the true edge as possible. The reciprocal of the standard deviation of the edge location error is maximized.

3. Minimize the number of responses to a single edge: This is implicit in the first constraint, but it's made explicit here. This is achieved by forcing the distance between peaks in the noise response of the detector to approximate the width of the detectors response to a single step edge.

Through a complicated analysis of 2D continuous step edges, Canny derived a detector that is the sum of four exponential terms. This function, however, is very similar to the first derivative of a Gaussian, so this is what he implemented.

The Canny edge detector first smooths the image with a 2D Gaussian to reduce the effects of noise. It then computes the image gradient to locate pixels with high spatial derivatives. The gradient magnitude and direction at each pixel is computed from derivatives in the x and y directions. These derivatives are estimated by finite differences. The algorithm then performs nonmaximal suppression: any pixel whose

gradient magnitude is not maximal along its gradient direction is suppressed. The gradient array is then further reduced by hysteresis thresholding where a high and low threshold is used: pixels whose gradient magnitude exceed the high threshold are accepted as edge points; pixels whose gradient magnitude is lower than the low threshold are rejected; and, pixels whose gradient magnitude is between the two thresholds are accepted only if there is a path consisting only of pixels above the low threshold that connects to a pixel above the high threshold. In other words, each "weak" edge point must be connected to a "strong" edge point by other edge points (weak or strong).

The gradient of a Gaussian filtered image could be computed in one step by convolving the image with the derivative of a Gaussian. This is a first-order detector: edges will occur at peaks in the detector response. A second-order detector would compute the second derivative of the Gaussian filtered image, or would convolve the image with a Laplacian of Gaussian, which is approximately a rotationally-invariant second derivative of a Gaussian. The second derivative operator locates edges at zero crossings in its output; peaks in the first derivative correspond to zero crossings in the second derivative. The Laplacian of Gaussian filter can be approximated by filtering with the difference of two Gaussians.

Once edge points are detected in an image, these must be clustered into straight lines. The Hough Transform [77, 44] is a popular approach for line detection from binary edge images. The most common form of the Hough Transform uses a (θ, ρ) parameterization of line space. In this case, the equation of a straight line is represented by the equation $x \cos\theta + y \sin\theta = \rho$. With this, each edge point (x, y) is

transformed into a sinusoidal curve. Curves corresponding to collinear image points will have a common point of intersection in (θ, ρ)-space. A discrete accumulator array is formed and for each point (x, y), all cells in the accumulator array along it's (θ, ρ)-curve are incremented. After processing all edge points in the image, accumulator cells with high counts are identified as corresponding to nearly collinear subsets of edge points, i.e., long, straight lines. This algorithm is very sensitive to the quantization of θ and ρ. With finer quantization, better resolution is obtained at the expense of more computation, however some lines may then be spread over two or more separate bins. With coarser quantization, the method may find lines corresponding to unrelated groups of collinear image points.

The Burns et. al. line detector [16] is an approach to extracting straight lines from intensity images. A goal of their approach is to be able to detect lines of arbitrarily low contrast. The general approach to is group pixels into line-support regions on the basis of gradient orientation, and then to extract a straight line segment from each region. This process can be broken down into the following four steps:

1. Two small 2×2 masks are first used to compute the gradient magnitude and orientation. Each pixel votes for one of two possible quantized orientations closest to the gradient orientation of that pixel. Each quantized orientation determines a set of contiguous pixels of similar orientation; these are called *line-support regions*. A pixel votes for the orientation (line-support region) that provides the longest line through that pixel. The line-support regions

with a majority of votes are selected, and their supporting pixels are given common labels. Every pixel in the image is given the label of one line-support region; many of these support regions will be very small, corresponding to lines possibly caused by image noise.

2. For each candidate line support region, a planar surface is fit to the greylevel values of the associated pixels. The location of the line is then determined by the intersection of this plane with a horizontal plane whose height is set to the average intensity of the pixels in that support region.

3. Extract attributes of each line using pixels in the line-support regions: contrast, steepness, width, straightness

4. Lines are filtered according to the application requirements. Lines may be filtered based on length and steepness to remove lines due to noise and clutter.

Lowe [95] extracts straight lines from binary edge images using a recursive subdivision algorithm. Binary edge points are first linked together into contours by tracking adjacent edge points. When a junction is encountered along an edge contour, one of the branches is arbitrarily selected to continue the contour; the other branch is eventually processed as a separate edge. Each edge contour is then recursively subdivided at the point of maximum deviation from a line connecting its endpoints. This process is repeated until each segment is no more than four pixels in length. Then, adjacent segments are merged into a single segment whenever the length-to-deviation ratio of the merged segment is greater than that of either unmerged segment.

2.2 Correspondence Estimation

The problem of determining the correspondences between the features in two images, or an image and a 2D or 3D model, is very difficult with many applications in computer vision and image analysis. When features are being matched between two images, one with m features and the other with n features, some sort of feature correlation or similarity measure (e.g., [15, 97]) may be used to reduce the mn possible correspondences to a much smaller number of likely correspondences. However, when one of the images is replaced by a geometric model described only by points or lines, this is no longer possible because any point or edge in the model can look like any point or line in the image. In this case, all mn correspondences must be considered, and global geometric constraints must be used to separate the correct from incorrect correspondences.

In combinatorial optimization problems, an energy (or cost) function is defined in terms of finite set of discrete variables and the goal is to find a state of those variables that minimizes the energy function [63]. The correspondence problem is a combinatorial optimization problem: correspondences between n image features and m model features can be described by an $n \times m$ binary matrix; there are mn binary-valued variables. A variety of general-purpose methods exist to solve combinatorial optimization problems. These include exhaustive search, dynamic programming, steepest descent, linear programming, branch-and-bound, simulated annealing, genetic algorithms, and neural networks, among others [63]. For complex problems, where the search space grows exponentially with the size of the input,

algorithms that are guaranteed to find an optimal solution will require impractical run times. For the class of NP-complete problems, all known algorithms that are guaranteed to find an optimal solution require a runtime that increases exponentially in the size of the input [116]. In these cases, heuristics may be used to reduce the size of the space actually searched, but this is done at the expense of no longer being able to guarantee an optimal solution: these heuristic algorithms are easily trapped in suboptimal local minima.

Ullman [151] stated that a good *global* correspondence (i.e., for all features) between two sets of features could be obtained by satisfying the following three *local* criteria:

1. The *similarity principle* requires that corresponding features should have similar appearance.

2. The *exclusion principle* requires that the mapping between the two sets should be one-to-one.

3. The *proximity principle* says that, all other factors being equal, the smaller the distance between two features, the greater the preference for that match.

These principles are valid provided that the two sets of features were observed by cameras close in pose. A number of authors have developed methods which, given an initial correspondence (or match) matrix (as defined on page 11), maps this into a new matrix that better satisfies Ullman's three principles. Because the use of correspondence matrices is integral to the 3D and 2D recognition algorithms

presented in later chapters of this dissertation, a number of approaches for refining correspondence matrices are reviewed below.

Scott and Longuet-Higgins [132] present a method for determining the correspondences between two 2D point patterns of different sizes where the only features of the patterns are the positions of the points. The method uses properties of the singular value decomposition (SVD) to satisfy the exclusion and proximity principles set forth by Ullman. Because all points have identical appearance, the similarity criteria is not considered. The algorithm first builds a proximity matrix G between the two sets of points. It then computes the singular value decomposition of $G = VDU^{\mathsf{T}}$. Next, the diagonal matrix D is mapped to a new matrix D' by replacing every diagonal element with a 1. The matrix $P = VD'U^{\mathsf{T}}$ is similar to G but it has the property that good matches are accentuated and bad matches are attenuated. Feature i is put into correspondence with feature j if P_{ij} is the largest element in both its row and column. The exclusion principle is satisfied because P is orthogonal and the sum of squares of the values in each row of P equal to one. The proximity principle is satisfied because P produces a minimum squared distance mapping, as witnessed by the fact that the trace of $P^{\mathsf{T}}G$ is maximized. Pilu [119] has extended this method by implementing a correlation-weighted proximity matrix in place of G.

Shapiro and Brady [133] present a method to determine correspondences between two patterns of 2D points that have undergone a 2D similarity transformation. Unlike the Scott and Longuet-Higgins algorithm [132], the Shapiro and Brady algorithm accounts for the structural relationships between the features in an image. In

this algorithm, each feature is mapped into a higher dimensional feature space so that each feature "has knowledge" of its local surrounding features. The algorithm first analyzes the shape of each pattern to generate its *modes*. Modes encode the shape of a point pattern based on the inter-point (within a single image) distances. A matrix H of inter-point distances is computed for each image. From the singular value decomposition $H = VDV^{\mathsf{T}}$, the eigenvalues of H are computed; these are the *modes* of the corresponding image. The columns of V are its eigenvectors. The i^{th} row of V gives the coordinates of the i^{th} feature with respect to modal basis vectors. To compare images with m and n features, respectively, only the first $\min\{m, n\}$ components of each feature vector are kept. When the shapes of two images are similar, features from the two images with the same local shapes will have similar modal features. The initial correspondence matrix between the two images is now based on the Euclidean distances between the corresponding feature vectors. This matrix replaces the initial proximity matrix in the Scott and Longuet-Higgins algorithm.

Gold and Rangarajan [55] developed the softassign algorithm that maps a distance matrix into a doubly stochastic assignment matrix. All rows and columns of a *doubly stochastic matrix* sum to one, so that the assignment matrix represents a probability function for the correspondences between the two sets of features. The assignment matrix is first initialized using an exponential function of the distances between corresponding feature points. Then, the Sinkhorn algorithm [135], which consists of alternating row and column normalizations, is used to ensure that the doubly stochastic constraint is satisfied. In [56], the authors show that iterated row

and column normalization used in softassign produces the same results as would be obtain in solving the equivalent constrained optimization problem with Lagrange multipliers. The softassign algorithm is a component of the algorithms presented in later chapters of this dissertation and is describe in more detail in Chapter 3.

2.3 Calculating Camera Pose from Point Correspondences

Photogrammetry [102] is the calculation of lengths and angles from measurements made using one or more images. It involves determining the position and orientation of cameras and objects in a scene. Classical photogrammetry is concerned with obtaining accurate measurements from noncontact imaging. The discipline of computer vision is related to photogrammetry, but is also concerned with the speed and robustness of the reconstructions, for use in real-time systems. There are four main problems in photogrammetry: absolute orientation, relative orientation, exterior orientation, and interior orientation [73]. The *absolute orientation* problem is to determine the position and orientation relating two sets of corresponding 3D points. The *relative orientation* (or structure from motion) problem is to determine the relative position and orientation of two cameras viewing a common scene given corresponding 2D image points. The *exterior orientation* (or *perspective-n-point*, or *pose estimation*) problem is the problem of determining the position and orientation (translation and rotation, respectively) of a camera relative to an object given correspondences between a set of 3D object features and a set of 2D image features observed by the camera. Finally, the *interior orientation* (or *camera cali-*

bration) problem is that of determining the internal parameters of a camera (e.g., focal length, image center, skew, etc.) from images obtained from the camera. In order to determine an object's pose, the internal parameters of the camera must be known or computed. Sometimes, camera calibration and object pose are combined into one problem and solved simultaneously (e.g., [148]). We assume throughout this dissertation that the camera internal parameters are known, so that only the position and orientation of the camera needs to be determined.

In this and the next section, we discuss approaches for solving the exterior orientation problem, since solving the simultaneous pose and correspondence problem will require solving exterior orientation. This section discusses methods that use point features, and the next section those that use line features. The methods for point features are divided into two groups: those that use a perspective camera model, and those that use simpler approximate camera models.

2.3.1 Perspective Cameras

The pose of a camera has six degrees of freedom, three for rotation and three for translation. The most common geometric camera model is the *perspective* or *pinhole* camera [51]. In this model, scene points project along straight rays through a single point **O** (the *pinhole*, or *center of projection*) onto a planar image that's perpendicular to the camera's optical axis and a distance f (the *focal length*) from **O**. The point on the image plane at which the camera's optical axis intersects the image plane is known as the *principal point* of the image, and for the pinhole

camera model, coincides with the center of the image. Under this model, a scene point (X, Y, Z) projects to an image point (x, y) according to the equation

$$\begin{aligned} x &= f\frac{X}{Z} \\ y &= f\frac{Y}{Z}. \end{aligned} \tag{2.1}$$

The pinhole camera model may be used with cameras that don't exactly fit the above model (because of image plane skew, an offset principal point, or lens distortion) by determining the internal parameters of the camera [148] and then transforming the image according to these parameters so that it appears to have come from a true pinhole camera.

As shown in Equation (2.1), each point correspondence provides two constraints on camera pose. Thus a minimum of three correspondences are necessary to determine pose. However, in many cases with a small number of correspondences, the solution is not unique [49, 72]. With three correspondences, there can be up to four real solutions corresponding to object points in front of the camera, and up to four false solutions corresponding to object points behind the camera. There is a unique solution when using four correspondences to coplanar object points. Four or five correspondences with object points in general position (noncoplanar) admits up to two real solutions. Finally, six or more correspondences in general position always has a unique solution.

When a pose computation produces multiple solutions, a number of methods can be used to reduce these to a smaller number. First, the translation of the object

must be such that visible object points are in front of the camera. Second, when a solid model of the object is available, the pose must not cause visible object points to be occluded by other parts of the object. Finally, one can check that additional object points project sufficiently close to their corresponding image points.

The problem of estimating a camera's pose using correspondences of image and model points has been extensively studied. There are a large number of solutions to this problem which may be loosely grouped into three classes: analytic (close-form) methods, numerical linear methods, and iterative methods. The remainder of this section gives a brief survey of these methods.

Closed-form methods ([49, 92, 66, 72]) analytically solve small systems of non-linear equations for the desired pose. The advantages of these methods are that they are fast because they are noniterative, they require only a few correspondences, and they don't require initial guesses for the object's pose. Their main disadvantages are they cannot take advantage of large numbers of correspondences to reduce the effects of noise, and they often give multiple solutions. These methods are only applicable when a small number of points are involved. For three corresponding points, the methods generally start by relating the image points to the depths of the corresponding object points using the law of cosines and then, through various substitutions and changes of variables, reduce the problem to that of finding the zeros of a polynomial equation in one unknown. Hourd et al. [72] gives a closed-form solution for four corresponding points. An alternative for four or more correspondences is to apply the above 3-point algorithms to subsets of three correspondences and then find the common solutions. Because these methods use only a small num-

ber of points, they fail to take advantage of the redundancy in large data sets, and consequently their accuracy is very sensitive to measurement noise and to the order in which equations are used [66].

Numerical linear methods ([2, 48, 67, 147, 123], e.g.) solve for object pose by setting up a system of linear equations whose solution is the desired pose. The advantages of these methods are that they are relatively fast because they are non-iterative, they can take advantage of large numbers of correspondences to reduce the effects of noise, they are not subject to getting trapped in local minima, and they don't require an initial guess for the object's pose. Their main disadvantage is that, because the error functions minimized by these methods cannot account for nonlinear problem constraints, the error functions are not geometrically or statistically meaningful, and hence the resulting solutions are often not optimal from a geometrical or statistical perspective.

Quan and Lan [123] present a linear algorithm to compute pose from a redundant system of polynomial equations for four or more point correspondences. Every pair of correspondences between an image point and an object point gives rise to a quadratic constraint on the depths of the two object points in the camera reference frame. By combining these constraints for three pairs of correspondences, a fourth degree polynomial in one unknown depth is obtained. For four or more points, a system of fourth degree nonlinear equations is produced which can be viewed as a homogeneous linear equation in powers of the unknown depth. By applying SVD twice for four correspondences, or once for five or more correspondences, the unknown depths can be solved for. The pose problem is thus reduced to the absolute

orientation problem, which has well-known closed-form solutions.

The linear algorithm of Fiore [48] uses an orthogonal decomposition to split the unknowns (scale, rotation, translation, and object point depths) into two groups with separate equations. First, the object point depths are found by solving a system of linear equations defined from six or more point correspondences. Then, the unknown pose parameters are found by solving an absolute orientation problem using SVD methods.

Ansar and Daniilidis [2] present another exact linear solution to the pose estimation problem for four or more point or line correspondences, provided the solution is unique. They first recover the depths of the object points using the geometric rigidity constraint between all pairs of object points. A change of variables is used to linearize a system of quadratic equations that relates the depths and distances between the object points to their corresponding image points. Once the object point depths in the frame of the camera have been recovered, the camera pose is found by solving the absolute orientation problem using either unit quaternions [74] or SVD [75] methods. Their method can also be applied to the case of corresponding image and object lines.

For a six or more correspondences, the approximate *direct linear transform* (DLT) algorithm may be used [67]. This method, by ignoring the orthonormality constraints on the rotation matrix, produces a set of linear equations whose solution is an approximation to the 3×4 homogeneous camera projection matrix (for a non-calibrated camera). Proper data normalization is necessary to obtain good results. Once the camera projection matrix is obtained, the approximate camera rotation

and translation can be found in closed-form by decomposing the camera matrix into its constituent parts [52].

Iterative methods determine object pose by searching for the pose that minimizes a nonlinear objective function. These methods start with an initial guess for the pose and iteratively refine it until some convergence criteria is met. The advantages of these methods are that they can be very accurate due to the ability to incorporate any arbitrarily accurate and complex imaging model, and they can take advantage of large numbers of correspondences to reduce the effects of noise. Their main disadvantages are that they are often slow to converge, or even worst, they may diverge, they at best return a local minimizer of the objective function, and they require a good initial guess of the pose in order to converge to the correct answer.

Well known nonlinear optimization methods, such as gradient descent, the Gauss-Newton method, and the Levenberg-Marquardt method, can be used to optimize nonlinear objective functions based on geometric constraints in the image or world. These methods require the calculation of the first and possibly second partial derivatives of the objective function, and consequently can be computationally intense. The works of Haralick et al. [65], Yuan [154], Lowe [96], and Phong et al. [118] are examples this approach.

Other iterative methods more carefully integrate the geometry of the pose problem into the pose refinement steps of the algorithm. The POSIT algorithm of DeMenthon and Davis [38] uses a scaled orthographic projection (SOP) camera model to approximate perspective projection. The iteration begins by approximat-

ing SOP image points with the measured perspective image points. On each step of the iteration, the approximate SOP image points are used to compute a new pose estimate, and then the new pose estimate is used to improve the estimated SOP image points. When the depth of the object is small compared to the distance of the object from the camera, the initial approximation is accurate and the iteration converges to the true pose in a small number of steps. More details of this algorithm are given in Chapter 3.

In the iterative method of Lu et al. [98], the pose estimation problem is formulated as that of minimizing the sum of the squared distances between object points (mapped into the camera reference frame) and their orthogonal projections onto the lines of sight of their corresponding image points. Their *orthogonal iteration* algorithm first minimizes this objective function with respect to rotation by solving an absolute orientation problem (using SVD) [75] with translation held fixed. Then, with this new rotation matrix, the translation that minimizes the objective function is calculated in closed-form. The process is repeated until a fixed point is obtained, which they show always occurs (even when the object model *should not* match to the image).

2.3.2 Approximate Cameras

Although the perspective camera model is an accurate model for a wide variety of existing cameras, the projection equations are nonlinear because the image of a scene point is inversely related to the distance of that point from the camera. When

the depth of a scene being viewed is small compared to the distance of the camera from the scene, and when the distance of scene points from the camera's optical axis is small, *affine* models of camera projection provide a good approximation to perspective projection [68, p. 156]. These approximations result in linear projection equations, which greatly simplify the calculation of pose.

The simplest affine camera model is the *orthographic projection* model [68, p. 158], whose projection equations are

$$\begin{aligned} x &= X \\ y &= Y. \end{aligned} \tag{2.2}$$

Orthographic projection projects points onto the image plane through rays that are parallel to the camera's optical axis; the distance of scene points from the camera has no affect on their projections in the image. Because the size of an object's image cannot be modeled with this camera, this model is not often used.

Next, in terms of complexity, is the *scaled orthographic projection* model [68, p. 158], whose projection equations are

$$\begin{aligned} x &= sX \\ y &= sY, \end{aligned} \tag{2.3}$$

where s is a constant isotropic scale factor. This is similar to orthographic projection except the size of an image can be modeled using the parameter s. When the average distance of a scene from the camera is $\overline{Z}$, and the depth of the scene is

small compared to $\overline{Z}$ (i.e., the object plane is parallel to the image plane), then taking $s = f/\overline{Z}$ results in a good approximation to perspective projection by scaled orthographic projection. The scaled orthographic camera has six degrees of freedom: three for the 3D rotation, two for the location of the principal point, and one for the scale factor s. Since each point correspondence $(X, Y, Z) \leftrightarrow (x, y)$ provides two constraints, three point correspondences are sufficient to compute the camera's pose from the linear equations (Eq. 2.3).

The *weak perspective* projection model is similar to scaled orthographic projection model except that two scale factors are used, one for the X direction and one for the Y direction [68, p. 159]. It has seven degrees of freedom and therefore requires four point correspondences (really just $3\frac{1}{2}$) to determine the pose of a camera using this model.

The most general affine camera model has projection equations given by

$$\begin{aligned} x &= a_1 X + a_2 Y + a_3 Z + t_1 \\ y &= a_4 X + a_5 Y + a_6 Z + t_2 \end{aligned}$$

where the a_i and t_i are constants. This model covers the composed effects of an affine transformation of 3-space, followed by an orthographic projection onto the image, and then followed by an affine transformation of the image. An salient property of an affine camera is that parallel lines in a scene are projected to parallel lines in the image. The orthographic, scaled orthographic, and weak perspective camera models discussed above are all specific instances of affine cameras. Because the

general affine camera has eight degrees of freedom, four point correspondences are needed to determine the pose of a camera using this general model.

It is well-known [68] that a scaled orthographic or weak perspective image of a *planar* object is related to the original object by a 2D affine transformation of the plane containing that object. Since $Z = 0$ in this case, the affine projection equations become

$$\begin{aligned} x &= a_1 X + a_2 Y + t_1 \\ y &= a_4 X + a_5 Y + t_2. \end{aligned}$$

For this projection model, which has six degrees of freedom, camera pose can be computed from the linear equations given by three point correspondences. This model is accurate, and therefore commonly used, when viewing planar scenes from a distance.

Finally, it can also be seen [68] that the image of a planar object, taken by a perspective camera in *any* position, is related to the original object by a 2D homographic transformation (i.e., a general 2D projective transformation) of the plane containing that object. Because a homography has eight degrees of freedom, four point correspondences are required to determine the pose of a camera using this model. This is not an affine camera model (the image of parallel lines in the scene needn't be parallel in the image), but when homogeneous coordinates are used for the scene and image points, the projection equations are linear.

2.4 Calculating Camera Pose from Line Correspondences

As discussed in the previous section, the pose of a camera can be computed from correspondences between point features in the object and image. A problem with matching point features, however, is that it is generally very difficult to reliably extract them from an image, unless templates are available to correlate to the image. Reliable point extraction may be possible in some controlled environments (e.g., circuit board inspection on an assembly line), but is usually not possible in general 3D environments where a scene point can have drastically different appearances depending on the camera viewpoint. Line features, on the other hand, can be reliably extracted from images because they can be located along curves in the image of maximum intensity gradient, which are easily identified. By fitting a line model to the image using interpolative methods, a high degree of accuracy in the position of the line can be obtained. Because most real world objects have some straight edges that will project to straight lines in an image, it is not very restrictive to assume that object and image lines will be straight. We therefore make this assumption in this dissertation.

When using line correspondences, one should not expect that the ends of the object lines correspond to the ends of the corresponding image lines. Due to occlusions, shadows, and imperfect line detection algorithms, the ends of object lines are often not accurately located in images. Thus, the constraint derived from a correspondence between an image line and object line is that the projection of the object line should be collinear with that image line. Correspondences between

image and object lines produce constraints that are weaker than correspondences between image and object points. A set of point correspondences can be used in a line correspondence algorithm by synthesizing a line between each pair of points in both the image and object. A set of line correspondences, however, can only be used by a point correspondence algorithm when the object lines provide a sufficient number of 3D points from their intersections in 3-space.

Image and object lines may be represented in a number of different ways. The particular representation chosen depends on whether or not the line has finite or infinite extent, and on the particular requirements of the application. A 2D line has three degrees of freedom. Possible representations for a 2D line include the three constants in the equation $ax + by + c = 0$, two 2D endpoints, a 2D point and a 2D vector, and the distance of the line from the image origin and the angle it makes with one coordinate axis. A 3D line has four degrees of freedom. Representations for 3D lines include two 3D points, a 3D point and vector, two planes, and it's 4×4 Plucker matrix [68].

Due to occlusion of the object and faulty image processing, one should not expect that the endpoints of an object line will project on or near the endpoints of the corresponding image line; we can only expect that the projected model line is collinear with the corresponding image line. In other words, if object line L_j corresponds to image line l_i, this constrains L_j to lie in the plane passing through the camera center $\mathbf{C}$ and l_i. See Figure 2.1. In the following, we assume that the camera internal parameters are known (pixel coordinates are given in a normalized image coordinate system), and that the image and object lines are represented by

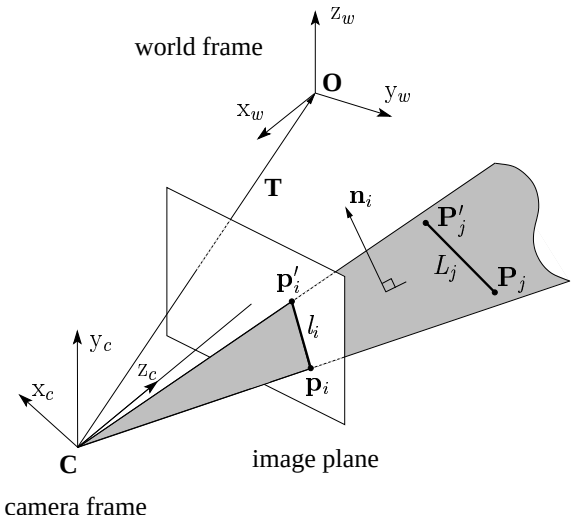


Figure 2.1: When image line l_i corresponds to 3D model line L_j, L_j is constrained to lie in the plane passing through the camera center $\mathbf{C}$ and l_i.

their 2D and 3D endpoints, respectively, so that $l_i = \{\mathbf{p}_i, \mathbf{p}'_i\}$ and $L_j = \{\mathbf{P}_j, \mathbf{P}'_j\}$. The normal to the plane passing through $\mathbf{C}$ and l_i is $\mathbf{n}_i = (\mathbf{p}_i, 1)^\mathsf{T} \times (\mathbf{p}'_i, 1)^\mathsf{T}$. This is also the homogeneous representation of the image line. The constraint that L_j lie in this plane is given by

$$\begin{aligned} n_i^\mathsf{T} \left(R\mathbf{P}_j + \mathbf{T}\right) &= 0, \\ n_i^\mathsf{T} \left(R\mathbf{P}'_j + \mathbf{T}\right) &= 0, \end{aligned} \tag{2.4}$$

where R is the rotation matrix and $\mathbf{T}$ is the translation vector that maps points from the world reference frame to the camera reference frame. Thus, each image-to-object line correspondence provides two constraints on the pose of the object. Because pose has six degrees of freedom, a minimum of three line correspondences are needed to determine it. Navab and Faugeras [111] have shown that with three object lines in general position (noncoplanar and not all intersecting at a common point) there can be up to three distinct solutions to the object's pose. When the three object

lines are coplanar, the three intersections of these lines can be used to reduce the problem to a perspective-3-point problem, which, as discussed in Section 2.3, can have up to four distinct solutions. With multiple solutions, the true pose can be determined using a number of approaches. First, the translation of the object must be such that visible object lines are in front of the camera. Second, when a solid model of the object is available, the pose must not cause visible object lines to be occluded by other parts of the object. Finally, additional object lines can be checked to determine if the given pose projects them sufficiently close to their corresponding image lines.

Methods for computing pose from line correspondences may be loosely grouped into three classes: analytic (closed-form) methods, numerical linear methods, and iterative methods. The advantages and disadvantages of the various methods are the same as those for point correspondences as discussed in Section 2.3. The remainder of this section gives a brief survey of algorithms that compute pose from line correspondences.

Closed-form methods analytically solve small systems of nonlinear equations for the desired pose. These methods are impractical for more than three line correspondences. Dhome et al. [42] present an analytic method of determining pose from correspondences between three image lines and three model lines. First, the rotation is decomposed using a number of intermediate coordinate systems. With these, an eighth degree polynomial in one unknown is obtained, and from this, the three rotational parameters are found. When the three object lines are coplanar or intersect at a common point, this equation is reduced to fourth degree. Thus, there

can be four or eight solutions to rotation. Then, given the rotation, the translation is found by solving a linear system of equations which constrain the rotated points to lie on the planes defined by the camera origin and the corresponding image lines.

Numerical linear methods solve for pose using only efficient linear algebra. The approximate direct linear transformation (DLT) algorithm can be applied using Equation 2.4. Luxen and Forstner [99] use this approach with an uncalibrated camera with both point and line features. From any combination of six or more point and line correspondences, they compute the camera projection matrix and its covariance matrix. From the camera projection matrix, the pose of an object can be readily found [52]. The method of Ansar and Daniilidis [2], discussed in Section 2.3, is another linear solution to the pose estimation problem for four or more line correspondences.

Iterative methods are used to find poses that optimizes nonlinear objective functions. Christy and Horaud [27] describe a method to iteratively compute pose from line correspondences using a scaled orthographic (weak) camera model. This is an extension of DeMenthon and Davis' POSIT algorithm for point correspondences [38]. When the depth of the object is small compared to the distance of the object from the camera, the perspective images of the object lines closely approximate the scaled orthographic projections of these lines. Substituting these approximations into the linear equations for scaled orthographic projection, the approximate camera pose can be found. New scaled orthographic image lines are computed and the process is repeated. The use of a weak perspective camera model requires only efficient linear algebra on each step of the iteration, but the final pose is still that

of the perspective camera. Christy and Horaud's method minimizes an algebraic distance on each step, unlike the POSIT algorithm for point correspondences, which minimizes a geometric error measure on each step [28].

Liu et. al. [93] determine object pose from line correspondences using a method that first computes the rotation, and then the translation. By ignoring the orthonormality constraints and treating the nine elements of the rotation matrix as independent unknowns, the rotation can be computed with a linear algorithm given eight or more correspondences. Alternatively, the rotation can be computed as three Euler angles for three or more correspondences using a nonlinear iterative algorithm to minimize an objective function. The objective function is linearized about the current estimate of the object's pose (e.g., $\cos(\theta + \Delta\theta) \approx \cos\theta - \Delta\theta \sin\theta$ where θ is the current estimate for a pose parameter and $\Delta\theta$ is the update to be solved for), and then on each iteration, linear adjustments to the pose are found which minimize this function. Once the rotation is found, the translation is solved from three or more correspondences from linear equations.

Lowe [95] computes pose from line correspondences in his SCERPO system using Newton's method to minimize the least squares error between image features and the corresponding projected object features. The object-to-image projection equations are reparameterized to simplify the calculation of the partial derivatives of the error function with respect to the pose parameters.

Kumar [88] shows that better noise immunity can be achieved if the rotation and translation are solved for simultaneously, as opposed to first rotation and then translation. His approach minimizes an objective function which is the weighted

sum of the squares of the perpendicular distances between the projected 3D object lines and the corresponding image lines, where either the projected object lines or the image lines are infinitely extended. The objective function is minimized using an iterative technique adapted from Horn [76]. This method linearizes the objective function about the current estimate for the object's pose, and then on each iteration, linear adjustments to the pose are found that minimize this function.

Phong et. al. [118] compute object pose using both point and line correspondences. By representing pose using a dual number quaternion, each feature correspondence gives rise to two quadratic equations whose value is zero at a solution. They use a trust region optimization method to minimize the sum of squares of these quadratic equations. The trust region method is similar to Newton's method, except that each new solution estimate is required to lie within a given dynamically determined radius of the previous estimate.

2.5 Object Recognition: Correspondence and Pose

Approaches to object recognition can broadly be classified as model-based or appearance-based. *Model-based* approaches model objects by sets of features (e.g., points, lines, texture regions, etc.) and their geometric relations and then attempt to determine a geometric transformation of a model that allows it to be *matched* (or *aligned*, or *registered*) with part of an image. *Appearance-based* methods (e.g., [150, 80, 110, 112]), in contrast, model objects by sets of images (2D arrays of pixels) for many different viewpoints and illumination conditions and then attempt to find

a model image that matches the input image. (Because models are used in both of these approaches, perhaps *feature-based* would be a better designation for the class of model-based approaches, but model-based is the standard.) The advantages of model-based approaches is that they allow for a very compact representation of objects, and the algorithms tend to be more robust to partial occlusion of objects and to changes in illumination. Appearance-based approaches, in contrast, have the advantages that searching in image space is generally simpler than searching through pose space or correspondence space, and modeling objects as sets of images is simple in principle, requiring no feature extraction from the model or image. Because this dissertation is concerned with model-based object recognition using geometric features, we do not discuss appearance-based approaches further.

Some model-based approaches to 3D object recognition do not directly match 3D models to image features, but instead match 2D models to the image features. In these cases, 3D objects may be represented with a single 3D model or with a number of view-dependent 2D models. In the case that objects are represented with a single 3D model, the 3D model will be projected to a 2D model for a particular view before being matched to an image. These approaches still have the advantages of compact representation and robustness to partial occlusion, but require searching over a set of views to find models that match. We call these approaches *view-based*; these should not be confused with the appearance-based methods discussed above.

The work of Roberts [126] was one of the first to describe a model-based method to recognized 3D polyhedral objects from 2D perspective images. In his system, the greyscale image is first mapped into a set of 2D lines using a process of

edge detection, edge linking, least-squares line fitting, and line filling and merging. These line segments were then grouped into polygons. Correspondences were formed between polygons in the image and polygonal faces of models when the image polygon had the same number of sides as a model face; these correspond to nonoccluded faces of the model. Then, vertices of image polygons were matched to model vertices with topologically similar surrounding polygons. From these point correspondences, a 3D to 2D affine transformation was computed that maps the model into the image and this was used to verify that the edges of the model polygons aligned with the edges of the corresponding image polygons. The basic steps of Roberts' approach – detect features, hypothesize correspondences, calculate pose from correspondences, and verify models – are still used today in many feature-based approaches to object recognition.

Knowledge-based approaches to object recognition (e.g., [10, 19, 141]) use object and domain-specific knowledge in the recognition process. Contextual scene knowledge may be used to drive the low-level image processing as well as the high-level interpretation of the scene. For example, if one knows that an image is an overhead view of an airport, then special modules can be used to detect runways and planes; a top-down and bottom-up control architecture will look for runways wherever planes are detected, and planes wherever runways are detected. Other sensor information, such as color, stereo, motion sequences, acoustic, and 3D range, can also be integrated into the recognition process. These types of approaches will be better able cope with the difficult real-world environments that many vision applications are required to operate in. The general pose and correspondence algorithms

that we discuss below may be viewed as one component of these more complete approaches.

2.5.1 Hypothesize-and-Test Approaches

Almost all approaches to model-based object recognition that match geometric features take what is called a "hypothesize-and-test" approach [51]. In the "hypothesize" step, hypotheses for correspondences between sets of image features and sets of model features are used to generate hypotheses for the pose of objects relative to the camera. These pose hypotheses are then used in the "test" step to project the object models back into the image; the projections that are sufficiently similar to the image "verify" the presence of objects in the scene. The main distinguishing feature of these model-based methods is how the hypotheses are generated. Hypothesize-and-test approaches can be loosely grouped into three categories: alignment methods, indexing methods, and pose clustering methods.

2.5.1.1 Alignment Methods

Alignment methods hypothesize sufficiently large sets of model-to-image feature correspondences which enable an object's pose to be calculated. From such a set of correspondences, the pose of the model is computed that aligns the corresponding features and then the remainder of the model is projected into the image and its similarity with the image is measured.

When three model-to-image feature correspondences are used to instantiate an

object's pose (as discussed in Section 2.3) and there are no constraints on which of the m model feature may match which of the n image features, then there are $\binom{m}{3}\binom{n}{3}3!$ possible sets of correspondences that can be examined (each of which can give rise to 4 different poses for the object). It would be intractable to try to examine all of these correspondence sets for all but very small inputs (small m and n): assuming that a pose can be tested (verified) against the remaining image points in $\mathcal{O}(m \log n)$ time[1], the complexity of this naive alignment algorithm is $\mathcal{O}(m^4 n^3 \log n)$. Geometric constraints must be used to limit the size of this search space.

A number of general-purpose robust parameter estimation methods have been developed [49, 128, 146] that can handle gross outliers in the input data. In the object recognition problem, a gross outlier is an incorrect correspondence between a model feature and image feature. Fischler and Bolles [49] developed the Random Sample Consensus (RANSAC) algorithm for robust parameter estimation and applied the approach to the problem of pose determination given a 3D model and a single camera image, which they coined the perspective-n-point problem. RANSAC operates by sampling a small random set of correspondences, computing the camera pose from those correspondences, and then testing if the computed pose is well supported by the full set of image points. If the computed pose is not supported by the data, then the process is repeated. Otherwise, the sampling and pose estimation procedure is repeated up to some maximum number of times. The maximum number of samples that need to be examined to guarantee with a certain probability that

[1]Given a set $\mathcal{P}$ of n 2D points, the problem of finding the point in $\mathcal{P}$ closest to a 2D query point is known as the *2D closest point problem* in computational geometry. This problem can be solved in $\mathcal{O}(\log n)$ time by searching a fast planar point location data structure constructed from a Voronoi diagram of those points [3].

at least one good set of samples is examined can be determined from the sample size (three in the case of pose estimation) and the ratio of inlier to outlier samples. This number of samples does not depend on the size of the input or on the number of outliers in the data, but only on the ratio of inliers to outliers. For problems where the probability that a random sample is an outlier is less than 0.5, the required number of samples turns out to be very small when compared to the size of the search space. However, in the pose problem, where a sample consists of random model feature and a random feature point, the chance of a random sample being an outlier is very nearly 1.0, and as shown in Appendix A, $\mathcal{O}(n^3)$ triples of correspondences must be examined in order to ensure with high probability that at least one set is free of outliers. Hence, the RANSAC algorithm when applied to simple geometric features is intractable for most practical object recognition problems.

The interpretation tree approach of Grimson and Lozano-Perez [58], which is an instance of the more general consistent labeling problem [64], addresses the problem of locating polyhedral objects from data describing the position and orientation of 3D planar patches. A tree search is used where geometric constraints are enforced between levels in the tree in order to reduce the size of the search space: at the k^{th} level of the tree, the k^{th} data patch is assigned to all model faces which are consistent with individual previous assignments. When a leaf node is reached and all pairwise constraints are satisfied, then all constraints are used to determine the global consistency of the set of matches from the root of the tree to that leaf. If the set is consistent, then the pose of the object is computed and its projection is verified against the rest of the image.

Ayache and Faugeras's HYPER system [4] uses a tree-pruning approach to locate partially occluded 2D objects in an image. Models and images are represented by sets of line segments. The method seeks the similarity transformation that best matches models to the image. Lines are represented by their midpoint, length, and orientation relative to the horizontal axis. The ten longest lines in the model are identified as "privileged" segments. Privileged line 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 needs to be visible to identify a model. The algorithm first generates 2D pose hypotheses by matching the endpoints of each privileged model line segment to the endpoints of each compatible image line segment. A model segment is *compatible* with an image segment if the angle between the model segment and its preceding neighbor is within 30° of the angle between the image segment and its preceding neighbor, and if the ratio of the lengths of the two matched lines is close to an *a priori* estimate of the scale (when this is available). A 2D similarity transformation is computed for each match. After all hypotheses are generated, they are ranked based on the compatibility of the corner angles and lengths of the matched segments, and then the best hypotheses are evaluated against the image. A hypothesis is evaluated by augmented it with additional matching model and image segments. Each model segment, starting with those closest to the first model segment, is transformed by the current pose and is matched to the image segment that minimizes a dissimilarity measure. The dissim-

ilarity between a model and image segment is a weighted sum of the difference in orientations of the two segments, the Euclidean distance between their midpoints, and the relative difference between their lengths. Each time a new match is added to the current hypothesis, the model's pose is updated by computing the similarity transformation that minimizes the weighted sum of the Euclidean distances between the midpoints of the model segments to the infinitely extended image lines. A quality measure of the augmented hypothesis is then computed which is the ratio of the sum of the lengths of the matched image segments to the total length of the transformed model segments. Hypothesis evaluation is efficiently implemented as a tree-search using this quality measure in a branch-and-bound technique: as soon as the maximum possible quality of a branch of the search becomes less than the minimum possible quality of a previously evaluated hypothesis, that branch is pruned from the search. Also, to speed up the process of finding similar image segments, the image segments are initially sorted based on their orientation. Hypothesis evaluation ends when a fixed number of hypotheses have been evaluated, or when a very high quality hypothesis has been found.

The SCERPO system of Lowe [95] recognizes 3D objects from line features in single images. It uses the technique of perceptual organization to locate groupings of image lines that are stable over wide ranges of viewpoints and which are unlikely to occur by accident; these include lines that have nearby endpoints, are parallel, or are collinear. Each perceptual grouping in the image is hypothesized to match to each structure in the object that could give rise to that type of grouping. Then, a pose estimation and verification procedure is used to either accept or reject that

match. Newton's method is used for pose estimation to minimize the least squares error between image features and the corresponding projected object features. The object-to-image projection equations are reparameterized to simplify the calculation of the partial derivatives of the error function with respect to the pose parameters, and the initial guess of the object's pose is obtained from the poses that allow object features of the match set to *independently* project onto the corresponding image features. After computing an object's pose, a probabilistic ranking is used to extend the initial set of feature correspondences into a larger set which is consistent with the estimated pose: the least ambiguous correspondences are used prior to the more ambiguous correspondences. The correspondences and computed pose are accepted as correct when this set of consistent correspondences is sufficiently large.

Huttenlocher and Ullman's [79] alignment approach uses an affine (scaled orthographic) approximation to perspective projection to recognize 3D objects from single 2D images. They use point features which consist of corners and inflection points of image and model contours. The image contours are generated using the Canny edge detector [21] and edge linking. They show that a triple of corresponding model and image points determines two possible model poses (under affine projection) which differ by a reflection about the plane containing the three model points. When the model and image features also include orientation information (in addition to position), a third feature point can be inferred from two features, and it is shown that a pair of corresponding points determines four possible model poses. All possible pairs of two model and image features are used to solve for possible poses. Each pose is verified by projecting the edge contours of the model into the

image and comparing the projected edges with the nearby image edges. Poses which project sufficiently many model edges close to image edges of similar orientation are returned by the algorithm.

A number of authors (e.g., [5, 18, 23]) use linear programming and linear constraints from model-to-image point correspondences to find transformations of the model that satisfy a maximum number of these constraints. Breuel [17, 18] formulates the object recognition problem as one of finding the largest set of model and image points that can be brought into correspondence through a 2D similarity transformation (rotation, translation, and scale). He shows that error bounds on the location of a single image point with respect to the corresponding transformed model point constrain the set of compatible transformations to a four-dimensional convex polyhedron. Multiple correspondences constrain the transformation space to the intersection of the convex polyhedron of the individual correspondences. The problem of finding the largest set of corresponding model and image features then becomes one of finding the region in transformation space where the largest number of these convex polyhedron intersect. A depth-first search of transformation space is implemented that starts with a region known to include all feasible transformations. At each branch of the search, the current region is adaptively subdivided into smaller regions of transformation space. For each new region, the algorithm evaluates an upper bound on the number of convex polyhedron that may have a nonempty intersection in that region. If this upper bound is smaller than the minimum required solution, or if it is smaller than the best solution found so far, then that branch of the search is abandoned; otherwise, the region is subdivided into even smaller

regions and the process is repeated. Although not as obvious as the alignment algorithms described above, Breuel's algorithm also fits the alignment paradigm because it searches for sets of convex polyhedra – each defined by a correspondence – whose associated transformations verify the largest number of correspondences.

Extending the work of [5, 18, 23] on the bounded-error recognition problem from affine to perspective cameras, DeMenthon and Davis [37] proposed an approach using binary search by bisection of pose boxes in two 4D spaces, but it had high-order complexity. The approach taken by Jurie [85] was inspired by [37] and belongs to the same family of methods. In [85], using an affine camera model, an initial volume of pose space is guessed, and all of the correspondences compatible with this volume are first taken into account. Then the pose volume is recursively reduced until it can be viewed as a single pose. As a Gaussian error model is used, boxes of pose space are pruned not by counting the number of correspondences that are compatible with the box as in [5, 18, 23, 37], but on the basis of the probability of having an object model in the image within the range of poses defined by the box.

Unlike many previous authors, the random-start local search algorithm of Beveridge and Riseman [12] addresses the object recognition problem for full-perspective camera models. From a randomly generated initial pose, the 3D model is projected into the image and the set of all possible image-to-model line segment correspondences are generated. These correspondences are required to meet certain position and shape similarity constraints. A subset of these correspondences is then randomly selected as the starting point of the local search. A steepest descent search in the space of correspondences is used to find the locally optimal pose and corre-

spondence. The quality of a set of correspondences is a function of two error terms: the residual squared error of the image lines and the corresponding projected model lines (infinitely extended), and a term that penalizes matches which omit portions of the model from the match. At each step of the algorithm, neighboring correspondence sets (those that differ by at most one correspondence pair) are first evaluated and then ranked, the current correspondence set is replaced by the highest ranking neighboring correspondence set, and then a new pose is computed from these correspondences. A neighboring correspondence set is evaluated by using a weak-perspective pose algorithm to determine the best pose for those correspondences, and then computing the goodness of those matches given the pose just computed. Then, a full-perspective pose algorithm is used to compute the model pose after making a move to a new set of correspondences. The hybrid algorithm runs much faster than an algorithm using full-perspective at all stages, and is therefore able to examine a much larger solution space in a fixed amount of time.

2.5.1.2 Indexing Methods

The shape of the 2D projection of a 3D object depends on the geometric properties of the 3D object, the pose of that object, and on the internal parameters of the camera. In order to recognize an object, alignment and pose clustering methods must hypothesize a set of correspondences and then a pose, and then check the rest of the model against the image. Because of the variability of an object's appearance with pose, a large number of hypothesis may need to be generated in order to find a

correct one. Indexing methods for object recognition attempt to reduce this search by measuring image properties that are invariant to object pose, and then searching tables that map the invariant image properties into models and their poses [50, 109].

When perspective projection is approximated by an affine camera (see Section 2.3.2), the image of a set of coplanar object points will be related to the object points by an affine transformation [51]. Any three of these coplanar points may be chosen to form a basis for this 2D space, and the remaining points can be expressed as a linear combination of the basis vectors in terms of their affine coordinates. Lamdan et al. [90] showed that the affine coordinates of a point are invariant to affine transformations provided the same points are used to form the basis vectors. When homogeneous coordinates are used, the cross ratio of a set of four collinear points is invariant to perspective projection [91]. A wide range of invariants exist for various combinations of points, lines, and conics in the plane [109]. These are useful when simplified camera models are appropriate and objects contain planar faces. However, Burns et al. [16] have shown that there are no pose-invariant image features for any number of 3D object points in general position under perspective projection or weak perspective projection. Still, invariants have been successfully used in a number of indexing approaches to object recognition.

The geometric hashing method of Lamdan et al. [89] uses affine invariant features of planar point sets to vote for object hypotheses. Unlike other hypothesize-and-test methods, a single set of features in geometric hashing can simultaneously vote for any number of hypotheses. There is an off-line and an on-line phase to geometric hashing. In the off-line phase, the object models are analyzed in order

to build a hash table. For pair of model basis vectors formed by a triple of model points, the affine coordinates of every other model point are computed and are used as an index to insert the model and basis points into the hash table. The on-line phase of the approach consists of selecting triples of image points to form a pair of basis vectors and then computing the affine coordinates of every other image point relative to that basis. The affine coordinates are used to look up in the hash table the identities of objects and basis triples that may generate the given coordinates, and votes are cast for these objects and basis points. Triples of image points that don't correspond to points on a single object should generate random votes, whereas triples of image points that do correspond to points on a single object should generate large numbers of votes (one for all other points on the object) for a single object and basis. After all triples of image features have been examined, the objects and basis points with the most votes are considered the most likely hypothesis and are examined in more detail. Geometric hashing searches over sets of image features, but does not need to search over sets of corresponding model features. For models with m feature points and an image with n points, this 2D recognition algorithm has complexity $O(m^4)$ for the off-line phase and $O(n^4)$ for the on-line phase.

Lamdan [89] extends geometric hashing to the cases of line features and 3D objects. For example, 3D polyhedral objects may be recognized from 2D images (generated by an affine camera) by establishing a correspondence between a set of points on some planar section of the object and a set of their projections in the image. Once the pose of a planar section of a rigid object is determined, the pose of the entire object is determined. Hashing is still based on coordinates of points

in a plane, but now votes are for *(model, plane, model basis)* tuples. This approach requires that there be a sufficient number of model points on any planar section of a model used for recognition, which may be difficult to achieve.

In geometric hashing, a set of image features is matched to a set of model features whenever the two sets map to the same location in the index space (the hash table). To get large clusters, the mapping from the image to the index space should be such that the images produced by a given set of model features should map to a single index in the table. This can be accomplished if one can find an invariant representation of the images of objects. But, as Burns points out, there are no pose-invariant image features for 3D point objects under perspective projection or weak perspective projection. Jacobs [83, 84], however, shows that when two index spaces are used instead of one, and a linear projection model is used, then the set of all images of a 3D point model may be represented as a pair of lines in a high-dimensional space. This leads to a simple, space-efficient approach to indexing: match the image, which is represented as a pair of high-dimensional points, to pairs of lines in the high-dimensional space. He shows that indexing becomes significantly more difficult when it is extended to oriented point features. The hash table is constructed by mapping ordered sets of connected model points into pairs of lines in a high-dimensional space; these lines are quantized in the hash table and contain pointers back to the models and their ordered points. Then, the on-line recognition process uses the hash table to map ordered sets of connected image points to models and their ordered point sets. Because each group of image points matches only a small number of model groups, each of these hypotheses is compared to the image

for verification.

Beis and Lowe [9] have developed an indexing algorithm for 3D object recognition that uses non-invariant image features. High-dimension feature vectors are generally used in order to allow sufficient discrimination among large numbers of objects. Unlike approaches that use true viewpoint-invariant features, non-invariant image features may be mapped to entries in the index table that are different from the model features used to create the index table. Because of this, a nearest-neighbor search of the index table is needed in order to determine which object and pose hypotheses to vote for. They use a kd-tree for the index table, instead of the standard hash table, because a nearest-neighbor search can be performed in a kd-tree in time that is logarithmic in the dimension of the index, but requires time that is exponential in the dimension of the index when the search is performed in a hash table. The off-line stage analyzes sample images of all objects over a range of viewpoints. Coterminating and parallel line segments are extracted and the angles between adjacent segments and their length ratios are used to generate multi-dimensional feature vectors. These features are partially viewpoint invariant: they are invariant to translation and rotation in the image plane. The corresponding object and pose is stored for each feature vector using a separate kd-tree for each type of feature. In most of their experiments, 240 views of the models were used for training, with up to 10 models. During the on-line recognition stage, an approximate k-nearest neighbor search is performed in the kd-tree to find the objects and poses with similar features. The highest probability hypotheses are the nearest neighbors. The probability of each object-to-image match is computed based on the number of features and distances

between them. The probability that an image feature corresponds to a particular object feature in a particular pose is proportional to the fraction of training samples in a small volume centered at the current image feature that were generated by the hypothesized object feature. After estimating these probabilities, hypotheses are sorted according to the likelihood of being correct. Hypotheses are verified, from most likely to least likely, iteratively using two steps: the pose is calculated using the current set of matching features and then the matches are extended to include additional image to model correspondences. This process is repeated until no more matches are added to the match set.

2.5.1.3 Pose Clustering Methods

Pose clustering is a method of object recognition that involves the accumulation of low-level evidence in support of various poses, followed by a peak detection step that selects the poses with the most support. This works because sets of correspondences containing outliers tend to vote for random poses, while sets of correspondence containing only inliers, even though there are many fewer of these, all vote for approximately the same pose: correct poses are likely to have more votes than any random incorrect pose.

The best know pose clustering algorithm is the generalized Hough transform [6, 35, 81, 138, 142, 70] which can be used to detect arbitrary shapes. Originally, the Hough transform was developed to detect lines and analytically defined curves in 2D point sets [77, 44]. Evidence is accumulated into an accumulator array that

quantizes the space of all possible model parameters. For every pair of image feature and model, the range of model parameters that could have generated the image feature is analytically determined and used to increment the appropriate bins in the accumulator array. Then, by searching the accumulator array for peaks, the model parameters that best explain the image features are found. The model parameters corresponding to these peaks should be further analyzed to verify the presence of the model. This approach was later extended to the problem of detecting arbitrary 2D shapes (represented by sets of 2D boundary points) that have undergone geometric transformations including translation, rotation, and scale changes [6, 35, 138, 81]. The approach is similar, except that the shape of a model is described by a table that maps feature points to the ranges of model parameters that could possibly generate the feature points. The feature points for both the original and generalized Hough transform are usually the 2D position and orientation of the boundary points of the shapes.

The pose clustering technique has been applied to the problem of recognizing 3D objects from 2D images [134, 139, 144, 92, 114]. Now, instead of representing objects as collections of boundary points, objects are typically represented using 3D wire-frame models. Generating candidate poses during the evidence accumulation phase of the algorithm entails hypothesizing correspondences between model and image features and then computing the poses that make those correspondences possible. Allowing for a full 6-dimensional rigid transformation of the model would require a 6-dimensional accumulator array, which is an impractical memory requirement for any sufficiently fine quantization of pose space. Various approaches

have been used to reduce this memory requirement, including hierarchical clustering [35, 138], sequential clustering in orthogonal subspaces [7, 144, 92, 114], and non-binning methods [140] similar to k-means clustering [45]. These methods, however, are not guaranteed to find the largest clusters in the entire pose space.

The generalized Hough transform is robust to image clutter, partial object occlusion, and measurement noise. However, it requires a lot of storage and computation. Because standard pose clustering considers all possible correspondences, it can be very slow. For m model features and n image features, and correspondences defined by three corresponding image-to-model points, there are $\mathcal{O}(m^3n^3)$ correspondence triples that must be considered. For m and n large, it would be impractical to examine all of these correspondence triples. These methods can also require an excessive amount of memory, especially when discretizing a 6-dimensional pose space to a sufficient resolution. Two methods that have been proposed to reduce this problem are coarse-to-fine clustering and decomposing the pose space into orthogonal subspaces in which clustering can be performed sequentially [35]. The problem with these optimizations is that the largest clusters in the first (coarse) clustering step do not necessarily correspond to the largest clusters in the entire pose space.

Olson [114] has shown that by using a randomized algorithm to judiciously select triples of point correspondences, the complexity of pose clustering can be reduced from $\mathcal{O}(m^3n^3)$ to $\mathcal{O}(mn^3)$. He observes that randomly selecting $\mathcal{O}(n^2)$ pairs of corresponding features ensures with high probability that at least one pair provides two correct correspondences. Then, assuming a pair of correct corresponding

features was available, the same clusters of poses as would be formed by examining all triples of correspondences can also be formed by examining only those triples of correspondences that include this correct pair of correspondences. Clusters for each of these subproblems (with a fixed pair of correspondences) can be found in $\mathcal{O}(mn)$ time. Combining the $\mathcal{O}(n^2)$-time randomized selection of correspondence pairs with the $\mathcal{O}(mn)$-time exhaustive generation of single correspondences, gives a randomized pose clustering algorithm with $\mathcal{O}(mn^3)$ time complexity.

Olson [115] uses methods of perceptual organization [129] to further improve the performance of the generalized Hough transform. The key observation that he makes is that object poses should not be estimated from sets of model-to-image point correspondences if the image points are likely to have come from different objects; since all of the model points will belong to a single object, a pose calculated from such a set of correspondences will be worthless. Based on their connectedness in an edge image, pairs of image points are identified that are likely to be images of the same object. Then, only image and model points that are likely to be from single objects are allowed to form correspondences. This heuristic, while not perfect, eliminates a large number of incorrect correspondence hypotheses while only pruning a small number of correct correspondence hypotheses. This heuristic can be applied to any of the object recognition algorithms described above.

Grimson et al. [60, 62] have estimated the probability of occurrence of false positive poses (i.e., incorrect poses that receive as much positive evidence as correct poses), as a function of measurement uncertainty, occlusion, and clutter when the alignment and pose clustering methods are used to recognize both 2D and 3D ob-

jects from point features. In pose clustering, for example, measurement uncertainty requires a given triple of correspondences to vote for the range of poses consistent with the errors present in the data. This increases the chances of producing spurious peaks in the Hough accumulator. The problem becomes even worst when clutter and occlusion are considered. They conclude that these algorithms are useful mainly for low-dimensional (i.e, 2D) matching problems, but that they do not scale well when applied to complex, cluttered scenes.

Many of the hypothesize-and-test algorithms can be optimized in a variety of ways, sometimes providing significant reduction in complexity. This problem of false positives may be ameliorated by using more descriptive image and model features to reduce the number of possible correspondences. However, there is a trade-off associated with the complexity of the image and model features used to compute pose hypothesis. Simple image features, such as isolated points and line segments, are easy to produce from an image, but place few constraints on which model feature they can correspond to. Consequently, there may be many pose hypotheses to examine. In contrast, algorithms that match higher-level or more distinctive features will spend more time locating features in images, but will have significantly fewer pose hypotheses to consider due to the greater distinctiveness of the features.

As described on page 23, Lowe [97] has developed an approach to locating distinctive image features (called SIFT features) that are invariant to changes in scale and rotation in the image plane, and partially invariant to changes in 3D viewpoint. These are used for 2D object recognition, or 3D object recognition when

multiple views of the object are trained in advance. Features from the reference images are extracted and stored in a database. Then, each feature from a new image is matched to the database feature that is closest in terms of the Euclidean distance of their feature vectors. For a given feature in the new image, if the distances of the closest and second closest database features are nearly the same, then the match is thrown out because it is unreliable. An approximate nearest neighbor search, called Best-Bin-First, is used to find the best matches for each feature. The algorithm is approximate in that it returns the nearest neighbor with high probability. The speedup obtained is about two orders of magnitude over normal kd-tree search algorithms. Because a high percent of the feature matches are outliers, a generalized Hough transform voting scheme is used to find the most probable object poses consistent with the feature matches. Each match (a *single* model-to-image correspondence) votes for a model, its 2D location, 2D orientation, and scale. For clusters of three or more matches, an affine approximation of 3D pose is computed, then the number of matches is extended, and then the pose is refined. This system works best for planar objects, or 3D objects viewed from within 30° of the training images. For larger rotations of 3D objects, training images from multiple viewpoints are required.

2.5.2 Continuation Methods

Most algorithms that perform numerical minimization of a cost function $f(\mathbf{x})$ require that $f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k)$ hold on each step of the optimization. These algo-

rithms will converge to a local optimum, but depending on the starting point $\mathbf{x}_0$, this may not be a global optimum. $\mathbf{x}_0$ must typically lie in the basin of attraction of the global optimum in order to guarantee convergence to it. For many practical problems, the number of local optima increase with the dimension of the problem [120]. To have a greater chance of converging to the global optimum from an arbitrary starting point, *continuation* methods may be used.

Continuation (or homotopy) methods of numerical optimization define an easy problem for which we know the solution, and a path between this easy problem and the hard problem that we actually want to solve [125]. The solution to the easy problem is gradually transformed into the solution of the hard problem by tracing this path. For example, we can define $h(\mathbf{x}, \lambda) = f(\mathbf{x}) - (1 - \lambda)f(\mathbf{x}_0)$. The minimization of $h(\mathbf{x}, \lambda)$ as a function of $\mathbf{x}$ with λ held fixed can then be solved for λ taking on values between 0 and 1. When $\lambda = 0$, the solution is clearly $\mathbf{x} = \mathbf{x}_0$. When $\lambda = 1$, the solution is the same as the solution to the original problem. Discussed below are a number of continuation methods that have been previously applied to pose and correspondence estimation problems.

2.5.2.1 Deterministic Annealing

Simulated annealing [86] is a stochastic optimization method inspired by annealing in metallurgy, a technique involving heating and controlled cooling of a material that allows the material to obtain a more stable (lower-energy) structure with some desirable properties, such as being harder or stronger. When simulated

annealing is used for numerical optimization, an iterative process is employed in which at each iteration step the current solution is replaced by a random nearby solution with a probability that depends on both the current temperature of the system and the energy of the new solution relative to the current solution. At higher system temperatures, higher-energy (less desirable) solutions are more likely to replace the current solution. However, even at high temperatures, lower-energy regions of the solution space are favored over higher-energy regions, and therefore will attract more provisional solutions. As the temperature is slowly lowered toward zero, the system is attracted more and more to low-energy states. The advantage of simulated annealing over non-stochastic optimization methods is that the random perturbations prevent the evolution of the solution from becoming stuck in sub-optimal local minima: transitions out of local minima are always possible as long as the system temperature is positive. Simulated annealing is known to converge to an optimal global solution given a slow enough annealing schedule [54], but it's usually too slow for most practical applications with a large number of optimization variables [82].

Simulated annealing is slow, in part, because it minimizes an energy function in a discrete variable space and so is unable to use the gradient of the energy function to speedup the search. Deterministic simulated annealing (or more concisely, deterministic annealing) addresses this problem by performing the search in an analog variable space that is slowly transformed into the original discrete variable space by the end of the search [1, 127]. In deterministic annealing, an annealing parameter (the temperature) determines the smoothness of the energy function. In

the beginning, the energy function is convex and the global optimal can easily be found. As the annealing progresses, the energy function becomes closer and closer to the unsmoothed energy function; the global optimal may move, and more and more local optima appear. The global optimal can be tracked from one step to the next by local optimization methods. Although there is no guarantee that the local optimal at the end of this trajectory will be the global optimal of the unsmoothed energy function, the final solution is expected to be good.

In deterministic annealing, an update function is used which, unlike the random updates used in stochastic simulated annealing, converges quickly to the final solution. At each temperature of the annealing schedule, an analog value is found for each variable which puts the entire system in a minimum energy state. Updating the state variables at each step can be performed either by repeatedly updating individual variables until an equilibrium is reached, or by deterministically solving a system of simultaneous equations that determines the states of all variables at the given temperature. Deterministic annealing is deterministic because it minimizes the energy function directly rather than via stochastic simulation of the system dynamics.

Like stochastic simulated annealing, deterministic simulated annealing seeks a global optimum; it is does not perform a simple greedy search to the local minimum. This is accomplished through the use of an energy function that is very smooth at high temperatures, but which is transformed into the energy function of the original discrete space problem at low temperatures. In practical problems, deterministic annealing has been shown to be much faster than simulated annealing (often by a

few orders of magnitude) and usually gives very similar solutions [1, 45].

Rangarajan et al. [124, 55] used deterministic annealing to match weighted graphs, and later extended the approach to matching two point sets of the same dimension (2D or 3D) under linear transformations. Their graduated assignment algorithm is discussed in detail in Section 2.5.2.2. Noll and von Seelen [113] use deterministic annealing for matching 2D shape models to images under 2D similarity transformations, where both the models and images are represented by sets of line segments. Liu et al. [94] have used deterministic annealing to match two sets of 3D points under 3D rigid transformations; they integrated a number of rigid motion constraints into the energy function which help the algorithm to avoid local minima arising from nonrigid motions.

2.5.2.2 Graduated Assignment

The SoftPOSIT algorithm described later in this dissertation uses a modified version of the graduated assignment algorithm developed by Gold and Rangarajan [55]. The graduated assignment algorithm was originally applied to the problem of graph matching, where one seeks a *match matrix* that defines the correspondences between nodes in two graphs, called the input and model graphs below. The optimal match matrix minimizes an energy function that depends on the compatibility of the corresponding nodes and links in the two graphs. This algorithm was later adapted to the problem of 2D and 3D point matching, where, in addition to a match matrix, a *linear* geometric transformation between the input and model was also sought

[56]. An overview of the graduated assignment algorithm is given below.

The Procrustes algorithm [57] is a well-known method used to find a similarity transformation that aligns two 2D point sets when correspondences between the two sets are known. The algorithm works as follows. First, each point set is normalized by shifting the centroids to the origin and scaling so that the sum of squared distances of points from the origin is unity. This gives the translation and scale and leaves rotation as the only unknown. The optimal rotation is then computed as the one that minimizes the sum of squared distances between the corresponding normalized points, and can be solved for analytically. The graduated assignment algorithm is an extension of the Procrustes algorithm for point sets of different sizes and where correspondences are not known in advance.

For an input graph with n nodes and a model graph with m nodes, the graduated assignment algorithm uses an $(n+1) \times (m+1)$ binary-valued *match matrix* M to represent the matches between nodes in these two graphs. A value of 1 at M_{ij} represents a match between input node i and model node j. The $(n+1)^{\text{st}}$ row and $(m+1)^{\text{st}}$ column of M are the *slack* row and column, respectively, which are used to account for missing and spurious nodes in the input graph. The goal of the graduated assignment algorithm is to find the assignment matrix M and an alignment transformation $\mathcal{A}$ that maximizes the objective function

$$E(M, \mathcal{A}) = \sum_{i=1}^{n} \sum_{j=1}^{m} M_{ij} Q_{ij}(\mathcal{A}) \tag{2.5}$$

where $Q_{ij}(\mathcal{A})$ is a measure of the compatibility of input node i with model node

j as a function of the alignment transformation $\mathcal{A}$. Usually, $Q_{ij}(\mathcal{A})$ depends on the distance between the input node $\mathbf{X}_i$ and the transformation applied to model node $\mathbf{Y}_j$ as in $Q_{ij}(\mathcal{A}) = -\left(\|\mathbf{X}_i - \mathcal{A}(\mathbf{Y}_j)\|^2 - \alpha\right)$. Here, α is a constant that biases the objective function towards matches, and it acts as a threshold on distance to determine when a correspondence must be treated as an outlier.

The process of matching the input to the model is based on three ideas: graduated nonconvexity, softassign, and sparsity. The method of *graduated nonconvexity* (also known as *deterministic annealing* or the *continuation method*) transforms the discrete search space (for a binary match matrix) into a continuous search space. The continuous analog of the match matrix is called the *assignment matrix*. The continuous space is indexed by a control parameter (denoted by β below) that determines the level of uncertainty (i.e., fuzziness) of the optimal assignment matrix, and hence the amount of smoothing implicitly applied to the energy function. The assignment matrix minimizing the energy function is tracked as this control parameter is slowly adjusted to force the continuous assignment matrix closer and closer to a binary match matrix. The result is that many poor local minima can be avoided.

Softassign is the process that maps a measure of the compatibility between the input and model nodes into a doubly stochastic assignment matrix. A *doubly stochastic matrix* is a square matrix of nonnegative real entries in which the sum of the entries in each row and the sum of the entries in each column is one. Since all rows and columns of a doubly stochastic matrix sum to one, the matrix represents a probability function for the correspondences between input and model nodes. At the end of the graduated nonconvexity process, this constraint forces each input

Algorithm 1 – SINKHORN1: Sinkhorn's original algorithm for square matrices.

repeat

// Row normalization:
$M_{ij}^{k+1} = M_{ij}^{k} / \Sigma_{s=1}^{n} M_{js}^{k}$, $1 \leq i \leq n$, $1 \leq j \leq n$.

// Column normalization:
$M_{ij}^{k+1} = M_{ij}^{k+1} / \Sigma_{s=1}^{n} M_{sk}^{k}$, $1 \leq i \leq n$, $1 \leq j \leq n$.

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

node to match at most one model node, and each model node to match at most one input node. There are two steps to softassign: the first step initializes the assignment matrix according to $M_{ij}^{0} = \exp(\beta Q_{ij})$, and the second step normalizes this matrix. To ensure that the doubly stochastic constraint is satisfied, a modified version of the Sinkhorn algorithm [135], which consists of alternating row and column normalizations, is used to perform the normalization. Sinkhorn's original algorithm is shown in Algorithm 1. The modified algorithm uses a slack row and column that are treated 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 multiple input nodes to be identified as clutter and to allow multiple model nodes to be identified as being occluded. This modified algorithm is shown in Algorithm 2.

Sinkhorn's original algorithm treats all rows and columns identically; it is designed only for square matrices without a slack row and slack column. This original algorithm does not handle outliers or occlusion, and therefore the number of input and model nodes must be identical. For this case, Sinkhorn proved [135] that

Algorithm 2 – SINKHORN2: Sinkhorn's algorithm modified for matrices with a slack row and column.

repeat

// Row normalization:
$M_{ij}^{k+1} = M_{ij}^{k} / \Sigma_{s=1}^{m+1} M_{js}^{k}$, $1 \leq i \leq n$, $1 \leq j \leq m+1$.

// Column normalization:
$M_{ij}^{k+1} = M_{ij}^{k+1} / \Sigma_{s=1}^{n+1} M_{sk}^{k}$, $1 \leq i \leq n+1$, $1 \leq j \leq m$.

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

any square matrix with positive elements is transformed into a doubly stochastic matrix by the process of alternating row and column normalizations. In the case that outliers may be present, however, Zheng and Doermann [155] proved that the modified Sinkhorn algorithm shown in Algorithm 2 applied to an $n \times m$ matrix with positive elements converges to a matrix in which each row and column, except the slack row and column, sum to one. This is ideal for handling outliers. Gold et. al. [56] show that iterated row and column normalization used in softassign produces the same results as would be obtained in solving the equivalent constrained optimization problem with Lagrange multipliers.

The concept of *sparsity* is simply that the entry in the match matrix corresponding to a pair of nodes is zero whenever the two nodes cannot match to each other. Thus the match matrix will be sparse when there are only a small number of compatible matches. Pseudocode for the graduated assignment algorithm is shown in Algorithm 3.

Use of the graduated assignment algorithm for matching problems is motivated

Algorithm 3 – **GRADUATED_ASSIGNMENT**: The graduated assignment algorithm.

---

initialize alignment $\mathcal{A}$, $\beta = \beta_0$.

while $\beta < \beta_{\text{final}}$ **do**

$Q_{ij}(\mathcal{A}) = -\left(\|\mathbf{X}_i - \mathcal{A}(\mathbf{Y}_j)\|^2 - \alpha\right)$

$M_{ij}^0 = \exp(\beta Q_{ij})$

$M_{n+1,j}^0 = M_{i,m+1}^0 = 1, \;\; j = 1, \ldots, m+1, \;\; i = 1, \ldots n+1$

$k = 0$

repeat

$M_{ij}^{k+1} = M_{ij}^k / \Sigma_{s=1}^{m+1} M_{js}^k$, $1 \leq i \leq n$, $1 \leq j \leq m+1$

$M_{ij}^{k+1} = M_{ij}^{k+1} / \Sigma_{s=1}^{n+1} M_{sk}^k$, $1 \leq i \leq n+1$, $1 \leq j \leq m$

$k = k + 1$

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

Compute the alignment $\mathcal{A}$ that maximizes the objective function in Equation 2.5.

$\beta = \beta_{\text{update}} \times \beta$

end

---

by the following. First, the softassign algorithm efficiently enforces the doubly stochastic constraint on the assignment matrix. Previously, this constraint was satisfied by inserting it into the energy function via Lagrange multipliers and then applying a gradient descent optimization algorithm [124]. Second, the deterministic annealing process allows poor local minima to be avoided. Third, the algorithm handles spurious and missing data through the use of slack rows and columns in the assignment matrix. Finally, the approach can be adapted to simultaneously solve for certain geometric transformations between the data and model.

2.5.2.3 Expectation Maximization

Although the standard expectation maximization (EM) algorithm [105] is not normally considered a continuation method, it is worth examining because it has been adapted to be a continuation method by some researchers (discussed below) and it has a number of similarities to the graduated assignment algorithm. The EM algorithm is an iterative procedure to find the maximum likelihood estimate for the parameters $\boldsymbol{\theta}$ of a model given some *observed* data $\mathbf{x}$. The problem is difficult because, in addition to depending on the observed data, the likelihood function for $\boldsymbol{\theta}$ also depends on some *unobserved* data $\mathbf{y}$. The maximum likelihood estimate for $\boldsymbol{\theta}$ is therefore $\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{^{arg\,max}} P(\mathbf{x}, \mathbf{y} \mid \boldsymbol{\theta})$ where $P(\mathbf{x}, \mathbf{y} \mid \boldsymbol{\theta})$ gives the probability of $\{\mathbf{x}, \mathbf{y}\}$ given the model parameters are $\boldsymbol{\theta}$. The EM algorithm estimates $\{\mathbf{x}, \mathbf{y}\}$ and $\boldsymbol{\theta}$ in an iterative two-step process. In the *expectation step*, given an estimate for $\boldsymbol{\theta}$ and the observed data $\mathbf{x}$, the expected value of the unobserved data $\mathbf{y}$ is computed by max-

imizing the expected value of the log-likelihood function. In the *maximization step*, using the expected value for $\mathbf{y}$ as if it were observed data, the maximum likelihood estimate of $\boldsymbol{\theta}$ is computed. These two steps are repeated until the estimates converge. Although computing the likelihood function is usually intractable because it involves summing over a combinatorial number of probability densities, it has been shown that the EM algorithm converges to a *local* maximum of the likelihood function [40].

The graduated assignment algorithm can be formulated as an EM-like algorithm in a variety of different ways. The following is one formulation in the context of aligning a set of m model points to a set of n data points. We first define the relevant variables. The model parameters $\boldsymbol{\theta}$ represent the parameters of the alignment transformation. The observed data $\mathbf{y}$ consists of the m model points and the n data points. The unobserved data consists of the $(n+1) \times (m+1)$ matrix M of data-to-model point correspondences. Given the data points and the model points, we want to estimate the alignment parameters $\boldsymbol{\theta}$ and the correspondence variables M. The EM formulation of the graduated assignment algorithm is shown in Algorithm 4.

One difference between the graduated assignment algorithm and the EM algorithm is that the graduated assignment iteration uses a parameter β that affects how much smoothing is applied in estimating the unobserved data (the correspondence variables), whereas EM does not. This allows graduated assignment to put off making hard correspondence decisions until a refined (and likely more accurate) alignment transformation has been computed. Another difference is in the expectation step: classical EM would set M^k to the correspondence matrix that maximizes

Algorithm 4 – EM_GAA: Expectation maximization formulation of the graduated assignment algorithm.

1. Let $\boldsymbol{\theta}^0$ be an initial guess for the unknown alignment $\boldsymbol{\theta}$.
2. $k = 0$. $\beta = \beta_0$.
3. **(Expectation Step)** M^k is set to the expected correspondence matrix given the data points, the model points, and given that $\boldsymbol{\theta} = \boldsymbol{\theta}^k$. M^k is first initialized to a matrix of weighted (by β) exponentials of the distances between the data points and model points transformed by $\boldsymbol{\theta}^k$. Then, Sinkhorn's method is applied to ensure that M^k is doubly stochastic.
4. **(Maximization Step)** $\boldsymbol{\theta}^k$ is set to the alignment transformation that minimizes the sum of the distances, weighted by M^k, between the data points and the aligned model points.
5. If the estimates have not converged, then set $k = k + 1$, $\beta = \beta_{\text{update}} \times \beta$, and go to step 3.

the log-likelihood function. Directly computing the maximum likelihood estimate of M that enforces the one-to-one correspondence constraints is a difficult problem. Instead, the graduated assignment algorithm initializes M^k to a matrix that is similar to the maximum likelihood estimate of M but which assumes there are no correspondence constraints, and then uses Sinkhorn's algorithm to enforce these constraints. Finally, in the maximization step, $\boldsymbol{\theta}^k$ will be identical to the EM algorithm's maximum likelihood estimate of $\boldsymbol{\theta}$ provided the measurement errors of the data points are independent and Gaussian distributed, and provided the objective function optimized in computing $\boldsymbol{\theta}^k$ represents a sum of Euclidean distances between data points and their corresponding transformed model points [68, p. 86-88]. Thus, the graduated assignment algorithm has a number of attributes in common with the EM algorithm.

Dellaert et al. [36] use the EM algorithm to solve the structure from motion problem [78] when point correspondences between frames of an image sequence are not known in advance. Although the application is different from ours, their approach has some similarities to the graduated assignment algorithm that is used in our algorithms. In the context of the EM algorithm as described above, the model parameters consists of the 3D scene points and the camera pose for each frame in the sequence; the observed data consists of the positions of the image points in each frame; and the unobserved data are variables that map each image point in each frame to one of the scene points. A limitation of their approach is that they assume that there is no occlusion, no spurious image points, and that all scene points are seen in each image. The EM algorithm starts with an initial guess for the scene structure and camera motion. The expectation step calculates the image-to-scene correspondences (for all frames) that maximizes the log likelihood function. In order to impose the constraint of one-to-one matches between image and scene points, a Monte Carlo sampling method (the Metropolis algorithm [8]) is used to generate samples of valid correspondences, and from these the probabilities of individual correspondences are estimated. In the maximization step, first the previously estimated correspondence probabilities are used to generate a "virtual" image of each scene point for each frame through a weighted average of the corresponding image points. Then, a new estimate of scene structure and motion is found by minimizing the weighted reprojection error over all frames for each scene point and its virtual image points. To avoid getting stuck in shallow local minima, annealing is integrated into the EM algorithm by using a large value for a noise parameter at the start of

the algorithm and then slowly decreasing its value at each iteration.

Chapter 3

The SoftPOSIT Algorithm

This chapter describes the *SoftPOSIT* algorithm for solving the model-to-image registration problem for models and images consisting of point features; the model features are 3D and the image features, which are assumed to have been generated by a perspective camera, are 2D. Chapters 4 and 5 describe two variations of the SoftPOSIT algorithm for line features. The SoftPOSIT algorithm integrates an iterative pose estimation technique called POSIT, developed by DeMenthon and Davis [38], and an iterative correspondence assignment technique called the graduated assignment algorithm (discussed in Section 2.5.2.2), developed by Gold and Rangarajan [55, 56], 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. Each of the components of the SoftPOSIT algorithm are described in detail below.

3.1 The POSIT Algorithm

The POSIT algorithm [38] computes an object's pose given a set of corresponding 2D image and 3D object points. We summarize this algorithm below in

its original form with known correspondences, and then present a variant of the algorithm, still with known correspondences, using the closed form minimization of an objective function. It is this objective function which is modified in the next section to analytically characterize the global pose-correspondence problem (i.e., without known correspondences) in a single equation.

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

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

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

where $\boldsymbol{P_0P} = (X, Y, Z)^T$ is the vector from P_0 to P. The homogeneous image point coordinates are defined up to a multiplicative constant; therefore the validity of the equality is not affected if we multiply all the elements of the perspective projection matrix by the same term $1/T_z$. We also introduce the scale factor $s = f/T_z$. We obtain

$$\begin{bmatrix} wx \\ wy \end{bmatrix} = \begin{bmatrix} s\mathbf{R}_1^T & sT_x \\ s\mathbf{R}_2^T & sT_y \end{bmatrix} \begin{bmatrix} \boldsymbol{P_0P} \\ 1 \end{bmatrix}. \tag{3.1}$$

with

$$w = \mathbf{R}_3 \cdot \boldsymbol{P_0P}/T_z + 1. \tag{3.2}$$

In the expression for w the dot product $\mathbf{R}_3 \cdot \boldsymbol{P_0P}$ represents the projection of the vector $\boldsymbol{P_0P}$ onto the optical axis of the camera. Indeed, in the world coordinate system where P is defined, $\mathbf{R}_3$ is the unit vector of the optical axis. When the depth range of the model along the optical axis of the camera is small with respect to the model distance, $\mathbf{R}_3 \cdot \boldsymbol{P_0P}$ is small with respect to T_z, and therefore w is close to one. In this case, perspective projection gives results that are similar to the following transformation:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} s\mathbf{R}_1^T & sT_x \\ s\mathbf{R}_2^T & sT_y \end{bmatrix} \begin{bmatrix} \boldsymbol{P_0P} \\ 1 \end{bmatrix}. \tag{3.3}$$

This expression defines the *scaled orthographic* projection p' of the 3D point P. The factor s is the scaling factor of this scaled orthographic projection. When $s = 1$, this equation expresses a transformation of points from a world coordinate system

to a camera coordinate system, and uses two of the three world point coordinates in determining the image coordinates: this is the definition of a pure orthographic projection. With a factor s different from one, this image is scaled and approximates a perspective image because the scaling is inversely proportional to the distance T_z from the camera center of projection to the object origin P_0.

The general perspective equation (3.1) can be rewritten as

$$\begin{bmatrix} X & Y & Z & 1 \end{bmatrix} \begin{bmatrix} s\mathbf{R}_1 & s\mathbf{R}_2 \\ sT_x & sT_y \end{bmatrix} = \begin{bmatrix} wx & wy \end{bmatrix}. \tag{3.4}$$

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

$\mathbf{R}_1$ and $\mathbf{R}_2$:

$$\begin{aligned} s &= (\|s\mathbf{R}_1\| \, \|s\mathbf{R}_2\|)^{1/2} \quad \text{(geometric mean)}, \\ \mathbf{R}_1 &= (s\mathbf{R}_1)/s, \; \mathbf{R}_2 = (s\mathbf{R}_2)/s, \\ \mathbf{R}_3 &= \mathbf{R}_1 \times \mathbf{R}_2. \end{aligned} \tag{3.5}$$

$\mathbf{T}$ is then easily found from s, sT_x, and sT_y:

$$T_x = (sT_x)/s, \; T_y = (sT_y)/s, \; T_z = f/s. \tag{3.6}$$

An additional intermediary step that improves the performance and quality of the results consists of using the unit vectors $\mathbf{R}'_1$ and $\mathbf{R}'_2$ that are mutually perpendicular and closest to $\mathbf{R}_1$ and $\mathbf{R}_2$ in the least square sense. These vectors can be found by singular value decomposition (SVD) (see the Matlab code in [39]).

How can we compute the w_k components required to compute the right-hand side rows $(w_k x_k, w_k y_k)$ corresponding to image point p_k? We saw that setting $w_k = 1$ for every point is a good first step because it amounts to solving the problem with a scaled orthographic model of projection. Once we have the pose result for this first step, we can compute better estimates for the w_k using equation (3.2). Then we can solve the system of equations (3.4) again to obtain a refined pose. This process is repeated, and the iteration is stopped when the process becomes stationary.

The POSIT algorithm is usually faster than competing linear algorithms. Even though it is iterative, each step of the iteration is very fast, having only to invert a 4×4 matrix and perform some other simple matrix algebra, and the iteration typically converges in 3-5 steps. Although the direct linear transform algorithm [67]

has to invert a $2m \times 12$ matrix only one time (for m feature correspondences), it is typically slower than POSIT. In comparison to nonlinear methods, POSIT is faster and doesn't need an initial guess.

3.2 Geometry and Objective Function

We now look at a geometric interpretation of this method in order to propose a variant using an objective function. As shown in Figure 3.1, consider a pinhole camera with center of projection at O, optical axis aligned with Oz, an image plane Π at distance f from O, and an image center (principal point) at c. Consider an object, the origin of its coordinate system at P_0, a point P of this object, a corresponding image point p, and the line of sight L of p. The image point p' is the scaled orthographic projection of object point P. The image point p'' is the scaled orthographic projection of point P_L obtained by shifting P to the line of sight of p in a direction parallel to the image plane.

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

$$\boldsymbol{cp'} = s(\mathbf{R}_1 \cdot \boldsymbol{P_0P} + T_x, \mathbf{R}_2 \cdot \boldsymbol{P_0P} + T_y).$$

In other words, the left-hand side of equation (3.4) represents the vector $\boldsymbol{cp'}$ in the image plane. One can also show that the image plane vector from c to p'' is $\boldsymbol{cp''} = (wx, wy) = w\boldsymbol{cp}$. In other words, the right-hand side of equation (3.4) represents the vector $\boldsymbol{cp''}$ in the image plane. The image point p'' can be interpreted as a correction of the image point p from a perspective projection to a scaled orthographic projection

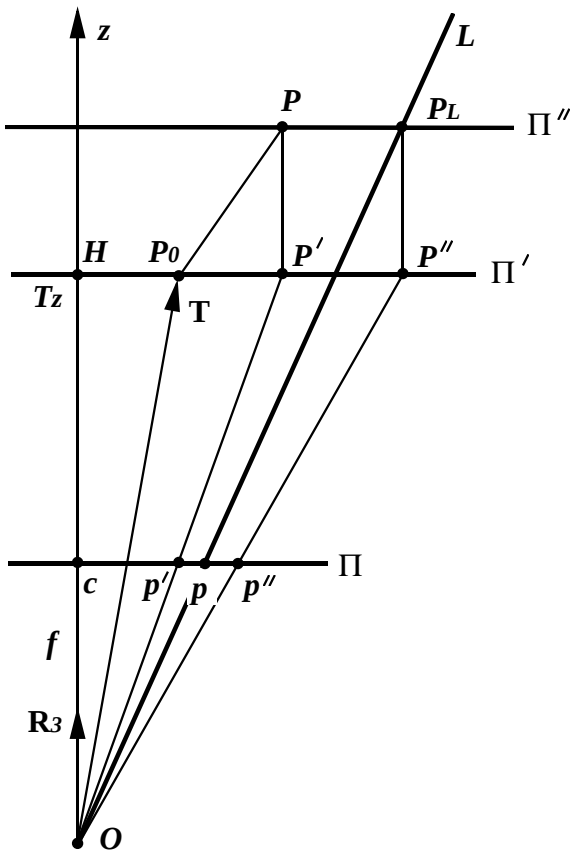


Figure 3.1: Geometric interpretation of the POSIT computation. Image point p', the scaled orthographic projection of world point P, is computed by one side of the POSIT equations. Image point p'', the scaled orthographic projection of point $P_L$on the line of sight of p, is computed by the other side of the equation. The equations are satisfied when the two points are superposed, which requires that the world point P be on the line of sight of image point p. The plane of the figure is chosen to contain the optical axis and the line of sight L. The points P_0, P, P', and p' are generally out of this plane.

of a point P_L located on the line of sight at the same distance as P. P is on the line of sight L of p if, and only if, the image points p' and p'' are superposed. Then $\boldsymbol{cp'} = \boldsymbol{cp''}$, i.e. equation (3.4) is satisfied.

When we try to match the points P_k of an object to the lines of sight L_k of image points p_k, it is unlikely that all or even any of the points will fall on their corresponding lines of sight, or equivalently that $\boldsymbol{cp'_k} = \boldsymbol{cp''_k}$ or $\boldsymbol{p'_k p''_k} = \mathbf{0}$. The least squares solution of equations (3.4) for pose enforces these constraints. Alternatively, we can minimize a global objective function E equal to the sum of the squared distances $d_k^2 = \mid \boldsymbol{p'_k p''_k} \mid^2$ between image points p'_k and p''_k:

$$\begin{aligned} E &= \textstyle\sum_k d_k^2 = \sum_k \left| \boldsymbol{cp'_k} - \boldsymbol{cp''_k} \right|^2 \\ &= \textstyle\sum_k ((\mathbf{Q}_1 \cdot \mathbf{S}_k - w_k x_k)^2 + (\mathbf{Q}_2 \cdot \mathbf{S}_k - w_k y_k)^2) \end{aligned} \tag{3.7}$$

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

$$\begin{aligned} &\mathbf{Q}_1 = (M_1, M_2, M_3, M_4) = s(\mathbf{R}_1, T_x), \\ &\mathbf{Q}_2 = (N_1, N_2, N_3, N_4) = s(\mathbf{R}_2, T_y), \\ &\mathbf{S}_k = (\boldsymbol{P_0 P_k}, 1). \end{aligned} \tag{3.8}$$

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

Referring again to Figure 3.1, notice that $\boldsymbol{p'p''} = s\boldsymbol{P'P''} = s\boldsymbol{PP_L}$. Therefore minimizing this objective function consists of minimizing the scaled sum of squared distances of model points to lines of sight, when distances are taken along directions

parallel to the image plane.

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

1. Compute the pose vectors $\mathbf{Q}_1$ and $\mathbf{Q}_2$ assuming the terms w_k are known (equation (3.7)).

2. Compute the correction terms w_k using the pose vectors $\mathbf{Q}_1$ and $\mathbf{Q}_2$ just computed (equation (3.2)).

We now focus on the optimization of the pose vectors $\mathbf{Q}_1$ and $\mathbf{Q}_2$. The pose vectors that will minimize the objective function E at a given iteration step are those for which all the partial derivatives of the objective function with respect to the coordinates of these vectors are zero. This condition provides 4×4 linear systems for the coordinates of $\mathbf{Q}_1$ and $\mathbf{Q}_2$ whose solutions are

$$\mathbf{Q}_1 = (\sum_k \mathbf{S}_k \mathbf{S}_k^T)^{-1} (\sum_k w_k x_k \mathbf{S}_k), \tag{3.9}$$

$$\mathbf{Q}_2 = (\sum_k \mathbf{S}_k \mathbf{S}_k^T)^{-1} (\sum_k w_k y_k \mathbf{S}_k). \tag{3.10}$$

The matrix $L = (\sum_k \mathbf{S}_k \mathbf{S}_k^T)$ is a 4×4 matrix that can be precomputed. Given $\mathbf{Q}_1$ and $\mathbf{Q}_2$, the rotation matrix $\mathbf{R}$ and translation vector $\mathbf{T}$ are easily calculated from Equations 3.5, 3.6, and 3.8.

With either method, the point p'' can be viewed as the image point p "corrected" for scaled orthographic projection using w computed at the previous step of the iteration. The next iteration step finds the pose such that the scaled or-

thographic projection of each point P is as close as possible to its corrected image point.

3.3 Pose from Unknown Point Correspondences

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

$$w_k = \mathbf{R}_3 \cdot \boldsymbol{P_0 P_k}/T_z + 1. \tag{3.11}$$

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

$$\boldsymbol{cp''_{jk}} = w_k \boldsymbol{cp_j}. \tag{3.12}$$

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

$$\boldsymbol{cp'_k} = \begin{bmatrix} \mathbf{Q}_1 \cdot \mathbf{S}_k \\ \\ \mathbf{Q}_2 \cdot \mathbf{S}_k \end{bmatrix}. \tag{3.13}$$

These squared distances are

$$d_{jk}^2 = \left| \boldsymbol{p'_k p''_k} \right|^2 = (\mathbf{Q}_1 \cdot \mathbf{S}_k - w_k x_j)^2 + (\mathbf{Q}_2 \cdot \mathbf{S}_k - w_k y_j)^2, \tag{3.14}$$

where x_j and y_j are the image coordinates of the image point p_j, $\mathbf{S}_k$ is the vector $(S_{k1}, S_{k2}, S_{k3}, S_{k4})^\mathsf{T} = (\boldsymbol{P_0 P_k}, 1)^\mathsf{T}$, and $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are pose vectors introduced in the previous section and recomputed at each iteration step. The term w_k is defined by equation (3.11).

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

$$\begin{aligned} E &= \sum_{j=1}^{n} \sum_{k=1}^{m} M_{jk}\, d_{jk}^2 \\ &= \sum_{j=1}^{n} \sum_{k=1}^{m} M_{jk}((\mathbf{Q}_1 \cdot \mathbf{S}_k - w_k x_j)^2 + (\mathbf{Q}_2 \cdot \mathbf{S}_k - w_k y_j)^2) \end{aligned} \tag{3.15}$$

where the M_{jk} are weights, equal to zero or one, for each of the squared distances d_{jk}^2, and where n and m are the number of image and model points, respectively. The M_{jk} are correspondence variables that define the assignments between image and model feature points. Note that when all the assignments are well-defined, this objective function becomes equivalent to the objective function defined in equation (3.7).

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

1. Compute the correspondence variables assuming everything else is fixed (see below).

2. Compute the pose vectors $\mathbf{Q}_1$ and $\mathbf{Q}_2$assuming everything else is fixed (see below).

3. Compute the correction terms w_k using the pose vectors $\mathbf{Q}_1$ and $\mathbf{Q}_2$ just computed (as described in Section 3.2).

This iterative approach is related to the general expectation-maximization (EM) algorithm [105] (see Section 2.5.2.3). In EM, given a guess for the unknown parameters (the pose in our problem) and a set of observed data (the image points in our problem), the expected value of the unobserved variables (the correspondence matrix in our problem) is estimated. Then, given this estimate for the unobserved variables, the maximum likelihood estimate of the parameters are computed. This process is repeated until these estimates converge.

3.3.1 The Pose Problem

We now focus on finding the optimal pose vectors $\mathbf{Q}_1$ and $\mathbf{Q}_2$, assuming the correspondence variables M_{jk} are known and fixed. As in the previous section, the pose vectors that will minimize the objective function E at a given iteration step are those for which all the partial derivatives of the objective function with respect to the coordinates of these vectors are zero: $\partial E/\partial \mathbf{Q}_1 = 0$ and $\partial E/\mathbf{Q}_2 = 0$. This condition provides 4×4 linear systems for the coordinates of $\mathbf{Q}_1$ and $\mathbf{Q}_2$ whose solutions are

$$\mathbf{Q}_1 = (\sum_{k=1}^{m} M'_k \mathbf{S}_k \mathbf{S}_k^{\mathsf{T}})^{-1} (\sum_{j=1}^{n} \sum_{k=1}^{m} M_{jk} w_k x_j \mathbf{S}_k), \tag{3.16}$$

$$\mathbf{Q}_2 = (\sum_{k=1}^{m} M'_k \mathbf{S}_k \mathbf{S}_k^{\mathsf{T}})^{-1} (\sum_{j=1}^{n} \sum_{k=1}^{m} M_{jk} w_k y_j \mathbf{S}_k), \tag{3.17}$$

with $M'_k = \sum_{j=1}^{n} M_{jk}$. The terms $\mathbf{S}_k \mathbf{S}_k^{\mathsf{T}}$ are 4×4 matrices. Therefore computing $\mathbf{Q}_1$ and $\mathbf{Q}_2$ requires the inversion of a single 4×4 matrix, $L = (\sum_{k=1}^{m} M'_k \mathbf{S}_k \mathbf{S}_k^{\mathsf{T}})$, a fairly inexpensive operation (note that because the term in column k and slack row

$n+1$ (see below) is generally greater than 0, $M'_k = \sum_{j=1}^{n} M_{jk}$ is generally not equal to 1, and L generally cannot be precomputed).

3.3.2 The Correspondence Problem

We optimize the correspondence variables M_{jk} assuming that the parameters d_{jk}^2 in the expression for the objective function E are known and fixed. Our aim is to find a zero-one *assignment matrix*, $M = \{M_{jk}\}$, that explicitly specifies the matchings between a set of n image points and a set of m model points, and that minimizes the objective function E. M has one row for each of the n image points p_j and one column for each of the m model points P_k. The assignment matrix must satisfy the constraint that each image point match at most one model point, and vice versa (i.e., $\Sigma_i M_{ji} = \Sigma_i M_{ik} = 1$ for all j and k). A *slack row* $n+1$ and a *slack column* $m+1$ are added. A one in the slack column $m+1$ at row j indicates that image point p_j has not found any match among the model points. A one in the slack row $n+1$ at column k indicates that the model point P_k is not seen in the image and does not match any image points. The objective function E will be minimum if the assignment matrix M matches image and model points with the smallest distances d_{jk}^2. This problem can be solved by Gold and Rangarajan's Graduated Assignment Algorithm [55, 56]. The iteration for the assignment matrix M begins with a matrix M^0 in which element M_{jk}^0 is initialized to $\exp(-\beta(d_{jk}^2 - \alpha))$, with β very small, and with all elements in the slack row and slack column set to a small constant. The parameter α determines how far apart two points must be before considering

the points unmatchable. The continuous matrix M^0 converges toward the discrete matrix M due to two mechanisms that are used concurrently:

1. First, the matrix normalization technique due to Sinkhorn [135] (described in Section 2.5.2.2) is applied. When each row and column of a square correspondence matrix is normalized (several times, alternating) 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.*

2. The term β is increased as the iteration proceeds. As β increases and each row or column of M^0 is renormalized, the terms M^0_{jk} corresponding to the smallest d^2_{jk} tend to converge to one, while the other terms tend to converge to zero. This is a deterministic annealing process [53] known as *softmax* [14]. 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.

3.3.3 Solving for Pose and Correspondences Simultaneously

This combination of deterministic annealing and Sinkhorn's technique in an iteration loop was called the Graduated Assignment Algorithm by Gold and Rangarajan [55, 56]; this algorithm was discussed in detail in Section 2.5.2.2 in the context of graph matching. The matrix M resulting from an iteration loop that comprises these two substeps is the assignment that minimizes the global objective function $E = \sum_{j=1}^{n} \sum_{k=1}^{m} M_{jk}\, d^2_{jk}$. An outline of the SoftPOSIT algorithm is given in Algorithm 5. As this outline shows, the two substeps to compute correspondences

Algorithm 5 – SOFTPOSIT: Outline of the SoftPOSIT algorithm.

initialize the pose vectors $\mathbf{Q}_1$ and $\mathbf{Q}_2$, and the certainty factor β.

repeat

- Compute the distances d_{jk} between all pairs of image and model points.
- Compute the correspondence variables M_{jk} given d_{jk}, β, $\mathbf{Q}_1$, and $\mathbf{Q}_2$.
- Compute the pose vectors $\mathbf{Q}_1$ and $\mathbf{Q}_2$ that minimize the objective function assuming known but uncertain correspondences M_{jk}.
- Compute the scaled orthographic correction terms w_k using $\mathbf{Q}_1$ and $\mathbf{Q}_2$.
- Increase the certainty factor β.

until convergence.

are interleaved in the iteration loop of SoftPOSIT, along with the substeps that optimize the pose and correct the image points by scaled orthographic distortions. More detailed pseudocode for the SoftPOSIT algorithm is shown in Algorithm 6.

Use of the graduated assignment algorithm for matching problems is motivated by the following. First, the softassign algorithm efficiently enforces the doubly stochastic constraint on the assignment matrix. Previously, this constraint was satisfied by inserting it into the energy function via Lagrange multipliers and then applying a gradient descent optimization algorithm [124]. Second, the deterministic annealing process allows poor local minima to be avoided. Third, the algorithm handles spurious and missing data through the use of slack rows and columns in the assignment matrix. Finally, the approach can be adapted to simultaneously solve for geometric transformations between the data and model.

Algorithm 6 – SOFTPOSIT: The SoftPOSIT algorithm.

Inputs: A list of n image feature points: $p_j = (x_j, y_j)^\mathsf{T}$

A list of m world points: $\mathbf{S}_k = (X_k, Y_k, Z_k, 1)^\mathsf{T} = (\boldsymbol{P_0 P_k}, 1)^\mathsf{T}$

Outputs: Rotation matrix: $R = [\mathbf{R}_1\ \mathbf{R}_2\ \mathbf{R}_3]^\mathsf{T}$

Translation vector: $\mathbf{T} = (T_x, T_y, T_z)^\mathsf{T}$

Assignments between image and world points: $M = \{M_{jk}\}$

$\beta = \beta_0$ *($\beta_0 \approx 0.0004$ if nothing is known about the pose, larger if an initial pose can be guessed)*

$\mathbf{Q}_1$ and $\mathbf{Q}_2$ are initialized using the expected pose or a random pose within an expected range.

$w_k = 1,\ 1 \le k \le m$

repeat *(Deterministic annealing loop)*

$d_{jk}^2 = (\mathbf{Q}_1 \cdot \mathbf{S}_k - w_k x_j)^2 + (\mathbf{Q}_2 \cdot \mathbf{S}_k - w_k y_j)^2$ for $1 \le j \le n$ and $1 \le k \le m$

$M_{jk}^0 = \exp(-\beta(d_{jk}^2 - \alpha))$ for $1 \le j \le n$ and $1 \le k \le m$

$M_{j,m+1} = M_{n+1,k} = 1$ for $1 \le j \le n+1$ and $1 \le k \le m+1$

repeat *(Sinkhorn normalization)*

$M'_{jk} = M_{jk}^k / \sum_{k=1}^{m+1} M_{jk}^k$

$M_{jk}^{k+1} = M'_{jk} / \sum_{j=1}^{n+1} M'_{jk}$

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

$L = (\sum_{k=1}^{m} M'_k \mathbf{S}_k \mathbf{S}_k^\mathsf{T})$ with $M'_k = \sum_{j=1}^{n} M_{jk}$

$\mathbf{Q}_1 = L^{-1}(\sum_{j=1}^{n} \sum_{k=1}^{m} M_{jk} w_k\, x_j \mathbf{S}_k),\ \ \mathbf{Q}_2 = L^{-1}(\sum_{j=1}^{n} \sum_{k=1}^{m} M_{jk} w_k\, y_j \mathbf{S}_k)$

$s = (\|(Q_{11}, Q_{12}, Q_{13})\|\ \|(Q_{21}, Q_{22}, Q_{23})\|)^{\frac{1}{2}}$

$\mathbf{R}_1 = (Q_{11}, Q_{12}, Q_{13})/s,\ \mathbf{R}_2 = (Q_{21}, Q_{22}, Q_{23})/s,\ \mathbf{R}_3 = \mathbf{R}_1 \times \mathbf{R}_2$

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

$w_k = \mathbf{R}_3 \cdot \boldsymbol{P_0 P_k}/T_z + 1,\ 1 \le k \le m$

$\beta = \beta_{update}\beta$ *($\beta_{update} \approx 1.05$)*

until $\beta > \beta_{final}$ *($\beta_{final} \approx 0.5$)*

3.3.4 Improvements to the Sinkhorn Algorithm

Sinkhorn's original algorithm [135], **SINKHORN1**, shown in Algorithm 1, treats all rows and columns identically. It cannot make a matrix doubly stochastic when the matrix is not square, that is, when the number of image features is different from the number of model features; this is the normal case when real imagery is involved. The modified Sinkhorn algorithm, **SINKHORN2**, shown in Figure 2, uses a slack row and column that are treated 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 multiple image features to be identified as clutter and to allow multiple model features to be identified as being occluded. A problem with **SINKHORN2**, however, is the following. Suppose that the nonslack value M_{jk} at row j and column k is a maximum in both its row and column. After normalizing row j, it is possible that M_{jk} is now smaller than the slack value for column k. When column k is then normalized, M_{jk} can be made smaller than the slack value for that row. The same thing can happen when rows and columns are normalized in the opposite order. As an example, consider the initial assignments matrix

$$M_1 = \begin{bmatrix} 1.0 & 0.7 & 0.8 \\ 0.7 & 1.0 & 0.8 \\ 0.8 & 0.8 & 0.0 \end{bmatrix}$$

where all of the slack values are 0.8. The assignments corresponding to the main diagonal are consistent with the one-to-one matching constraints and are better than the non-diagonal and slack matches. **SINKHORN2** will normalize M_1 to

$$M_2 = \begin{bmatrix} 0.32 & 0.23 & 0.45 \\ 0.23 & 0.32 & 0.45 \\ 0.45 & 0.45 & 0.00 \end{bmatrix}.$$

M_2 is doubly stochastic, but now the slack assignments are greater than all of the nonslack assignments. Therefore, no assignments should be made based on M_2. This matrix is inconsistent with the original assignment matrix. 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 normalization process. 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. The annealing process, not Sinkhorn normalization, is used to enforce the one-to-one matching constraint by shifting weights within an assignment matrix. A secondary problem with **SINKHORN2** 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 described above.

We have developed a new normalization algorithm called **SINKHORN3** that minimizes undesirable weight shifting. The main difference with **SINKHORN2** is

Algorithm 7 – SINKHORN3: Sinkhorn's algorithm modified to minimize undesirable weight shifting in matrices with a slack row and column.

$i = 0$

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

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 specific slack rows)*

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

  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 specific slack columns)*

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

  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**

that in **SINKHORN3** certain slack values are adjusted after each row and column normalization: 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 algorithm **SINKHORN3** is shown in Algorithm 7. When applied to the matrix M_1 from the above example, the new algorithm produces the matrix

$$M_3 = \begin{bmatrix} 0.40 & 0.28 & 0.32 \\ 0.28 & 0.40 & 0.32 \\ 0.32 & 0.32 & 0.00 \end{bmatrix}.$$

M_3 is doubly stochastic and is much better than M_2 in terms of representing the original assignment matrix M_1.

Figure 3.2 compares the normalization properties of algorithms **SINKHORN2** and **SINKHORN3** when applied to a number of 3×3 assignment matrices. As this figure shows, there are times – when the initial assignments are ambiguous – that it is reasonable and desirable for a matrix normalization algorithm to shift the weights in an assignment matrix; **SINKHORN3** appears to behave appropriately in these cases.

SINKHORN3 by design maintains the "preferred" assignments better than **SINKHORN2**, but it's not clear if **SINKHORN3** also produces a doubly stochastic matrix. To compare the normalization quality of **SINKHORN3** to **SINKHORN2**, we define the distance that an $n \times m$ nonnegative matrix M (with slack row and column) is from being doubly stochastic by

$$d_{\mathrm{ds}}(M) = \sum_{j=1}^{n-1} \left(\sum_{k=1}^{m} M_{jk} - 1 \right)^2 + \sum_{k=1}^{m-1} \left(\sum_{j=1}^{n} M_{jk} - 1 \right)^2. \tag{3.18}$$

As shown in Figure 3.3, **SINKHORN3** does not always produce a doubly stochastic matrix, while **SINKHORN2** does, but the final result is a good approximation: the

	Original Matrix	After normalization with **SINKHORN2**	After normalization with **SINKHORN3**
a)	1.0 0.7 | 0.8 0.7 1.0 | 0.8 0.8 0.8 | 0.0	0.32 0.23 | 0.45 0.23 0.32 | 0.45 0.45 0.45 | 0.00	0.40 0.28 | 0.32 0.28 0.40 | 0.32 0.32 0.32 | 0.00
b)	0.7 0.5 | 0.6 0.5 0.1 | 0.6 0.6 0.6 | 0.0	0.29 0.32 | 0.39 0.32 0.10 | 0.59 0.39 0.59 | 0.00	0.35 0.34 | 0.30 0.32 0.09 | 0.57 0.30 0.57 | 0.00
c)	1.0 0.5 | 0.1 0.5 0.1 | 0.1 0.1 0.1 | 0.0	0.40 0.54 | 0.06 0.54 0.29 | 0.17 0.06 0.17 | 0.00	0.39 0.57 | 0.04 0.57 0.26 | 0.17 0.04 0.17 | 0.00

Figure 3.2: Comparison of the results of applying algorithms **SINKHORN2** and **SINKHORN3** to a number of 3×3 assignment matrices with slack rows and columns. (In each matrix, a horizontal and vertical line separates the slack row and column, respectively, from the rest of the matrix.) In (a), the two assignments along the diagonal of the original matrix should be preferred since all of the off-diagonal elements are less than the slack values. These assignments, however, are lost by **SINKHORN2** because all of the normalized non-slack values are less than any of the normalized slack values; **SINKHORN3** correctly maintains these diagonal assignments. In (b), the assignment at the upper left corner of the original matrix (the only assignment larger than the slack values) is lost by **SINKHORN2** because the normalized value is less than all of the normalized slack values; again, **SINKHORN3** correctly maintains this assignment. In (c), the original assignments are ambiguous because there are three values in the original matrix that are larger than slack. Even though the upper left assignment is the largest of these, both **SINKHORN2** and **SINKHORN3** normalize the matrix so that the off-diagonal assignments are preferred; this is a reasonable result.

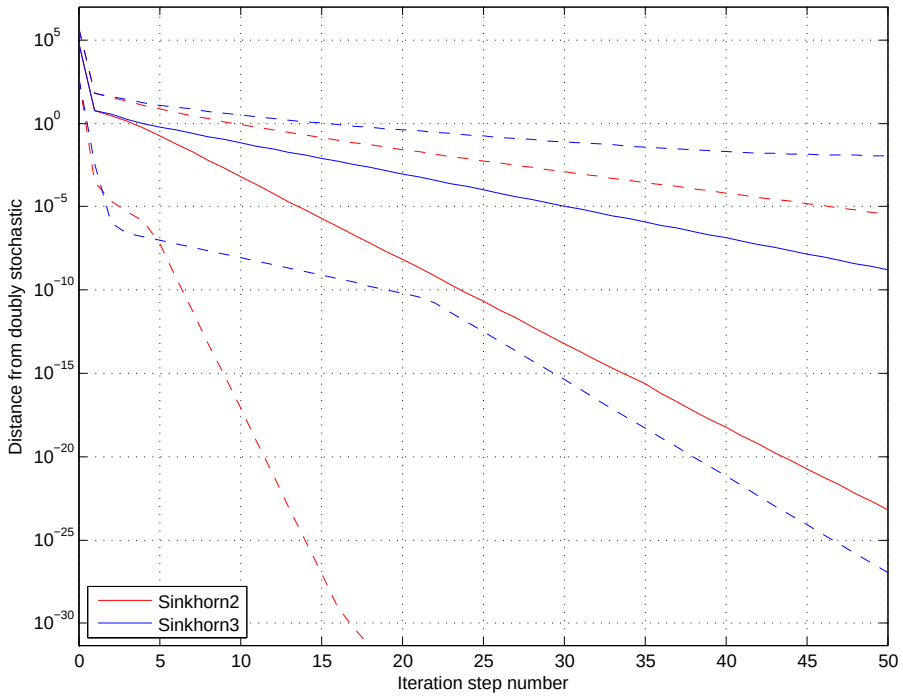


Figure 3.3: This plot compares the doubly stochastic qualities of matrices normalized with algorithm **SINKHORN2** to the same matrices normalized with algorithm **SINKHORN3**. 10000 random assignment matrices (random sizes between 10×10 and 100×100, not necessarily square, with random values) were normalized with **SINKHORN2** and **SINKHORN3** and the distance of the normalized matrix from being doubly stochastic (as determined by Equation 3.18) was measured after every set of row and column normalizations. The solid lines give the median distances, the upper dashed lines the maximum distances, and the lower dashed lines the minimum distances, for all matrices as a function of the iteration step number. The vertical axis represents the distance of the matrix from being doubly stochastic, and the horizontal axis gives the iteration step number. Results for **SINKHORN2** are shown as solid and dashed red lines in this plot, and results for **SINKHORN3** as solid and dashed blue lines. Although **SINKHORN3** does not always produce a doubly stochastic matrix, the final result is a good approximation

maximum distance from doubly stochastic after 50 normalization steps is 3.6×10^{-6} for **SINKHORN2** and 1.1×10^{-2} for **SINKHORN3**; the *median* distance from doubly stochastic after 50 normalization steps is 6.7×10^{-24} for **SINKHORN2** and 1.7×10^{-9} for **SINKHORN3**. As a consequence of this and its better nonweight-shifting property, we have observed better convergence to correct solutions when algorithm **SINKHORN3** is used instead of algorithm **SINKHORN2**. Consequently, algorithm **SINKHORN3** is used in place of **SINKHORN2** in all experiments described in this paper.

3.4 Random Start SoftPOSIT

The SoftPOSIT algorithm described above performs a deterministic annealing search starting from an initial guess for the object's pose. Because this is a local search, there is no guarantee of finding the global optimum. The probability of finding the globally optimal object pose and correspondences starting from an initial guess depends on a number of factors including the accuracy of the initial guess, the deterministic annealing cooling schedule, the number of model points, the number of image points, the number of occluded model points, the amount of clutter in the image, and the image measurement noise. A common way of searching for a global optimum, and the one taken here, is to run the search algorithm starting from a number of different initial guesses, and keep the first solution that meets a specified termination criteria. Our initial guesses range over the range of $[-\pi, \pi]$ for the three Euler rotation angles, and over a 3D space of translations known to contain the true

translation. In this section, we describe our procedure for generating initial guesses for pose when no knowledge of the correct pose is available, and then we discuss our termination criteria.

3.4.1 Generating Initial Guesses

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

However, this leads to a set of problems. First, a pseudo-random number generator will generate points that may cluster together or be widely spread apart in the parameter space. Thus we may revisit some regions of the space that have already been studied, or might miss some regions altogether. A possible solution to allow for rejection of some of the generated initial parameter values. However, this adds a layer of complexity to the software, and further does not provide a mathematical guarantee that the space is in some sense being optimally covered. Indeed since each search for correspondence and pose is relatively expensive, we would like to have a mathematical statement that allows us to make the claim that, for a given number of starting points, our starting points sample the parameter space in some optimal manner.

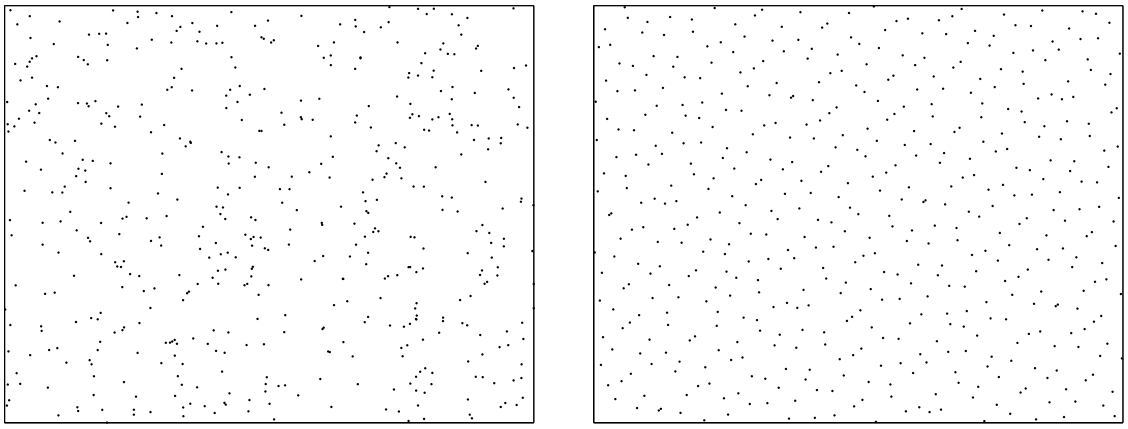
Another problem with such searches is that sometimes the minima may lie

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

Fortunately, there are a set of deterministic points that have such properties.These are the quasi-random, or low discrepancy sequences. These points are optimally self-avoiding, and uniformly space filling. Uniformity of a distribution of points can be characterized by the mathematical definition of discrepancy. Let a region with unit volume have N points distributed in it. Then, for uniform point distributions, any subregion with volume α would have αN points in it. The difference between this quantity and the actual number of points in the region is called the "discrepancy." Quasi-random sequences have low discrepancies and are also called low-discrepancy sequences. The error in uniformity for a sequence of N points in the k-dimensional unit cube is measured by its discrepancy, which is $O((\log N)^k N^{-1})$ for a quasi-random sequence, as opposed to $O((\log \log N)^{1/2} N^{-1/2})$ for a pseudo-random sequence [108].

Figure 3.4 compares the uniformity of distributions of quasi-random points and pseudo-random points. Figure 3.4a shows a set of random points generated in $(0,1)^2$ using a pseudo-random number generator. If the distribution of points were uniform one would expect that any region of area larger than $1/512$ would have at

least one point in it. As can be seen, however, many regions considerably larger than this are not sampled at all, while points in other regions form rather dense clusters, thus oversampling those regions. Figure 3.4b shows the same number of quasi-random points for the same area. These points do not clump together, and fill the spaces left by the pseudo-random points.



(a) (b)

Figure 3.4: 512 points in $(0,1)^2$ generated with (a) a pseudo-random number generator, and (b) a quasi-random number generator.

We use a standard quasi-random generator [121] to generate quasi-random 6-vectors in a unit 6D hypercube. These points are scaled to cover the expected ranges of translation and rotation.

3.4.2 Search Termination

Ideally, one would like to repeat the search from a new starting point whenever the number of model-to-image correspondences determined by the search is not maximal. With real data, however, one usually does not know what this maximal

number is. Instead, we repeat the search when the number of model points matching to image points is less than some threshold t_m. Due to occlusion and imperfect image feature extraction algorithms, not all model points will be detected as features in an image of that object. Let the fraction of detected model features be

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

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

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

We note that Grimson and Huttenlocher [61] have derived an expression for

a threshold on the number of matched model points necessary to accept a local optimum; their expression is a function of number of image and model points and of the sensor noise, and guarantees with a specified probability that the globally optimal solution has been found.

3.4.3 Early Search Termination

The deterministic annealing loop of the SoftPOSIT algorithm iterates over a range of values for the annealing parameter β. In the experiments reported here, β is initialized to $\beta_0 = 0.0004$ and is updated according to $\beta = 1.05 \times \beta$, and the annealing iteration ends when the value of β exceeds 0.5. (The iteration may end earlier if convergence is detected.) This means that the annealing loop can run for up to 147 iterations. It is usually the case that, by viewing the original image and, overlayed on top of this, the projected model points produced by SoftPOSIT, a person can determine very early on in the iteration (e.g., around iteration 30) whether or not the algorithm is going to converge to the correct pose. It is desired that the algorithm make this determination itself, so that whenever it detects that it is likely heading down an unfruitful path, it can end the current search for a local optimum and restart from a new random initial condition, thereby saving a significant amount of processing time.

A simple test is performed on *each* iteration of SoftPOSIT to determine if it should continue with the iteration or restart. At iteration i of SoftPOSIT, the match matrix $M^i = \{M^i_{j,k}\}$ is used to predict the final correspondences of model to

image points: upon convergence of SoftPOSIT, one would expect image point j to correspond to model point k if $M^i_{j,k} > M^i_{u,v}$ for all $u \neq j$ and all $v \neq k$ (however, this is not guaranteed). The number of predicted correspondences at iteration i, c_i, is just the number of pairs (j,k) that satisfy this relation. We then define the match ratio on iteration i as $r_i = c_i/(p_d K)$ where p_d is the fraction of detected model features as defined above.

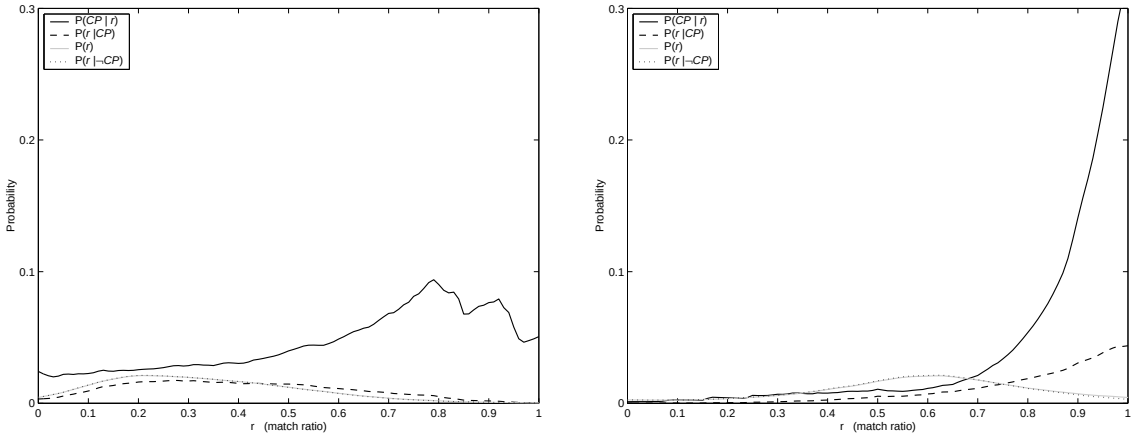
The early termination test centers around this match ratio measure. This measure is commonly used [61] at the end of a local search to determine if the current solution for correspondence and pose is good enough to end the search for the global optimum. We, however, use this metric within the local search itself. Let C denote the event that the SoftPOSIT algorithm eventually converges to the correct pose. Then, the algorithm restarts after the i^{th} iteration if $P(C \mid r_i) < \alpha P(C)$ where $0 < \alpha \leq 1$. That is, the search is restarted from a new random starting condition whenever the posterior probability of eventually finding a correct pose given r_i drops to less than some fraction of the prior probability of finding the correct pose. Notice that a separate posterior probability function is required for each iteration i because the ability to predict the eventual outcome using r_i changes as the iteration progresses. Although this test may result in the termination of some local searches which would have eventually produced a good pose, it is expected that the total time required to find a good pose will be less. Our experiments show that this is indeed be the case; we obtain a speedup by a factor of 2.

The posterior probability function for the i^{th} iteration can be computed from $P(C)$, the prior probability of finding a correct pose on one random local search,

and from $P(r_i \mid C)$ and $P(r_i \mid \overline{C})$, the probabilities of observing a particular match ratio on the i^{th} iteration given that the eventual pose is either correct or incorrect, respectively:

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

$P(C)$, $P(\overline{C})$, $P(r_i \mid C)$, and $P(r_i \mid \overline{C})$ are estimated in Monte Carlo simulations of the algorithm in which the number of model vertices and the levels of image clutter, occlusion, and noise are all varied. The details of these simulations are described in Section 3.5.1. To estimate $P(r_i \mid C)$ and $P(r_i \mid \overline{C})$, the algorithm is repeatedly run on random test data. For each test, the values of the match ratio r_i computed at each iteration are recorded. Once a SoftPOSIT iteration completes, ground truth information is used to determine whether or not the correct pose was found. If the pose is correct, then the recorded values of r_i are used to update histograms representing the probability functions $P(r_i \mid C)$; otherwise, histograms representing $P(r_i \mid \overline{C})$ are updated. Upon completing this training, the histograms are normalized. $P(C)$ is easily estimated based on the percent of the random tests that produced the correct pose. We also have $P(\overline{C}) = 1 - P(C)$. A few of these estimated probability functions are shown in Figure 3.5.



(a) (b)

Figure 3.5: Probability functions estimated for (a) the first iteration, and (b) the 31^{st} iteration, of the SoftPOSIT algorithm.

3.5 Experiments with Point Data

3.5.1 Monte Carlo Evaluation

The random-start SoftPOSIT algorithm has been extensively evaluated in Monte Carlo simulations. The simulations and the performance of the algorithm are discussed in this section. The simulations are characterized by the five parameters: n_t, m, p_d, p_c, and σ. n_t is the number of independent random trials to perform for each combination of values for the remaining four parameters. m is the number of points (vertices) in a 3D model. p_d is the probability that the image of any particular model point will be detected as a feature point in the image. p_d takes into account occlusion of the 3D model points as well as the fact that real image processing algorithms do not detect all desired feature points, even when the corresponding 3D points are not occluded. p_c is the probability that any particular image feature

point is clutter, that is, is not the image of some 3D model point. Finally, σ is the standard deviation of the normally distributed noise in the x and y coordinates of the non-clutter feature points, measured in pixels for a 1000×1000 image, generated by a simulated camera having a 37 degree field of view (focal length of 1500 pixels). The current tests were performed with $n_t = 100$, $m \in \{20, 30, 40, 50, 60, 70, 80\}$, $p_d \in \{0.4, 0.6, 0.8\}$, $p_c \in \{0.2, 0.4, 0.6\}$, and $\sigma \in \{0.5, 1.0, 2.5\}$[1]. With these parameters, there were 18,900 independent trials performed.

For each trial, a 3D model is created in which the m model vertices are randomly located in a sphere centered at the origin. Because the SoftPOSIT algorithm works with points, not with line segments, it is only the model vertices that are important to the current tests. However, to make the images produced by the algorithm easier to understand, each model vertex is connected by an edge to the two closest of the remaining model vertices. These connecting edges are not used by the SoftPOSIT algorithm. The model is then rotated into some arbitrary orientation, and translated to some random point in the view of the camera. Next, the model is projected into the image plane of the camera; each projected model point is detected with probability p_d. To those points that are detected, normally distributed, zero mean, and standard deviation σ noise is added to both the x and y coordinates of the feature points. Finally, randomly located clutter feature points are added to the true (non-clutter) feature points, so that $100 \times p_c$ percent of the total number of feature points are clutter; to achieve this, $mp_dp_c/(1 - p_c)$ clutter

[1]Because one of our main application is autonomous navigation in cities, and because image corner points of the type produced by buildings can be located with an accuracy of 1/10th of a pixel [13], these values of σ are larger than what is expected in real imagery.

points must be added. The clutter points are required to lie in the general vicinity of the true feature points. However, to prevent the clutter points from replacing missing true feature points, all clutter points must be further than $\sqrt{2}\sigma$ from any projected model point, whether or not the point was detected. Figure 3.6 shows a few examples of cluttered images of random models that are typical of those used in our experiments.

In our experiments, we consider a pose to be *good* when it allows 80% ($\rho = 0.8$ in section 3.4.2) or more of the detected model points to be matched to some image point. The number of random starts for each trial was limited to 10,000. Thus, if a good pose is not found after 10,000 starts, the algorithm gives up. Figures 3.7 and 3.8 show a number of examples of poses found by SoftPOSIT when quasi-random 6-vectors are used as the initial guesses for pose.

Figure 3.9 shows the success rate of the algorithm (percent of trials for which a good pose was found in 10,000 starts, given no knowledge of the correct pose) as a function of the number of model points for the case of $\sigma = 2.5$ and for all combinations of the parameters p_d and p_c. (The algorithm performs a little better for the cases of $\sigma = 0.5$ and $\sigma = 1.0$.) It can be seen from this figure that, for more than 92% of the different combinations of simulation parameters, a good pose is found in 90% or more of the associated trials. For the remaining 8% of the tests, a good pose is found in 75% or more of the trials. Overall, a good pose was found in 96.4% of the trials. As expected, the higher the occlusion rate (lower p_d) and the clutter rate (higher p_c), the lower the success rate. For the high-clutter tests, the success rate increases as the number of model points decreases. This is due to the

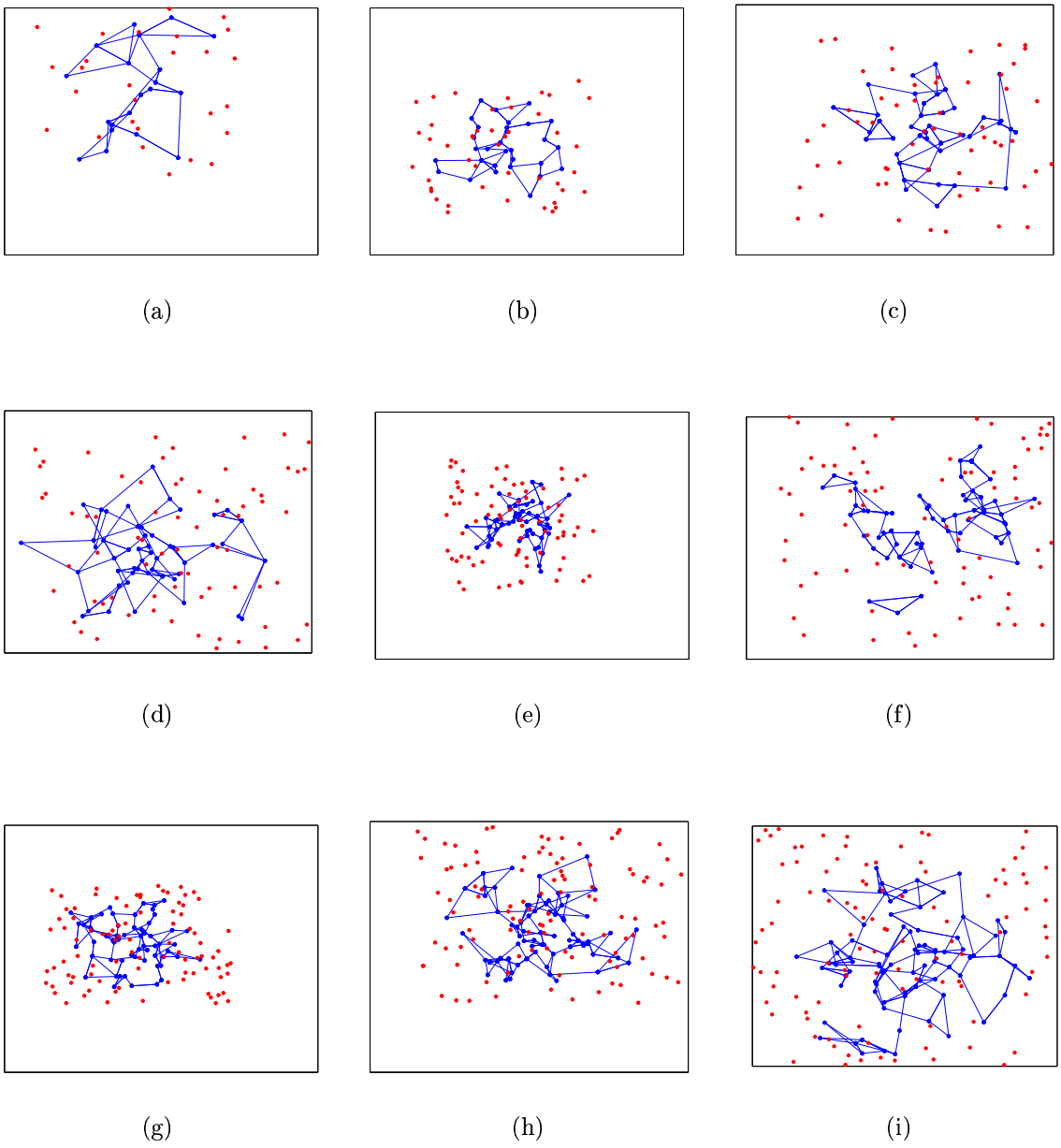


Figure 3.6: Typical images of randomly generated models and images. The blue points are projected model points and the red points are clutter points. The blue lines, which connect the model points, are included in these pictures to assist the reader in understanding the pictures; they are not used by the algorithm. The number of points in the models are 20 for (a), 30 for (b), 40 for (c), 50 for (d) and (e), 60 for (f) and (g), 70 for (h), and 80 for (i). In all cases shown here, $p_d = 1.0$ and $p_c = 0.6$. This is the best case for occlusion of the model (none), but the worst case for clutter. In the actual experiments, p_d and p_c vary.

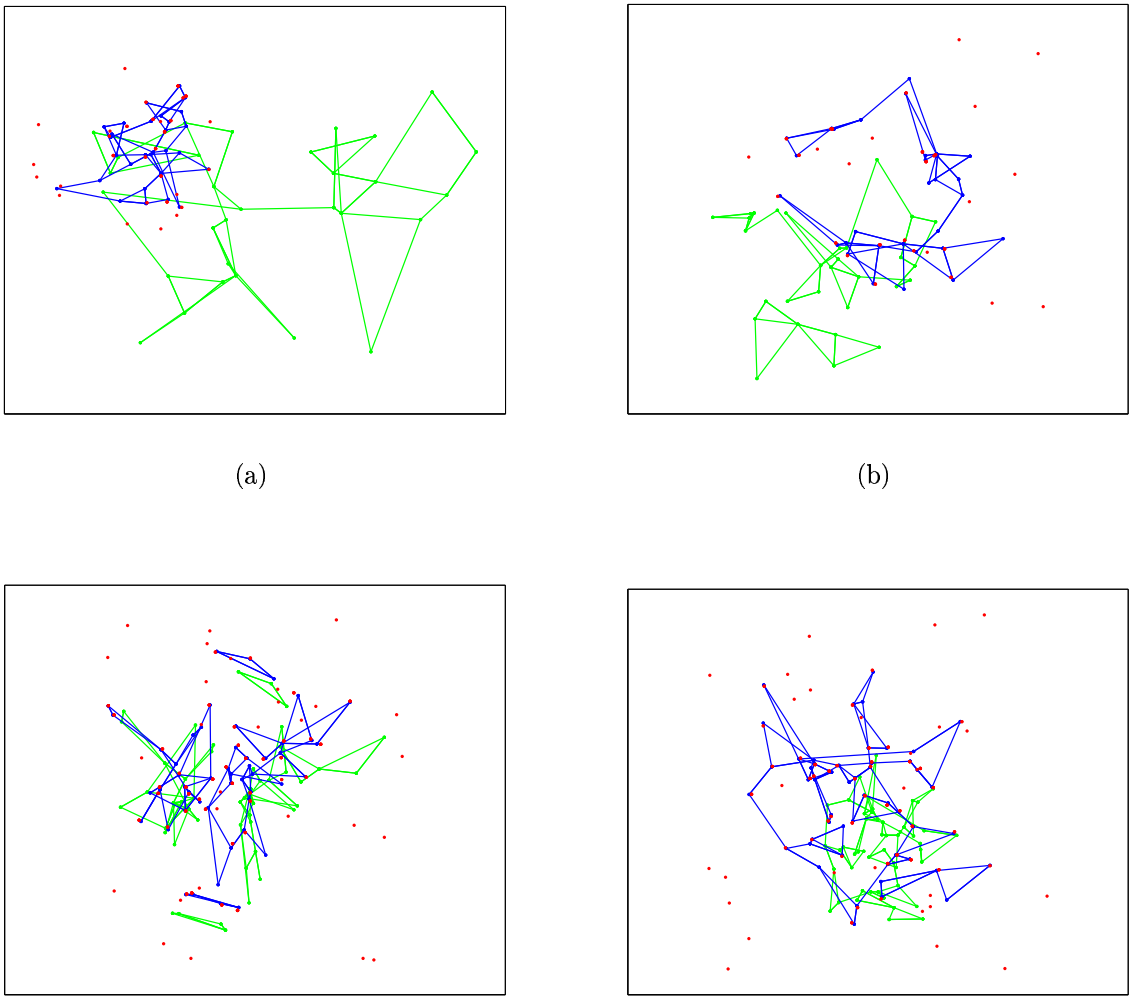


Figure 3.7: Some projected models and cluttered images for which SoftPOSIT was successful. The red points are the image points (including projected model and clutter) to which the models must be matched. The green points and lines show the projections of the models in the initial poses (random guesses) which lead to good poses being found. The blue points and lines show the projections of the models in the good poses that are found. The blue points that are not near any red point are occluded model points. Red points not near any blue point are clutter. Again, the green and blue lines are included in these pictures to assist the reader in understanding the pictures; they are not used by the algorithm. The Monte Carlo parameters for (a) and (b) are: $m = 30$, $p_d = 0.6$, $p_c = 0.4$, $\sigma = 2.5$; and, for (c) and (d): $m = 50$, $p_d = 0.8$, $p_c = 0.4$, $\sigma = 2.5$.

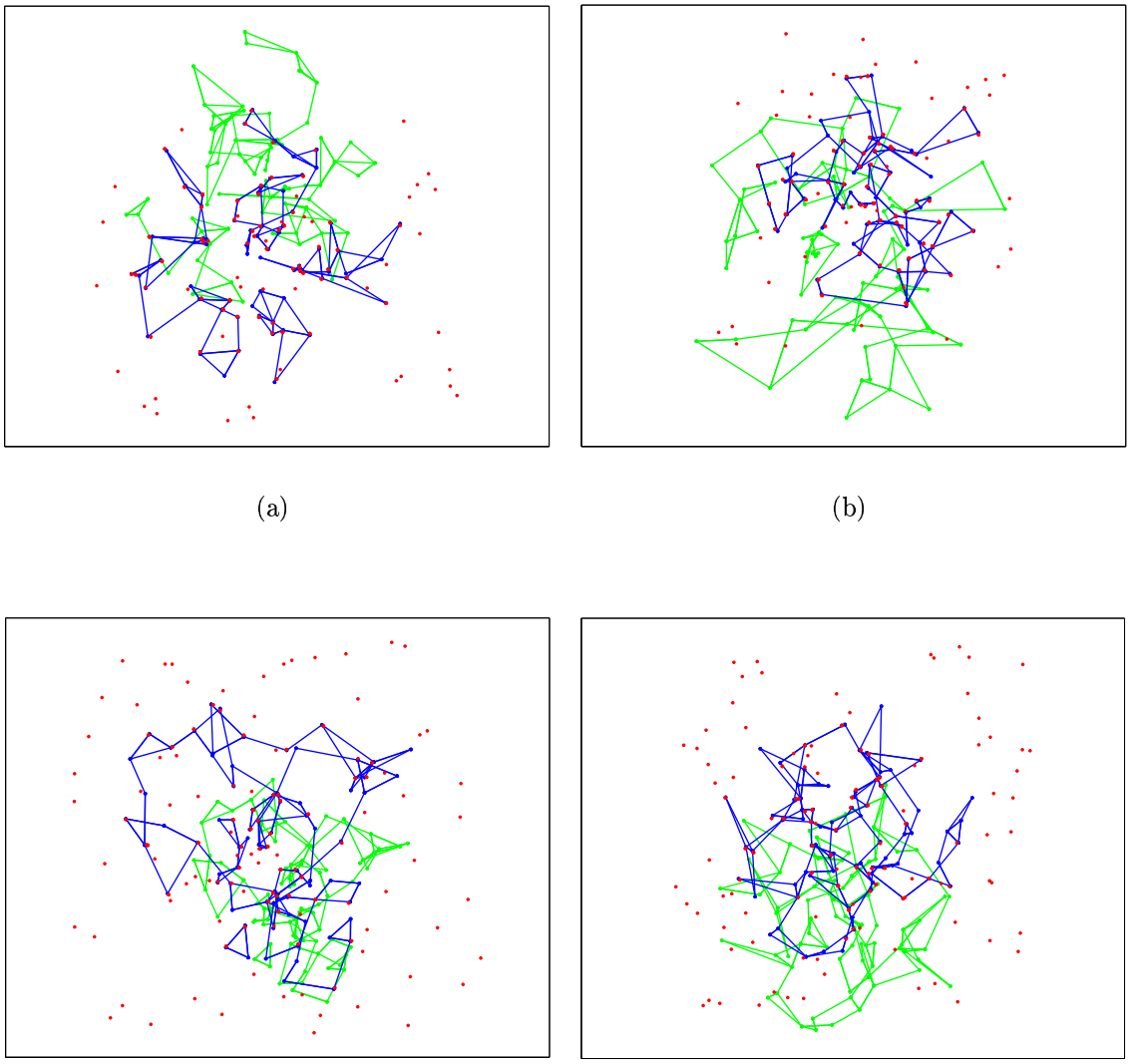


Figure 3.8: More projected models and cluttered images for which SoftPOSIT was successful. The Monte Carlo parameters for (a) and (b) are: $m = 70$, $p_d = 0.8$, $p_c = 0.4$, $\sigma = 2.5$; and, for (c) and (d): $m = 80$, $p_d = 0.6$, $p_c = 0.6$, $\sigma = 2.5$.

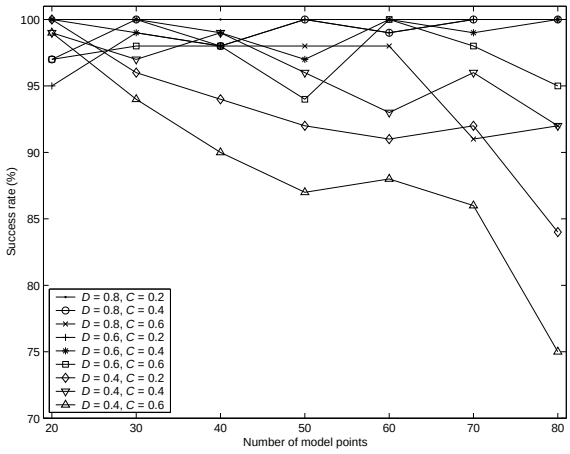
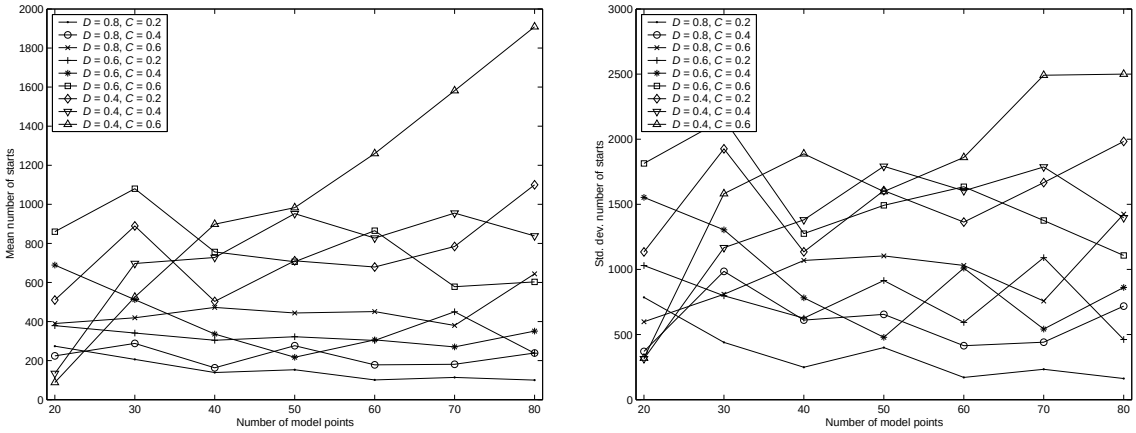


Figure 3.9: Success rate as a function of the number of model points for fixed values of p_d and p_c. (Note that p_d and p_c are denoted D and C, respectively, in the legend of this figure and in the next few figures.)

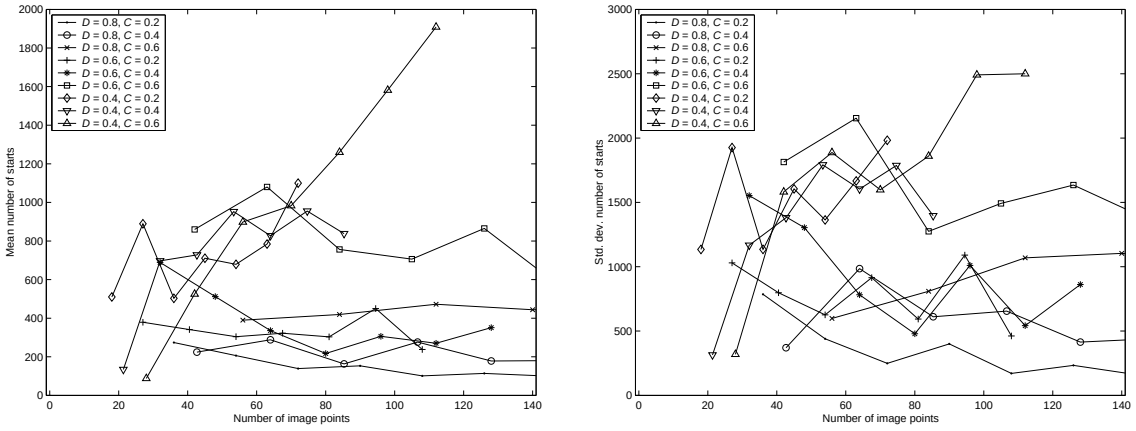
algorithm's ability to more easily match a smaller number of model points to clutter than a larger number of model points to the same level of clutter.

Figure 3.10 shows the average number of random starts required to find a good pose. These numbers generally increase with increasing image clutter and occlusion. However, for the same reason as given in the previous paragraph, the performance for small numbers of model points is better at higher levels of occlusion and clutter. Other than the highest occlusion and clutter case, the mean number of starts is about constant or very slowly increasing with increasing number of model points. Also, there does not appear to be any significant increase in the standard deviation of the number of random starts as the number of model points increases. The mean number of starts over all of the tests is approximately 500; the mean exceeds 1100 starts only in the single the hardest case. Figure 3.5.1 shows the same data but plotted as a function of the number of image points. Again, except for the two highest occlusion and clutter cases, the mean number of starts is about constant or



(a) (b)

Figure 3.10: Number of random starts required to find a good pose as a function of the number of model points for fixed values of p_d and p_c. (a) Mean. (b) Standard deviation.



(a) (b)

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

very slowly increasing as the number of image points increases.

3.5.2 Algorithm Complexity

The run-time complexity of a single invocation of SoftPOSIT is $\mathcal{O}(mn)$ where m is the number of object points and n is the number of image points. As shown in Figure 3.10, the mean number of random starts (invocations of SoftPOSIT) required to find a good pose in the worst (hardest) case, to ensure a probability of success of at least 0.95, appears to be bounded by a function that increases linearly with the size of the input; in the other cases, the mean number of random starts is approximately constant. That is, the mean number of random starts is $\mathcal{O}(n)$, assuming that $m < n$, as is normally the case. Then the run-time complexity of SoftPOSIT with random starts is $\mathcal{O}(mn^2)$. This is a factor of n better than the complexity of any published algorithm that solves the simultaneous pose and correspondence problem under a full perspective camera model.

3.5.3 Run Time Comparison

The RANSAC algorithm [49] is the best known algorithm to compute object pose given noncorresponding 3D object and 2D image points. In this section, we compare the expected run time[2] of SoftPOSIT to that of RANSAC for each of the simulated data sets discussed in Section 3.5.1.

The mean run time of SoftPOSIT on each of these data sets was recorded

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

during the Monte Carlo experiments. As will be seen below, to have run RANSAC on each of these data sets would have required a prohibitive amount of time. This was not necessary, however, since we can accurately estimate the number of random samples of the data that RANSAC will examine when solving any particular problem. The expected run time of RANSAC is then the product of that number of samples with the expected run time on one sample of that data.

The computational complexity of a pose problem depends on the three parameters m, p_d, and p_c defined in Section 3.5.1. For each combination of these three parameters, we need to determine the expected run time of RANSAC for a single sample of three object points and three image points[3] from that data. This was accomplished by running RANSAC on many random samples generated using the same set of three complexity parameters. The time per sample for a given problem complexity is estimated as the total time used by RANSAC to process those samples (excluding time for initialization) divided by the number of samples processed.

We now estimate how many samples RANSAC will examine for problems of a particular complexity. In Appendix A, we compute the probability, p, as a function of m, p_d, and p_c, that a random sample of three object points and three image points consists of three correct correspondences. Then, the number of random samples of correspondence triples that must be examined by RANSAC in order to ensure with probability z that at least one correct correspondence triple will be examined is

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

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

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

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

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

For a data set with complexity given by m, p_d, and p_c, SoftPOSIT has a given observed success rate which we denote by $z_{softPOSIT}(m, p_d, p_c)$ (see Figure 3.9). Since we did not run RANSAC on this data, we can't compare the success rates of SoftPOSIT and RANSAC for a given fixed amount of run time. However, we can compare the run time required by both to achieve the same rate of success on problems of the same complexity by estimating the run time of RANSAC when its required probability of success is $z = z_{softPOSIT}(m, p_d, p_c)$. These run times are

shown in Figures 3.11 and 3.12. From these figures, it can be seen that the RANSAC algorithm requires one to three orders of magnitude more run time than SoftPOSIT for problems with the same level of complexity in order to achieve the same level of success. Furthermore, for the majority of the complexity cases, run time as a function of input size increase at a faster rate for the RANSAC algorithms than for the SoftPOSIT algorithm. The totality of Monte Carlo experiments described in Section 3.5.1 required about 30 days for SoftPOSIT to complete. From this analysis it can be estimated that a RANSAC algorithm which examines s_1 samples would require about 19.4 *years* to complete the same experiments, and a RANSAC algorithm which examines s_2 samples would require about 4.5 years. Clearly, it would not have been practical to run RANSAC on all of these experiments.

3.5.4 Experiments with Images

3.5.4.1 Autonomous Navigation Application

The SoftPosit algorithm was applied to the problem of autonomous vehicle navigation through a city where a 3D architectural model of the city is registered to images obtained from an on-board video camera. All imagery was generated by a commercial virtual reality system. Figure 3.13 shows an image generated by this system and a world model projected into that image using the pose computed by SoftPOSIT. Image feature points are automatically located in the image by detecting corners along the boundary of bright sky regions. Because the 3D world model has over 100,000 data points, we use a rough pose estimate (as may be generated by

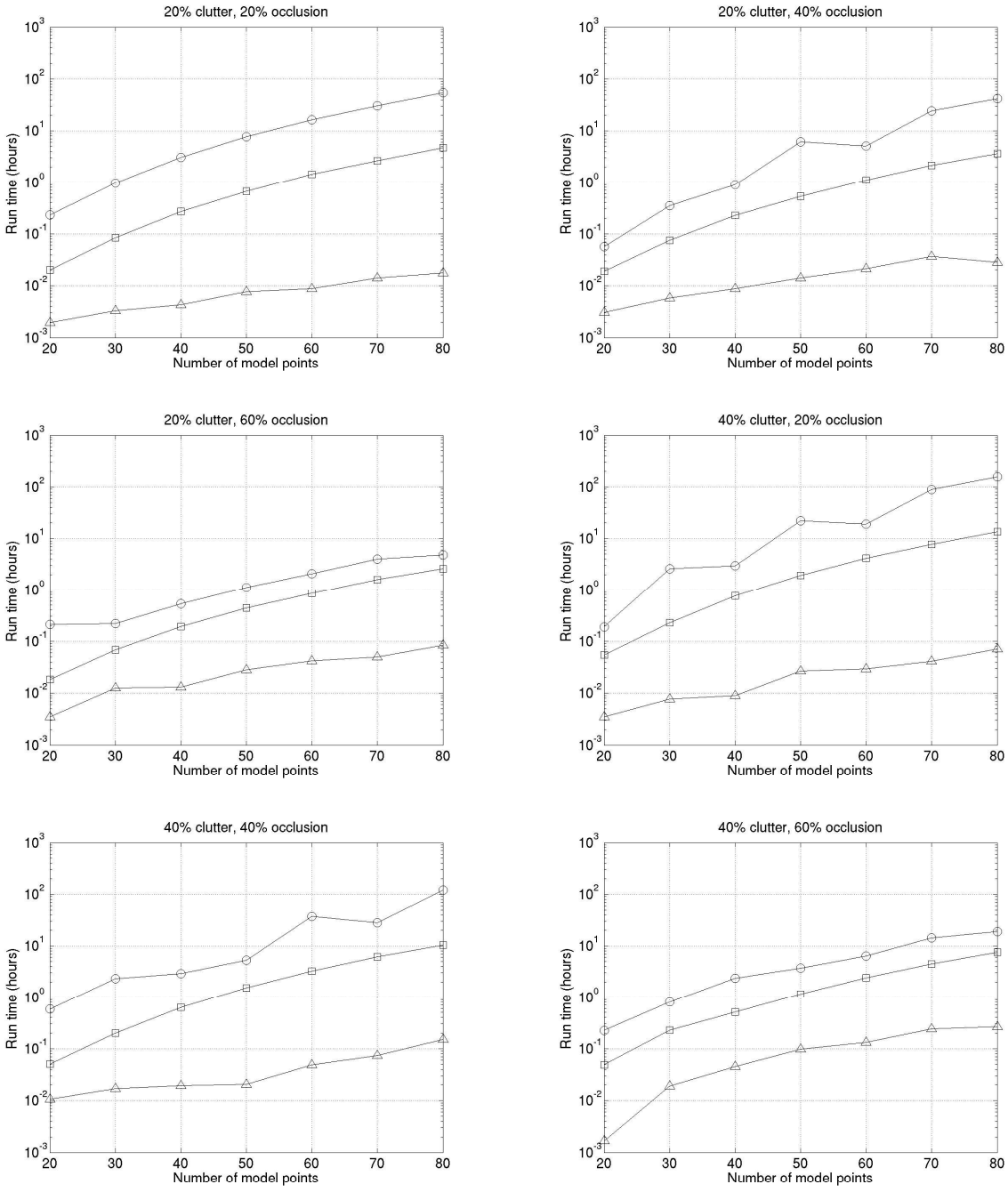


Figure 3.11: Comparison of the run times of SoftPOSIT to those of RANSAC for problems with 20-40% clutter and 20-60% object occlusion. The SoftPOSIT run times are marked with triangles. The RANSAC run times are marked with circles for the case that the number of samples is determined by s_1, and with squares for the case that the number of samples is determined by s_2.

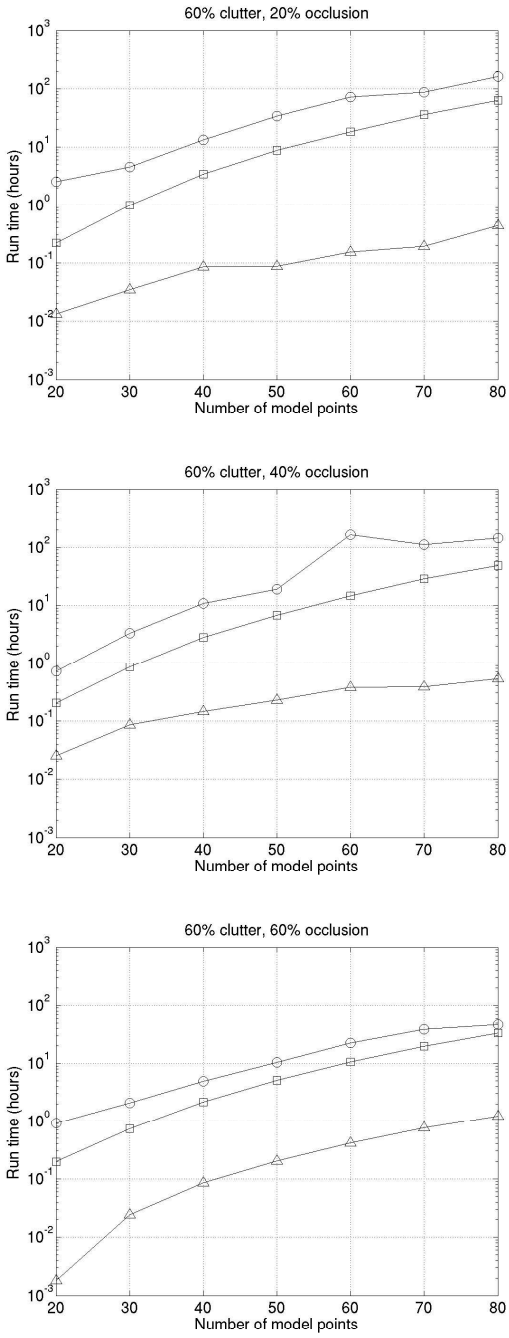


Figure 3.12: Comparison of the run times of SoftPOSIT to those of RANSAC for problems with 60% clutter and 20-60% object occlusion. The SoftPOSIT run times are marked with triangles. The RANSAC run times are marked with circles for the case that the number of samples is determined by s_1, and with squares for the case that the number of samples is determined by s_2.

(a) (b)

Figure 3.13: Pose estimation from a city image. (a) Original image from a virtual reality system. (b) World model (white lines) projected into this image using the pose computed by SoftPOSIT.

an onboard navigation system) to cull the majority of model points that don't project into the estimated field of view. Then, those world points that do fall into this estimated view are further culled by keeping only those that project near the detected skyline. Although this is not real imagery, the virtual reality system used is very sophisticated, and as such, should give a good indication of how the system will perform on real imagery. The figure shows that the correct pose is found, as indicated by the fact that the world model of the skyline projects close to the skyline in the image.

3.5.4.2 Robot Docking Application

The robot docking application requires that a small robot drive onto a docking platform that is mounted on a larger robot. Figure 3.14 shows a small robot docking onto a larger robot. In order to accomplish this, the small robot must determine

Figure 3.14: A small robot docking onto a larger robot.

the relative pose of the large robot. This is done by using SoftPOSIT to align a 3D model of the large robot to corner points extracted from an image of the large robot.

The model of the large robot consists of a set of 3D points that are extracted from a triangular faceted model of the robot which was generated by a commercial CAD system. To detect the corresponding points in the image, lines are first detected using a combination of the Canny edge detector, the Hough transform, and a sorting procedure used to rank the lines produced by the Hough transform. Corners are then found at the intersections of those lines that satisfy simple length, proximity, and angle constraints. Figure 3.15 shows the lines and corner points detected in one image of the large robot. In this test there are 70 points in the object; 89% of these are occluded (or not detected in the image), and 58% of the image points are clutter. Figure 3.16a shows the initial guess generated by SoftPOSIT which led to

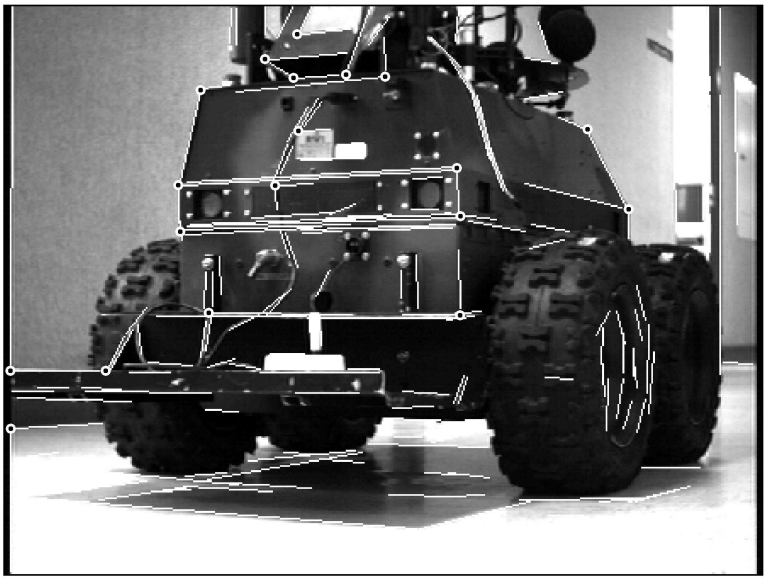
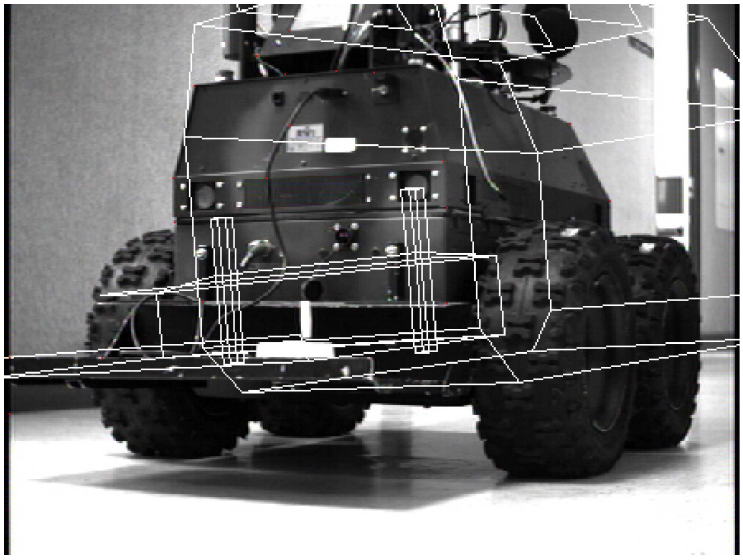
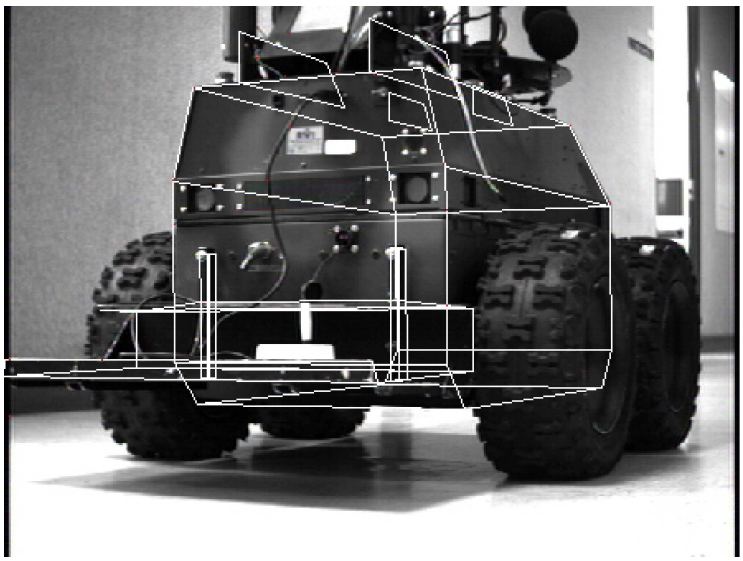


Figure 3.15: An image of the large robot as seen from the small robot's point of view. Long straight lines detected in the image are shown in white, and their intersections, which ideally should correspond to vertices in the 3D object, are shown in black.

the correct pose being found, and Figure 3.16b shows this correct pose.



(a)



(b)

Figure 3.16: Pose estimation from a robot image. The initial guess at the robot's pose (a) that leads to the correct pose as shown in (b). The lines shown in this figure are not used by the point-based SoftPOSIT algorithm; they are shown to aid in interpretation of the results. Only the points at the junctions of these lines are used in the pose calculation.

Chapter 4

SoftPOSIT for Line Endpoints

We extend the SoftPOSIT algorithm from matching point features to the case of matching line features: 3D model lines are matched to image line segments 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. This 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.

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 4.1. 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 $\mathbf{n}_i = (\mathbf{p}_i, 1)^\mathsf{T} \times (\mathbf{p}'_i, 1)^\mathsf{T}$, and 3D points $\mathbf{P}$ in the camera

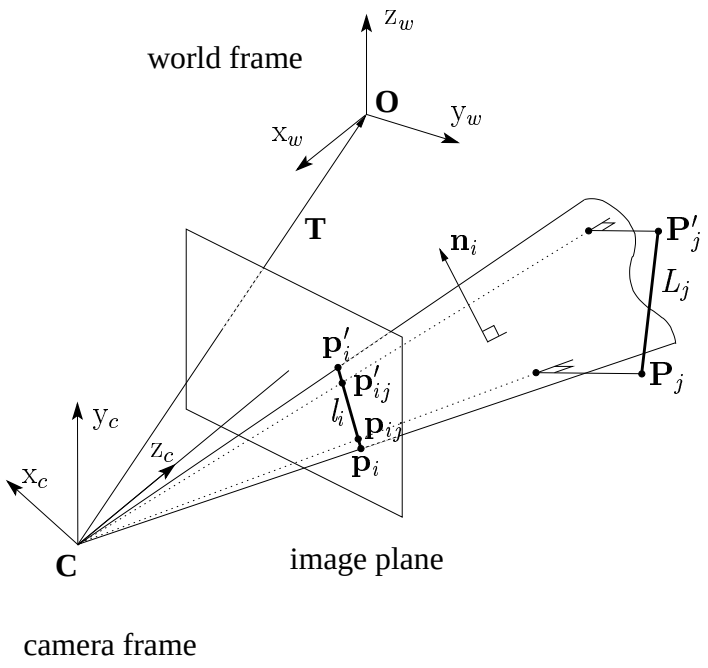


Figure 4.1: The geometry of line correspondences.

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:

$$\begin{aligned} \widetilde{\mathbf{S}}_{ij} &= R\mathbf{P}_j + \mathbf{T} - (R\mathbf{P}_j + \mathbf{T}) \cdot \mathbf{n}_i, \\ \widetilde{\mathbf{S}}'_{ij} &= R\mathbf{P}'_j + \mathbf{T} - (R\mathbf{P}'_j + \mathbf{T}) \cdot \mathbf{n}_i, \end{aligned} \tag{4.1}$$

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 $\widetilde{\mathbf{S}}'_{ij}$:

$$\mathbf{p}_{ij} = \frac{(\tilde{\mathbf{s}}_{ij_x}, \tilde{\mathbf{s}}_{ij_y})}{\tilde{\mathbf{s}}_{ij_z}}, \quad \mathbf{p}'_{ij} = \frac{\left(\tilde{\mathbf{s}}'_{ij_x}, \tilde{\mathbf{s}}'_{ij_y}\right)}{\tilde{\mathbf{s}}'_{ij_z}}. \tag{4.2}$$

4.2 Computing Pose and Correspondences

The pose and correspondence algorithm for *points* (SoftPOSIT as described in Chapter 3) involves iteratively refining estimates of the pose and correspondences for the given 2D and 3D point sets. The current 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 line endpoints are estimated by the point set

$$P_{\text{img}}(R, \mathbf{T}) = \left\{\mathbf{p}_{ij}, \mathbf{p}'_{ij}, 1 \leq i \leq n, 1 \leq j \leq m\right\}, \tag{4.3}$$

which is computed using equations (4.1) and (4.2). For every 3D endpoint of an object line, there are n 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 set of object points used by SoftPOSIT is fixed and is the set of $2m$ object line endpoints: $P_{\text{obj}} = \left\{\mathbf{P}_j, \mathbf{P}'_j, 1 \leq j \leq m\right\}$.

We now have a set of $2mn$ image points and a set of $2m$ 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 4.3. 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 $(2mn+1) \times (2m+1)$, only $4mn$ of its 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 $O(mn^2)$ (see Section 3.5.2).

Algorithm 8 shows the high-level pseudocode for the line-based SoftPOSIT algorithm. This algorithm 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 Section 3.4 for details.

4.3 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.

Algorithm 8 – SOFTPOSIT_LINE_ENDPOINTS: High-level pseudocode for the line-based SoftPOSIT algorithm.

initialize R, $\mathbf{T}$, β, and P_{obj}.

repeat

- 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.
- Initialize the assignment matrix as $M_{ij}^0 = \exp(-\beta(d_{ij}^2 - \alpha))$.
- Normalize M with Sinkhorn's algorithm (**SINKHORN3**).
- Compute $P_{\text{img}}(R, \mathbf{T})$ (Equation (4.3)).
- Solve for $\mathbf{Q}_1$ and $\mathbf{Q}_2$ (Equations (3.16) and (3.17)) using M and the point sets P_{obj} and P_{img}.
- Compute R and $\mathbf{T}$ from $\mathbf{Q}_1$ and $\mathbf{Q}_2$ (Equations (3.5) and (3.6)).
- Set $\beta = \beta_{\text{update}} \cdot \beta$.

until R and $\mathbf{T}$ have converge.

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. As explained below, this metric applies a type of Hough transform to all lines (image and projected model) and then measures the Euclidean 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.

4.4 Experiments with Line Data

4.4.1 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 4.2 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 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 4.2 differs from that of the true pose by 28° about each of the coordinate axis.)

4.4.2 Real Images

Figure 4.3 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

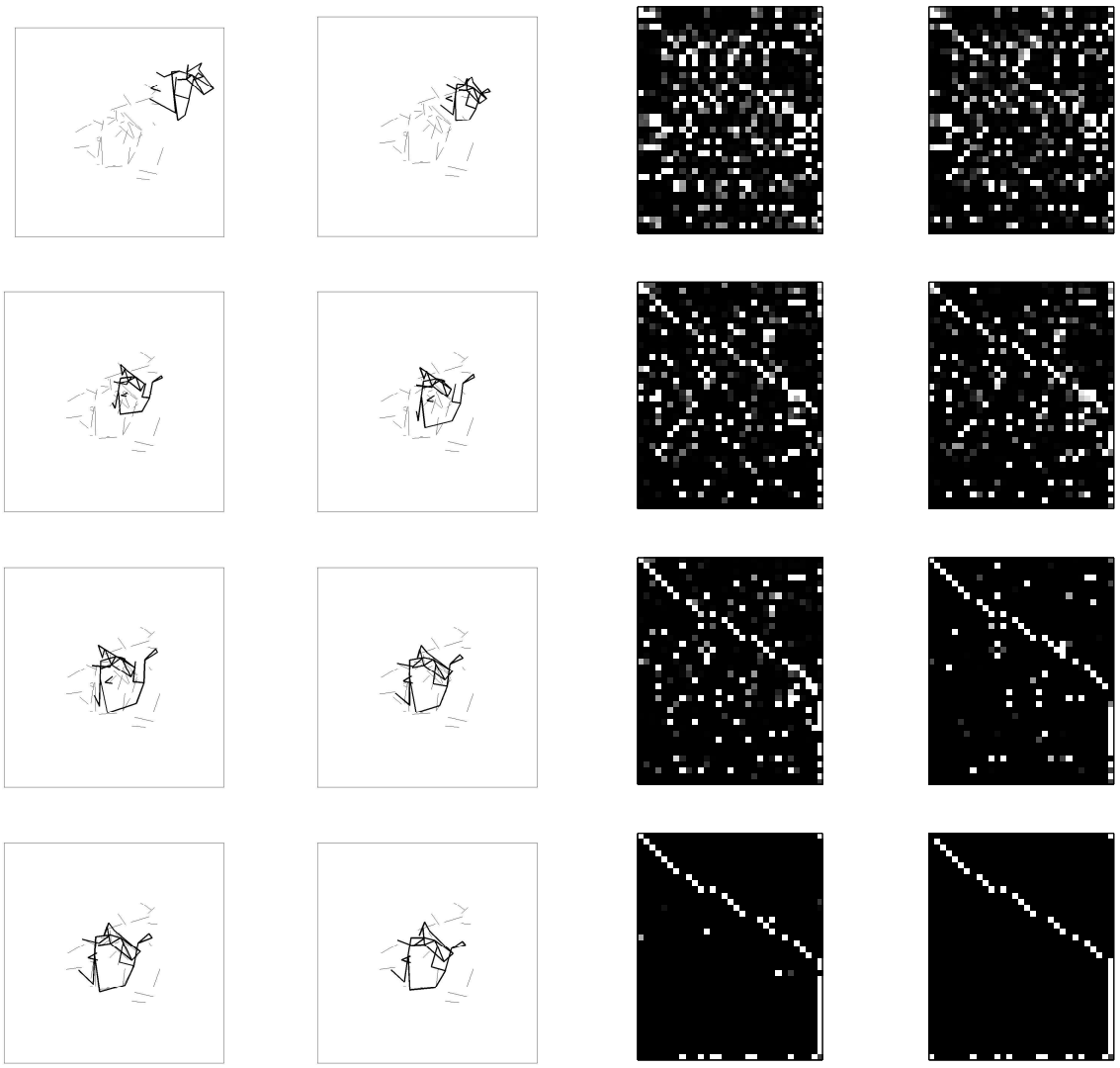


Figure 4.2: Example application of our algorithm to a cluttered image. The eight frames on the left show the estimated pose at initialization (upper left) and at steps 1, 3, 5, 12, 20, 27, and 35 (left-to-right, top-to-bottom) of the iteration. The thin lines are the image lines and the bold lines are 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.

Figure 4.3: Determining the pose of a CAD model from a real image. Straight lines are automatically detected in the image (top left). The initial guess for the pose of the hallway model differed from the true pose by about 14° about each coordinate axis (top right). The projection of the model after finding its pose with our algorithm (bottom).

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.

Chapter 5

SoftPOSIT for Lines

This chapter extends again the SoftPOSIT algorithm from matching point features to matching line features: 3D model lines are matched to image lines in 2D perspective images. Christy et al. [27] have established the basic equations linking perspective and scaled orthographic perspective images of 3D lines. Their equations assume that the correspondences between model lines and image lines is known. In model-to-image registration, such correspondences are not known in advance. We integrate these line constraints with the SoftPOSIT algorithm to produce an efficient algorithm that solves the model-to-image registration problem in difficult high-clutter and high-occlusion problems.

5.1 Camera Models

Let $\mathbf{X}$ be the coordinates of a 3D point in a world coordinate frame. If a camera placed in this world coordinate frame is used to view $\mathbf{X}$, then the coordinates of this point in the camera coordinate frame may be written as $\mathbf{X}_c = R\mathbf{X} + \mathbf{t}$. Here, R is a 3×3 rotation matrix representing the orientation of the camera coordinate frame with respect to the world coordinate frame, and the translation with respect to the rotated world frame is $\mathbf{t} = -R\mathbf{C}$ where $\mathbf{C}$ are the coordinates of the camera center in the world coordinate frame.

We assume that the camera is calibrated, so that pixel coordinates can be replaced by normalized image coordinates. Then, the perspective projection of 3D points in the world coordinate frame onto the 2D image is accomplished by the 3×4 homogeneous camera matrix

$$P = [R \mid \mathbf{t}].$$

The homogeneous image coordinates of a 3D point $\mathbf{X}$ are $\mathbf{x} = P(\mathbf{X}^\mathsf{T}\ 1)^\mathsf{T}$. Let the i^{th} row of R be denoted by $\mathbf{r}_i^\mathsf{T}$ and let the translation be $\mathbf{t} = (t_x, t_y, t_z)^\mathsf{T}$. Since P is homogeneous, we can multiply all entries of P by $1/t_z$. Making the substitutions

$$\begin{gathered} \mathbf{I} = \mathbf{r}_1/t_z,\ \mathbf{J} = \mathbf{r}_2/t_z,\ \mathbf{K} = \mathbf{r}_3/t_z, \\ x_0 = t_x/t_z,\ \text{and}\ y_0 = t_y/t_z, \end{gathered} \tag{5.1}$$

we have

$$P = \begin{bmatrix} \mathbf{I}^\mathsf{T} & x_0 \\ \mathbf{J}^\mathsf{T} & y_0 \\ \mathbf{K}^\mathsf{T} & 1 \end{bmatrix}.$$

Thus, the perspective image of a point $\mathbf{X}$ is

$$\mathbf{x} = \begin{bmatrix} \mathbf{I}^\mathsf{T} & x_0 \\ \mathbf{J}^\mathsf{T} & y_0 \\ \mathbf{K}^\mathsf{T} & 1 \end{bmatrix} \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{I}^\mathsf{T}\mathbf{X} + x_0 \\ \mathbf{J}^\mathsf{T}\mathbf{X} + y_0 \\ \mathbf{K}^\mathsf{T}\mathbf{X} + 1 \end{pmatrix}. \tag{5.2}$$

We will also need to use the scaled orthographic projection model, which makes the assumption that the depth of a scene is small compared to the distance of the

scene from the camera, and that visible scene points are close to the optical axis. The scaled orthographic model will be used iteratively in the process of computing the full perspective pose. Under the scaled orthographic assumption, $\mathbf{K}\cdot\mathbf{X} \approx 0$ since $\mathbf{r}_3$ is a unit vector in the world coordinate frame that is parallel to the camera's optic axis. The scaled orthographic projection camera matrix is therefore

$$P_w = \begin{bmatrix} \mathbf{I}^\mathsf{T} & x_0 \\ \mathbf{J}^\mathsf{T} & y_0 \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}.$$

5.2 Pose from Known Line Correspondences

The set of points lying on an infinite line L_j in 3-space is represented by the equation

$$\mathbf{X}_\lambda = \mathbf{\Omega}_j + \lambda \mathbf{D}_j \tag{5.3}$$

for $\lambda \in (-\infty, \infty)$. Here, $\mathbf{\Omega}_j$ is any fixed point on the line (the reference point), and $\mathbf{D}_j$ is the direction of the line. $\mathbf{\Omega}_j$, $\mathbf{D}_j$, and $\mathbf{X}_\lambda$ are all 3-vectors specified with respect to the world coordinate frame. Figure 5.1 illustrates this situation. A line in the image is represented by a homogeneous point $l_i = (a_i, b_i, c_i)$ and points $(u, v, w)^\mathsf{T}$ lying on this line satisfy the equation

$$a_i u + b_i v + c_i w = 0. \tag{5.4}$$

If image line l_i is the perspective projection of model line L_j, then, by com-

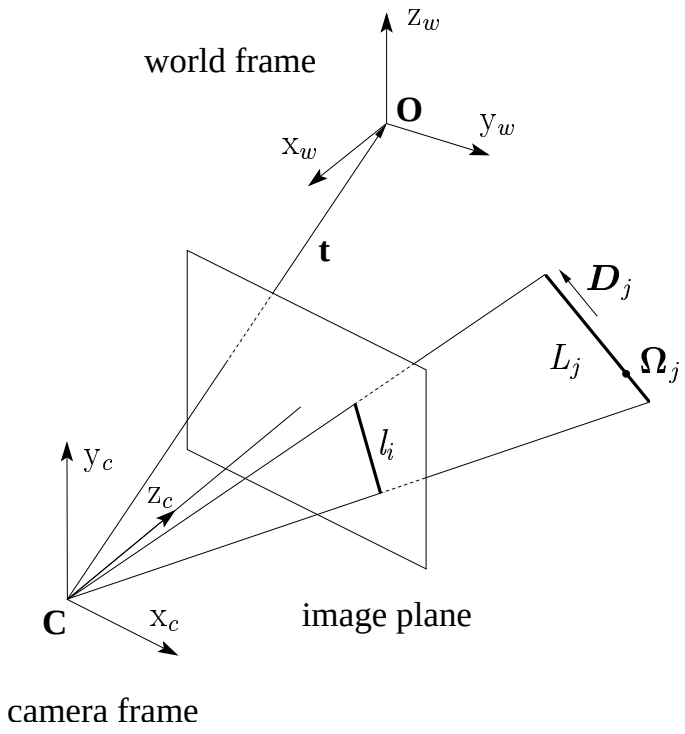


Figure 5.1: The projection of a 3D line into an image. Model line L_j, defined by the point $\mathbf{\Omega}_j$ and direction $\boldsymbol{D}_j$, projects to image line l_i. $\mathbf{O}$ is the world coordinate frame origin, $\mathbf{C}$ the camera center of projection, and $\mathbf{t}$ is the camera translation. The camera optical axis is aligned with z_c.

bining equations (3.1), (5.3), and (5.4), one obtains

$$\begin{aligned} a_i\mathbf{I}\cdot\mathbf{\Omega}_j + b_i\mathbf{J}\cdot\mathbf{\Omega}_j + a_i x_0 + b_i y_0 + c_i(1+\eta_j) &= 0 \\ a_i\mathbf{I}\cdot\mathbf{D}_j + b_i\mathbf{J}\cdot\mathbf{D}_j + c_i\mu_j &= 0 \end{aligned} \tag{5.5}$$

where $\eta_j = \boldsymbol{K}\cdot\mathbf{\Omega}_j$ and $\mu_j = \boldsymbol{K}\cdot\boldsymbol{D}_j$ [27]. It can be seen that $\eta_j + 1$ is the the depth of $\mathbf{\Omega}_j$ from the camera center in the direction of the optical axis, scaled by $1/t_z$, and that μ_j is the length of the projection of the line's direction vector, $\boldsymbol{D}_j$, onto the optical axis, scaled by $1/t_z$. Under the assumptions of scaled orthographic projection, all $\mathbf{\Omega}_j$ and $\mathbf{D}_j$ lie in a plane at depth t_z parallel to the image plane, and

so $\eta_j = 0$ and $\mu_j = 0$ for all j.

Given four or more image-to-model line correspondences, Christy et al. [27] use Equation (5.5) in the iterative POSIT algorithm [38] to solve for object pose ($\mathbf{I}$, $\mathbf{J}$, $\mathbf{K}$, x_0, and y_0). On each iteration of this algorithm, the known correspondences and current estimates for η_j and μ_j allow the formation of a system of linear equations in the unknown pose. The least squares solution of this system provides new estimates for pose, and therefore η_j and μ_j. This procedure is iterated until η_j and μ_j converge to fixed values.

5.3 Pose from Unknown Correspondences

In the simultaneous pose and correspondence problem, we aren't given any line correspondences, so equation (5.5) can't be used to solve for pose. Our approach is to minimize, over all poses and correspondences, the sum of distances between image lines and the projections of the corresponding model lines. Thus, we assume the existence of a match matrix M_{ij} where $0 \leq M_{ij} \leq 1$ and $\Sigma_i M_{ij} = \Sigma_j M_{ij} = 1$. The value of M_{ij} indicates the *degree of belief* that image line l_i corresponds to model line L_j. From equation (5.5), the squared distance between l_i and the projection of L_j into the image, as a function of the object pose, may be defined as

$$d_{ij}^2 = [a_i\mathbf{I}\cdot\mathbf{\Omega}_j + b_i\mathbf{J}\cdot\mathbf{\Omega}_j + a_i x_0 + a_i x_0 + b_i y_0 + c_i(1+\eta_j)]^2 + [a_i\mathbf{I}\cdot\mathbf{D}_j + b_i\mathbf{J}\cdot\mathbf{D}_j + c_i\mu_j]^2 \quad (5.6)$$

Then, the registration problem may be formulated as a minimization of the

objective function

$$E = \sum_{i=1}^{n} \sum_{j=1}^{m} M_{ij} d_{ij}^2 \tag{5.7}$$

where m is the number lines in the model and n is the number of lines in the image.

Equation (5.7) needs to be minimized with respect to the correspondence variables M_{ij} and the pose variables. We do this iteratively, first minimizing the cost function with respect to pose, and then with respect to the correspondence. This process is repeated until convergence

5.3.1 Optimization with respect to Pose

Assume that the correspondence variables M_{ij} are fixed. We wish to find $\mathbf{I}$, $\mathbf{J}$, $\mathbf{K}$, x_0, and y_0 that minimize E. These are found by solving $\partial E/\partial \mathbf{I} = \partial E/\partial \mathbf{J} = \partial E/\partial x_0 = \partial E/\partial y_0 = 0$. Let $\mathrm{I_k}$, J_k, Ω_{jk}, and D_{jk} denote the k-th element ($1 \leq k \leq 3$) of the respective 3-vector. Then it's easy to show that

$$\partial E/\partial \mathrm{I}_k = 2 \sum_{i,j} M_{ij} \left[\begin{pmatrix} a_i^2 \mathbf{V}_{jk} \\ a_i b_i \mathbf{V}_{jk} \\ a_i^2 \Omega_{jk} \\ a_i b_i \Omega_{jk} \end{pmatrix} \cdot \mathbf{W} + a_i c_i (\Omega_{jk}(1+\eta_j) + \mathrm{D}_{jk}\mu_j) \right]$$

where

$$\mathbf{V}_{jk} = \Omega_{jk}\mathbf{\Omega}_j + \mathrm{D}_{jk}\mathbf{D}_j \quad \text{and} \quad \mathbf{W} = \begin{pmatrix} \mathbf{I} \\ \mathbf{J} \\ x_0 \\ y_0 \end{pmatrix}. \tag{5.8}$$

Setting $\partial E/\partial \mathrm{I}_k = 0$, we get

$$\begin{pmatrix} \sum_{i,j} M_{ij} a_i^2 \mathbf{V}_{jk} \\ \sum_{i,j} M_{ij} a_i b_i \mathbf{V}_{jk} \\ \sum_{i,j} M_{ij} a_i^2 \Omega_{jk} \\ \sum_{i,j} M_{ij} a_i b_i \Omega_{jk} \end{pmatrix} \cdot \mathbf{W} = -\sum_{i,j} M_{ij} a_i c_i (\Omega_{jk}(1+\eta_j) + \mathrm{D}_{jk}\mu_j). \tag{5.9}$$

Equation (5.9) gives three linear equations ($k = 1, 2, 3$) in the unknown pose. Similarly, setting $\partial E/\partial \mathrm{J}_k = 0$, we obtain the three equations

$$\begin{pmatrix} \sum_{i,j} M_{ij} a_i b_i \mathbf{V}_{jk} \\ \sum_{i,j} M_{ij} b_i^2 \mathbf{V}_{jk} \\ \sum_{i,j} M_{ij} a_i b_i \Omega_{jk} \\ \sum_{i,j} M_{ij} b_i^2 \Omega_{jk} \end{pmatrix} \cdot \mathbf{W} = -\sum_{i,j} M_{ij} b_i c_i (\Omega_{jk}(1+\eta_j) + \mathrm{D}_{jk}\mu_j). \tag{5.10}$$

Setting $\partial E/\partial x_0 = 0$ we have

$$\begin{pmatrix} \sum_{i,j} M_{ij} a_i^2 \mathbf{\Omega}_j \\ \sum_{i,j} M_{ij} a_i b_i \mathbf{\Omega}_j \\ \sum_{i,j} M_{ij} a_i^2 \\ \sum_{i,j} M_{ij} a_i b_i \end{pmatrix} \cdot \mathbf{W} = -\sum_{i,j} M_{ij} a_i c_i (1 + \eta_j). \tag{5.11}$$

And, from $\partial E/\partial y_0 = 0$, we similarly have

$$\begin{pmatrix} \sum_{i,j} M_{ij} a_i b_i \mathbf{\Omega}_j \\ \sum_{i,j} M_{ij} b_i^2 \mathbf{\Omega}_j \\ \sum_{i,j} M_{ij} a_i b_i \\ \sum_{i,j} M_{ij} b_i^2 \end{pmatrix} \cdot \mathbf{W} = -\sum_{i,j} M_{ij} b_i c_i (1 + \eta_j). \tag{5.12}$$

Equations (5.9), (5.10), (5.11), and (5.12) give 8 linear equations in the 8 unknowns of $\mathbf{W}$. These can be written as

$$A\mathbf{W} = \mathbf{B} \tag{5.13}$$

where A is the 8×8 matrix

$$A = \begin{pmatrix} A_1 & A_2 & A_3 \\ A_2^{\mathsf{T}} & A_4 & A_5 \\ A_3^{\mathsf{T}} & A_5^{\mathsf{T}} & A_6 \end{pmatrix}$$

with

$$
\begin{aligned}
A_1 &= \Sigma_j(\Sigma_i M_{ij} a_i^2)(\mathbf{\Omega}_j\mathbf{\Omega}_j^\mathsf{T} + \mathbf{D}_j\mathbf{D}_j^\mathsf{T}),\\
A_2 &= \Sigma_j(\Sigma_i M_{ij} a_i b_i)(\mathbf{\Omega}_j\mathbf{\Omega}_j^\mathsf{T} + \mathbf{D}_j\mathbf{D}_j^\mathsf{T}),\\
A_3 &= \Sigma_j \mathbf{\Omega}_j \begin{pmatrix} \Sigma_i M_{ij} a_i^2 & \Sigma_i M_{ij} a_i b_i \end{pmatrix},\\
A_4 &= \Sigma_j(\Sigma_i M_{ij} b_i^2)(\mathbf{\Omega}_j\mathbf{\Omega}_j^\mathsf{T} + \mathbf{D}_j\mathbf{D}_j^\mathsf{T}),\\
A_5 &= \Sigma_j \mathbf{\Omega}_j \begin{pmatrix} \Sigma_i M_{ij} a_i b_i & \Sigma_i M_{ij} b_i^2 \end{pmatrix},\\
A_6 &= \Sigma_j \begin{pmatrix} \Sigma_i M_{ij} a_i^2 & \Sigma_i M_{ij} a_i b_i \\ \Sigma_i M_{ij} a_i b_i & \Sigma_i M_{ij} b_i^2 \end{pmatrix}.
\end{aligned}
$$

and $\mathbf{B}$ is the 8-vector

$$
\mathbf{B} = -\Sigma_j \begin{pmatrix} (\Sigma_i M_{ij} a_i c_i)((1+\eta_j)\mathbf{\Omega}_j + \mu_j \mathbf{D}_j) \\ (\Sigma_i M_{ij} b_i c_i)((1+\eta_j)\mathbf{\Omega}_j + \mu_j \mathbf{D}_j) \\ (1+\eta_j) \begin{pmatrix} \Sigma_i M_{ij} a_i c_i \\ \Sigma_i M_{ij} b_i c_i \end{pmatrix} \end{pmatrix}.
$$

The objective function (eq. 5.7) is then minimized by

$$
\mathbf{W} = A^{-1}\mathbf{B}. \tag{5.14}
$$

Having $\mathbf{W}$, the pose of the model is computed as

$$
t_z = 1/\sqrt{\|\mathbf{I}\|\,\|\mathbf{J}\|},\ t_x = t_z x_0,\ t_y = t_z y_0,
$$

$$
\mathbf{r}_1 = t_z\mathbf{I},\ \mathbf{r}_2 = t_z\mathbf{J},\ \mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2.
$$

5.3.2 Optimization with respect to Correspondence

We now optimize the correspondence variables M_{ij} assuming the pose of the object is known and fixed. Our aim is to find a zero-one *assignment matrix*, $M = \{M_{ij}\}$, that explicitly specifies the matches between a set of m model lines and a set of n image lines, and that minimizes the objective function E. M has one row for each of the n image lines l_i and one column for each of the m model lines L_j. The assignment matrix must satisfy the constraint that each image line match at most one model line, and vice versa (i.e., $\Sigma_k m_{ik} = \Sigma_k m_{kj} = 1$ for all i and j). A *slack row* $m+1$ and a *slack column* $n+1$ are added to handle model occlusion and image clutter. A value of 1 in the slack column $m+1$ at row i indicates that image line l_i has not found any match among the model lines. A value of 1 in the slack row $n+1$ at column j indicates that the model line L_j is not seen in the image and does not match any image line. The objective function E will be minimum if the assignment matrix M matches image and model lines with the smallest distances d_{ij}^2. This problem can be solved by the iterative softassign technique [55, 56]. The iteration for the assignment matrix M 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 lines must be before considering them unmatchable. See [56] for an analytical justification. The continuous matrix M_0 converges toward the discrete matrix M due to the combination of deterministic annealing with Sinkhorn's technique, as described in Section 3.3.2. The matrix M resulting from an iteration

Algorithm 9 – SOFTPOSIT_LINES: High-level pseudocode for the second line-based SoftPOSIT algorithm.

initialize $\mathbf{I}$, $\mathbf{J}$, x_0, and y_0 using Equation (5.1) and an initial estimate of R and $\mathbf{T}$.

initialize $\beta = \beta_0$.

repeat

- Project the model lines into the image using the current pose estimate.
- Using Equation (5.6) (or one of the distance measures described in Section 5.3.4), compute the distances d_{ij} between the true image lines and the projected model lines.
- Initialize the assignment matrix as $M_{ij}^0 = \exp(-\beta(d_{ij}^2 - \alpha))$.
- Compute M by normalizing M^0 with Sinkhorn's algorithm (**SINKHORN3**).
- Compute $\mathbf{I}$, $\mathbf{J}$, x_0, and y_0 using Equations (5.8) and (5.14).
- Set $\beta = \beta_{\text{update}} \cdot \beta$.

until $\mathbf{I}$, $\mathbf{J}$, x_0, and y_0 have converge.

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.

5.3.3 Computing Pose and Correspondences

High-level pseudocode for the second line-based SoftPOSIT algorithm is shown in Algorithm 9.

5.3.4 Alternate Distance Measures

The distance measure defined in equation (5.6) has been used with good results. However, we have observed that the regions of convergence to the true pose are usually significantly larger when alternate distance measures are employed in the correspondence optimization phase of the algorithm. Using different distance measures in the two parts of the algorithm is not a problem for the overall approach as long as they are monotonically related: a reduction in the value of the objective function for one distance function is guaranteed to produce a reduction when the other distance function is used. 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 are described below.

The value of d_{ij} defined in equation (5.6) is the sum of two values: (1) the squared orthogonal distance of the point 3D point $\mathbf{\Omega}_j$ to the plane defined by the image line l_i and camera center, and (2) the squared orthogonal projection of the line L_j's direction vector, $\mathbf{D}_j$, onto the unit vector normal to the plane defined by the image line l_i and the camera center. This distance measure implicitly assumes that image lines have infinite length, but that model lines have finite length. To be consistent over all lines in the model, $\mathbf{\Omega}_j$ should be chosen as one of the endpoints of L_j and $\mathbf{D}_j$ should have length equal to the length of L_j.

The second distance measure that we tried measures distances in the image plane, but implicitly assumes that both image and projected model lines have infinite

length. As explained below, 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 model 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 lines we sum the distances computed using five different reference points, one at each corner of the image and one at the image center: $d_{ij} = \sum_{k=1}^{5} \|h_k(l_i) - h_k(l_j)\|$. This distance measure produced better performance than the first by allowing our algorithm to converge to the true pose from initial guesses that were further from the true pose.

The third 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 model 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 model 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 either of the previous two, resulting in larger regions of convergence and fewer number of iterations to converge.

5.4 Experiments

This algorithm has been evaluated using 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. **Gaps:** Insert one or more random size gaps at random locations in randomly selected lines. Each of these lines is replaced by two or more smaller lines.
6. **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 5.2 shows our algorithm determining the pose of a random 3D model with 40 lines, from a simulated image with 50% occlusion of the model, 60% of the image lines being clutter, and normally distributed noise with standard deviation 2.5% of the image dimension (about 1.5 pixels for a 640×480 image). The initial rotation of the model shown in figure 5.2 differs from that of the true pose by $20 - 30°$

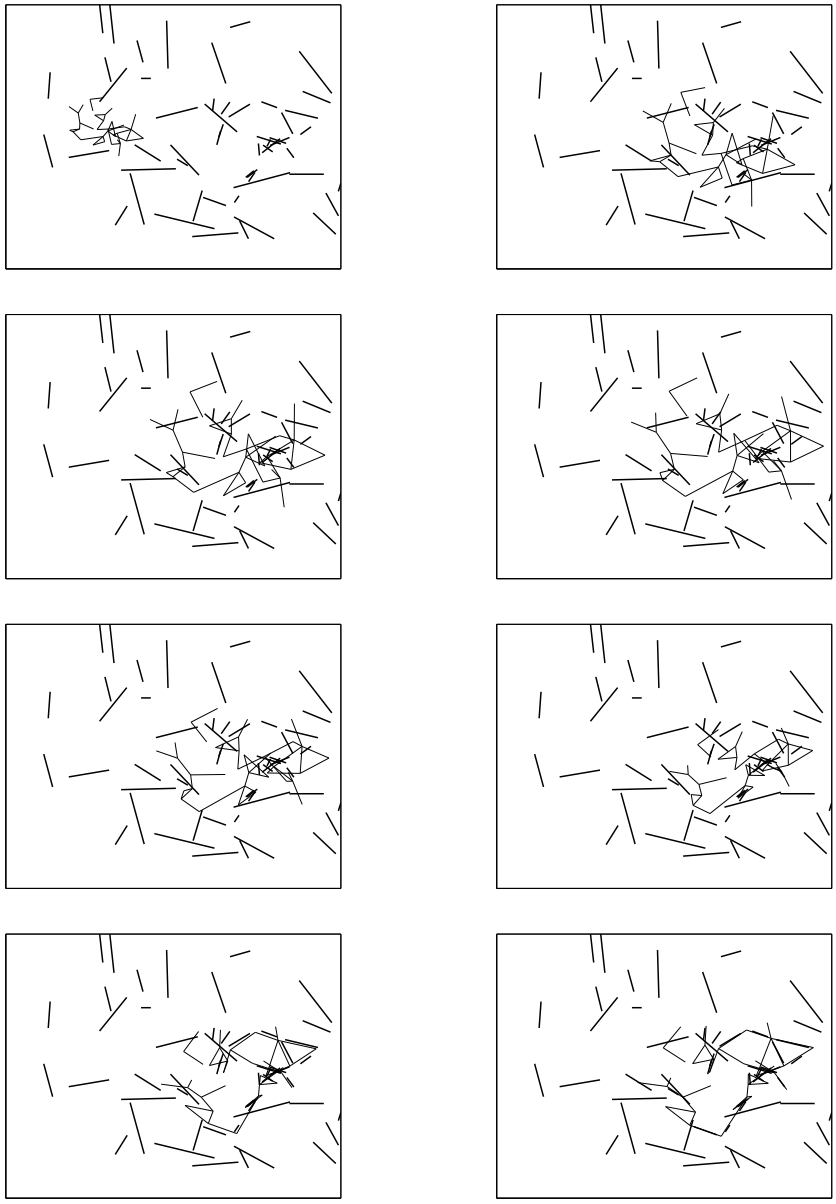


Figure 5.2: Example application of our algorithm to a cluttered image. The eight frames show the progress of the algorithm at every 5^{th} step of the iteration. The projection of the model in it's initial pose is shown at the upper left, and steps 5, 10, 15, 20, 25, 30, and 35 of the iteration follow in a left to right, top to bottom order. The bold lines are the image lines and the thin lines are the projection of the model at the current step of the iteration. The correct pose has been found by iteration step 35 shown at the lower right.

about each of the coordinate axis. As can be seen from this figure, it would be very 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 from such initial guesses.

5.5 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 routinely 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.

Chapter 6

Recognition Using Local Line Neighborhoods

This chapter presents an object recognition algorithm that uses model and image line features to locate complex objects in high clutter environments. In this 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, the graduated assignment algorithm is applied for refinement and verification, starting from the few best approximate poses produced by the previous stages.

6.1 Overview

This chapter 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 perspec-

tive 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.

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 [49], 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 (see Appendix A). 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 [55, 56], 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 Algorithm 10. 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 6.3 briefly discusses the line detection algorithm that we use and presents an evaluation of its ability to extract unfragmented lines from an image. Section 6.4 then shows how approximate pose hypotheses are generated from a minimal number of line correspondences. Next, in Sections 6.5 and 6.6,

Algorithm 10 – **RANKED_POSE_HYPS**: Outline of the 2D object recognition algorithm. The constant N is the number of pose refinements performed; as discussed in Section 6.8, good performance is obtained with $N = 4$.

inputs: A set of image line segments,

A set of models, each consisting of a set of 2D line segments.

outputs: 2D poses and model-to-image assignment matrices of recognized objects.

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, identify the line neighborhood of each line segment in that model.

for each model **do**

$\mathcal{H} = \emptyset$. *(Initialize hypotheses list to empty)*

for each pair of model line l and image line l' **do**

$\mathcal{C}$ = Pose hypotheses generated from l, l', and nearby corners.

$\mathcal{H} = \mathcal{H} \cup \mathcal{C}$.

Evaluate the similarity of model and image neighborhoods for poses $\mathcal{C}$.

end for

$\mathcal{P}$ = Sort $\mathcal{H}$ based on neighborhood similarity measure.

for $i = 1$ to N **do**

Apply the graduated assignment algorithm starting from pose $\mathcal{P}(i)$.

if a sufficient number of line correspondences are found **then**

An object has been recognized; record the model, it's pose, and it's assignment matrix.

end if

end for

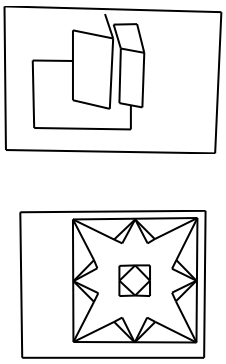
end for

we present our method for efficiently comparing local neighborhoods of model lines to local neighborhoods of image lines. Section 6.7 describes the pose refinement and verification algorithm that we use. Experiments with real imagery containing high levels of clutter and occlusion (see Figure 6.1, for example) are discussed in Section 6.8 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 chapter ends with conclusions in Section 6.9.

6.2 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.

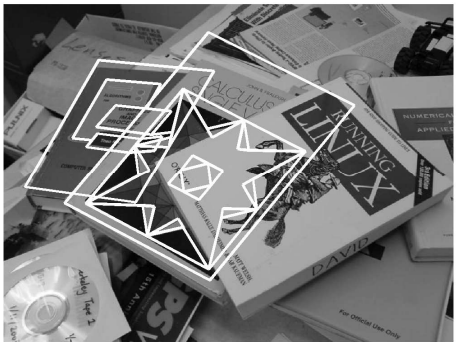
A wide variety of approaches to object recognition have been proposed since Robert's ground-breaking work on recognizing 3D polyhedral objects from 2D perspective images [126]. Among the pioneering contributions are Fischler and Bolles'



(a) Models.

(b) Test image.

(c) 519 detected lines.



(d) Recognized books.

Figure 6.1: Recognizing books in a pile. The two models were generated from frontal images of the books.

RANSAC method [49], Baird's tree-pruning method [5], and Ullman's alignment method [152]. 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 [69] 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 [130] have developed a rotationally invariant feature descriptor using the Harris corner detector. Lowe [97] 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 building facades, 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 preferred 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 [4], which we described on page 55. 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 contrast to the HYPER system, our pose hypotheses are based on affine transformations instead of similarity transformations, we use a dissimilarity measure (see Section 6.6) 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, instead of a tree search, we use the more robust and efficient graduated assignment algorithm [55] for pose refinement.

A number of more recent works [103, 22] have also used edges for object recognition of poorly textured objects. Mikolajczyk et al. [103] 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 [22] 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 [56] 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 (see Chapters 3-5, [28, 29, 30, 31, 32, 39]) for matching 3D model points and lines to 2D image points and lines, respectively, and we use it here for the pose refinement stage of our algorithm. Beveridge [12] matches points and lines using a random start local search algorithm. Denton and Beveridge [41] 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 only 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).

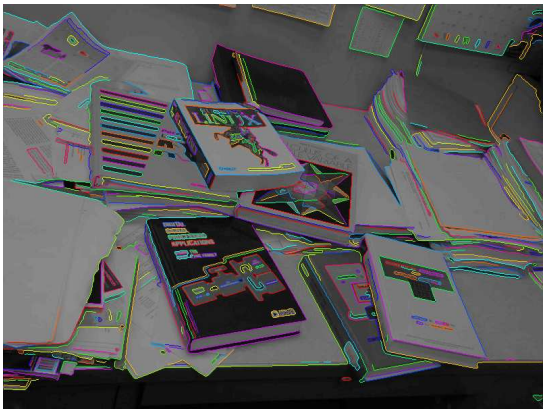
6.3 Line Detection

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 automatically locate model and image lines using Kovesi's implementation [87] of Lowe's line detection algorithm [95]. Briefly, this algorithm 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. These steps are illustrated in Figure 6.2.

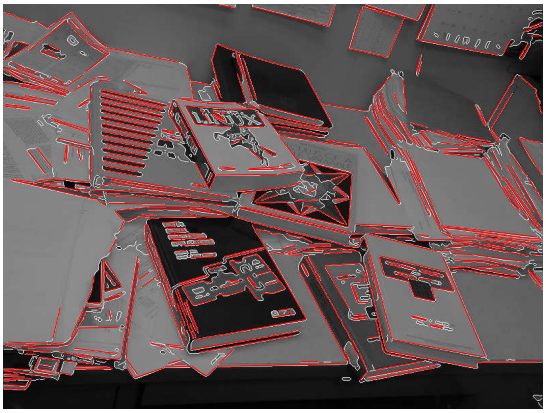
In our experience, for images with dense edges, this approach to line detection performs better than the Hough Transform approach [44]. The high-connectivity of the edges produced by the Canny edge detector greatly simplifies the process of fitting lines to those contours when the contour partitioning 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

(a)

(b)



(c)



(d)

Figure 6.2: Steps of the line detection algorithm. (a) Original image. (b) Binary edges detected with the Canny edge detector. (c) Edge contours. All edge points on a single contour are shown in the same color. Very short contours have been discarded. (d) Line segments produced from partitioning the edge contours.

points.

A requirement of our approach, as stated in Section 6.1, 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 chapter. Using six different images of books and office objects (as typified by images shown throughout this chapter), 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 6.3 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

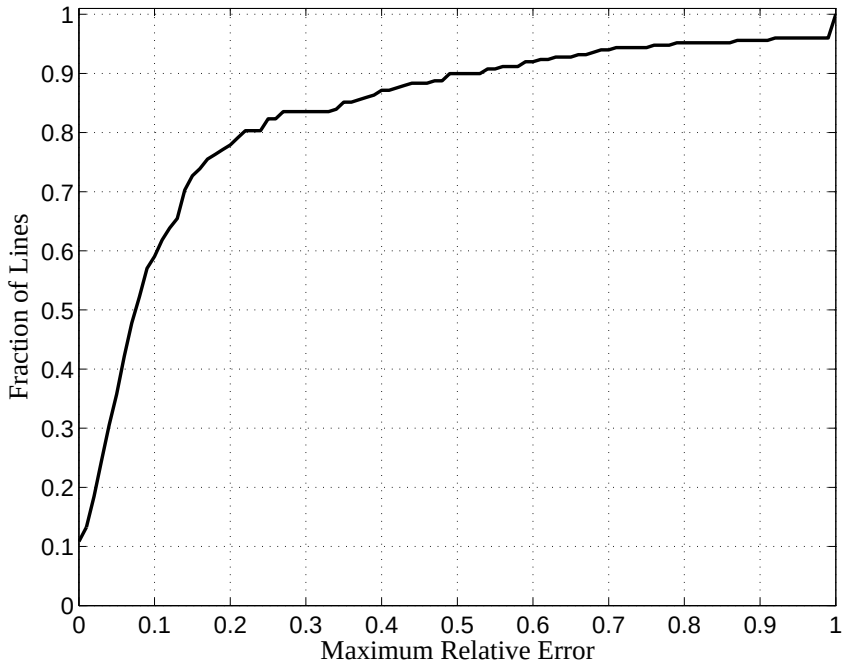


Figure 6.3: The accuracy of our line detector is depicted in this graph, which plots the relative error versus the fraction of the model lines detected with relative error no greater than that amount.

recognize an object.

6.4 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 [101]. 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.

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 6.4. 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

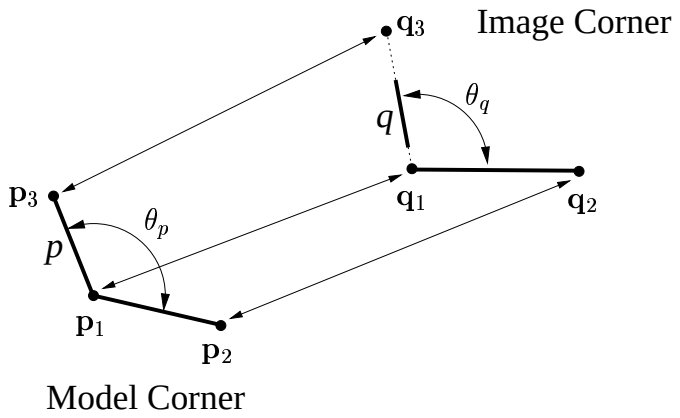


Figure 6.4: Geometry for calculation of the approximate affine transformation.

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 6.4. $\mathbf{p}_3$ 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 $\mathbf{q}_i = \begin{bmatrix} q_{i_x} & q_{i_y} \end{bmatrix}^T$ is

$$\begin{bmatrix} q_{i_x} \\ q_{i_y} \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} p_{i_x} \\ p_{i_y} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}. \tag{6.1}$$

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 6.5 shows the pose hypotheses generated for a particular correct base correspondence.

6.5 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

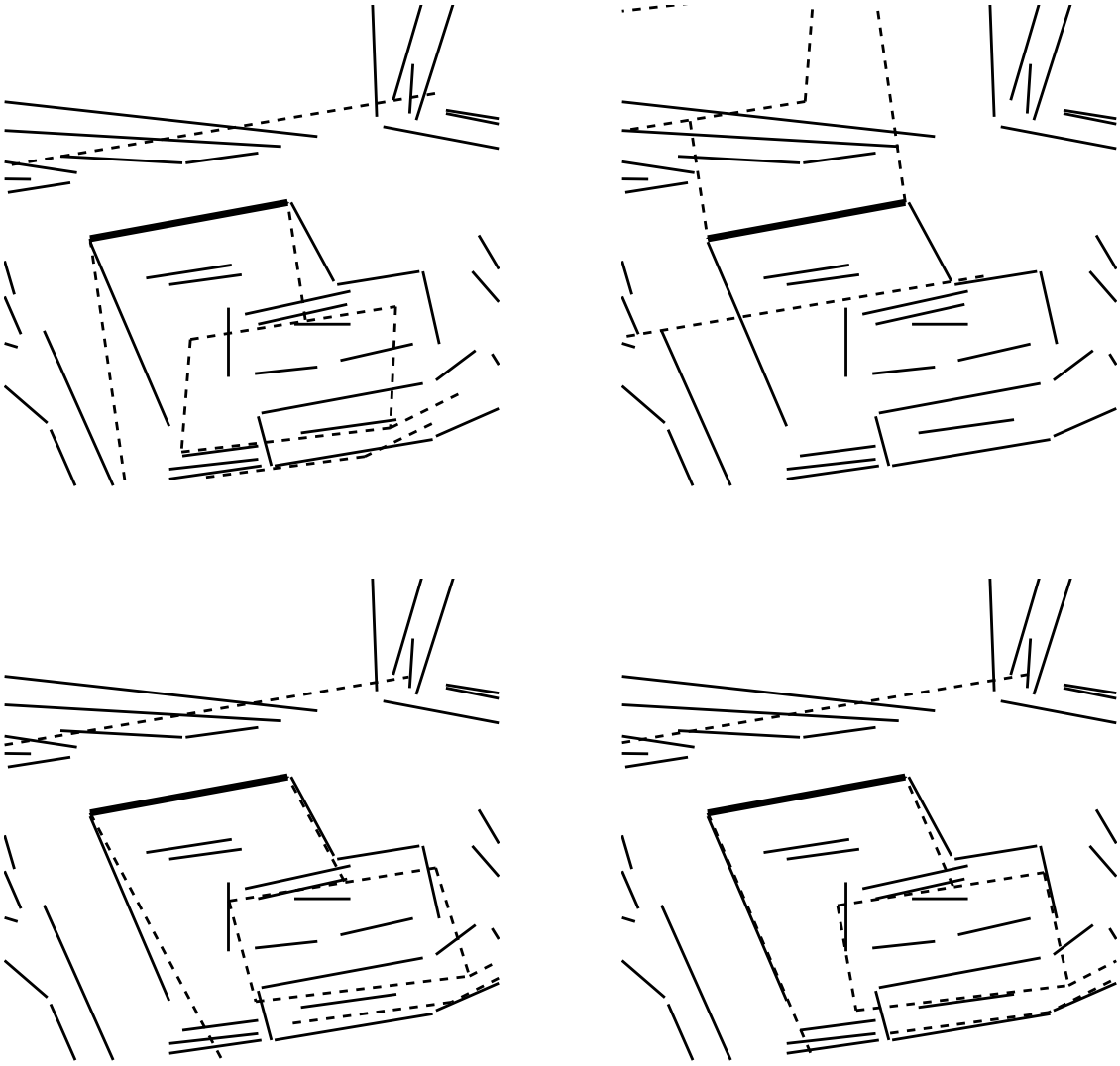


Figure 6.5: Pose hypotheses generated for a correct correspondence of a real model and image line are shown. The model lines (dashed lines and thick solid line) are shown overlaid on the image lines (thin lines). The one thick solid line in each image shows the base correspondence: a model line perfectly 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.

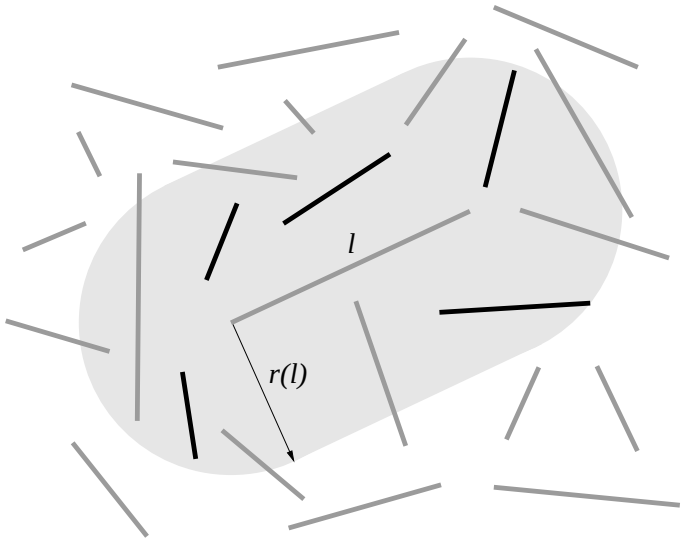


Figure 6.6: The neighborhood radius of line l, in the center of the image, is the minimum distance $r(l)$ for which both endpoints of N_{nbr} lines are within distance $r(l)$ of l. Here, $N_{\text{nbr}} = 5$, but in actual experiments, we take $N_{\text{nbr}} = 10$. The five dark lines are the neighbors of line l.

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 6.6 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 6.6) 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 set of image lines $\mathcal{I}$ is

$$S(\mathcal{N}, \mathcal{I}, A, \mathbf{t}) = \sum_{l' \in \mathcal{T}(\mathcal{N}, A, \mathbf{t})} \min \left\{ S_{\max}, \min_{l'' \in \mathcal{I}} d(l', l'') \right\}. \tag{6.2}$$

The smaller the value of $S(\mathcal{N}, \mathcal{I}, A, \mathbf{t})$, 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 $S(\mathcal{N}, \mathcal{I}, A, \mathbf{t})$ 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 6.6, the distance between a single model neighbor and the closest image line can be found in time $\mathcal{O}(\log n)$ when there are n image lines. Since $|\mathcal{N}| = N_{\text{nbr}}$, the time to compute $S(\mathcal{N}, \mathcal{I}, A, \mathbf{t})$ is $\mathcal{O}(\log n)$.

6.6 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 6.2 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

$$\left[\frac{\theta}{r_\theta}, \frac{x_{\text{mid}}}{r_m}, \frac{y_{\text{mid}}}{r_m}\right] \quad \text{and} \quad \left[\frac{\theta - \pi}{r_\theta}, \frac{x_{\text{mid}}}{r_m}, \frac{y_{\text{mid}}}{r_m}\right] \tag{6.3}$$

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 constant 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

$$\left\{ \left[\frac{\theta}{r_\theta}, \frac{\bar{x}_i}{r_m}, \frac{\bar{y}_i}{r_m} \right], \left[\frac{\theta - \pi}{r_\theta}, \frac{\bar{x}_i}{r_m}, \frac{\bar{y}_i}{r_m} \right], i = 1, 2, \ldots, N_{\text{pts}} \right\} \tag{6.4}$$

where

$$N_{\text{pts}} = \left\lceil \frac{\| [x_2 - x_1, y_2 - y_1] \|}{w} \right\rceil + 1, \tag{6.5}$$

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

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

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 6.5) along each transformed model line is sufficient to

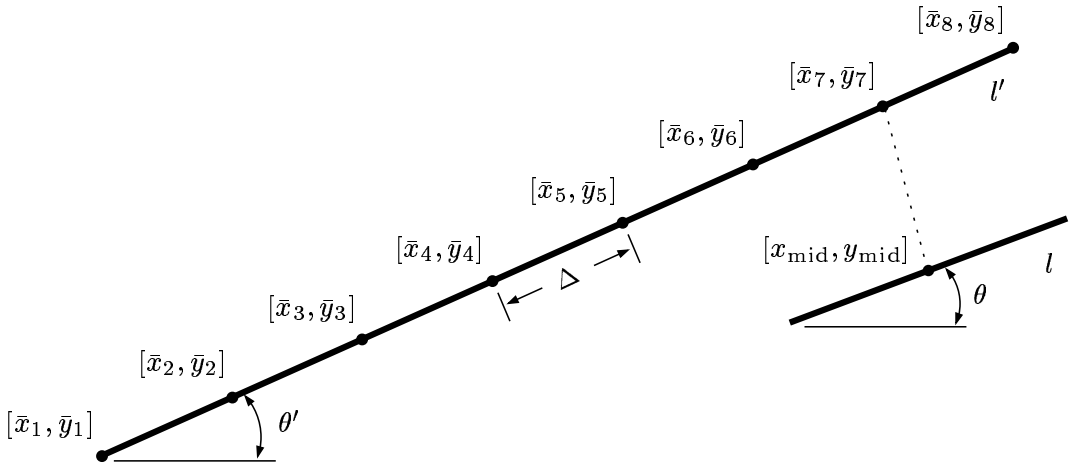


Figure 6.7: Sampling the position of model and image lines. Image line l is represented using its midpoint $[x_{\text{mid}}, y_{\text{mid}}]$ and its orientation θ. Projected model line l' is represented using the points $\{[\bar{x}_i, \bar{y}_i], i = 1, \ldots, N_{\text{pts}}\}$ and its orientation 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 $[\bar{x}_7, \bar{y}_7]$.

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 6.7 illustrates how model and image lines are sampled in the process of generating the points in Equations 6.3 and 6.4.

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 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 6.3 and 6.4. 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

$$r_m = \frac{\left(\sum_{l \in \mathcal{I}} \|l\|\right) \left(\sum_{l \in \mathcal{M}} r\left(l\right)\right)}{|\mathcal{I}| \sum_{l \in \mathcal{M}} \|l\|}. \tag{6.6}$$

The terms in Equation 6.6 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 $d\left(l', l''\right)$ we simply query the nearest neighbor data structure (generated using points from Equation 6.3) with all of the points listed in Equation 6.4 and then use the distance of the closest one. Because the complexity of the nearest neighbor search is $\mathcal{O}\left(\log n\right)$, the use of multiple points to represent lines does not significantly slow down the algorithm.

6.7 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. Our refinement and verification algorithm, which is based on Gold Rangarajan's graduated assignment algorithm [55, 56], 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 6.6 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.

In order to make this chapter self-contained, the following three paragraphs summarize material presented in Section 2.5.2.2 about the graduated assignment

algorithm.

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

$$E = \sum_{j=1}^{m} \sum_{k=1}^{n} M_{jk} \left(d\left(T\left(l_j\right), l'_k\right)^2 - \delta^2 \right). \tag{6.7}$$

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 6.7 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 6.7 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 ([55, 56]), 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

$$M_{jk}^0 = \begin{cases} 1 & \text{if } j = m+1 \text{ or } k = n+1 \\ \exp\left(-\beta\left(d\left(T\left(l_j\right), l'_k\right)^2 - \delta^2\right)\right) & \text{otherwise,} \end{cases} \tag{6.8}$$

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:

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

$$M_{jk}^0 = 1 \text{ when } d\left(T\left(l_j\right), l'_k\right)^2 = \delta^2,$$

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

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 [135] 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 con-

verges 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. Section 3.3.4 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 $T(l_j)$, 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 [68], 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 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

$$E'' = \sum_{j=1}^{m} \sum_{k=1}^{n} M_{jk} \sum_{i=1}^{2} \left(\left(\mathbf{n}_j^{\mathsf{T}} \left(A' p'_{ki} + \mathbf{t}' - p_{j1} \right) \right)^2 - \delta^2 \right) \tag{6.9}$$

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 transformation

$$T' = \{A', \mathbf{t}'\} = \left\{ \begin{bmatrix} a' & b' \\ c' & d' \end{bmatrix}, \begin{bmatrix} t'_x \\ t'_y \end{bmatrix} \right\} \tag{6.10}$$

that minimizes Equation 6.9 can be found by solving the system of six equations

$$\begin{aligned} &\partial E''/\partial a' = 0, \quad \partial E''/\partial b' = 0, \quad \partial E''/\partial c' = 0, \\ &\partial E''/\partial d' = 0, \quad \partial E''/\partial t'_x = 0, \quad \partial E''/\partial t'_y = 0 \end{aligned} \tag{6.11}$$

for a', b', c', d', t'_x, and t'_y. Expanding Equation 6.11, 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

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

$$\begin{pmatrix} x_{n_j}y_{n_j}\left(x'_{k1}y'_{k1}+x'_{k2}y'_{k2}\right) & x_{n_j}^2\left(x'_{k1}+x'_{k2}\right) & x_{n_j}y_{n_j}\left(x'_{k1}+x'_{k2}\right) \\ x_{n_j}y_{n_j}\left(y_{k1}'^2+y_{k2}'^2\right) & x_{n_j}^2\left(y'_{k1}+y'_{k2}\right) & x_{n_j}y_{n_j}\left(y'_{k1}+y'_{k2}\right) \\ y_{n_j}^2\left(x'_{k1}y'_{k1}+x'_{k2}y'_{k2}\right) & x_{n_j}y_{n_j}\left(x'_{k1}+x'_{k2}\right) & y_{n_j}^2\left(x'_{k1}+x'_{k2}\right) \\ y_{n_j}^2\left(y_{k1}'^2+y_{k2}'^2\right) & x_{n_j}y_{n_j}\left(y'_{k1}+y'_{k2}\right) & y_{n_j}^2\left(y'_{k1}+y'_{k2}\right) \\ x_{n_j}y_{n_j}\left(y'_{k1}+y'_{k2}\right) & 2x_{n_j}'^2 & 2x'_{n_j}y'_{n_j} \\ y_{n_j}^2\left(y'_{k1}+y'_{k2}\right) & 2x'_{n_j}y'_{n_j} & 2y_{n_j}'^2 \end{pmatrix}, \tag{6.12}$$

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

$$\mathbf{b} = \sum_{j=1}^{m} \sum_{k=1}^{n} M_{jk} \left(x_{n_j} x_{j1} + y_{n_j} y_{j1} \right) \begin{pmatrix} x_{n_j} \left(x'_{k1} + x'_{k2} \right) \\ x_{n_j} \left(y'_{k1} + y'_{k2} \right) \\ y_{n_j} \left(x'_{k1} + x'_{k2} \right) \\ y_{n_j} \left(y'_{k1} + y'_{k2} \right) \\ 2x_{n_j} \\ 2y_{n_j} \end{pmatrix} . \tag{6.13}$$

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

Figure 6.8 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 Algorithm 11. Figure 6.9 shows an example of how graduated assignment transforms an initial approximate pose into a more accurate pose.

6.8 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 6.10 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

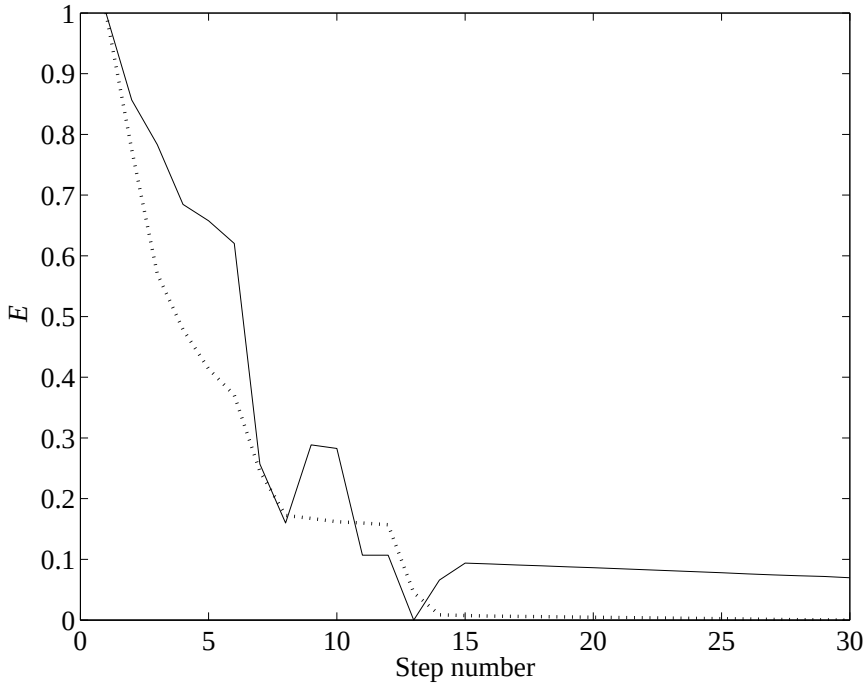


Figure 6.8: Comparison of the two objective functions for a typical minimization by the graduated assignment algorithm. The solid line is E, which uses the Euclidean distances to the nearest neighbor, and the dotted line is E', which uses the sum of the distances of the image line endpoints to the infinitely 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}.

Figure 6.9: Pose refinement using the graduated assignment algorithm: initial pose (left) and final pose after applying the algorithm (right).

Algorithm 11 – **GAA_LINES**: The graduated assignment algorithm for 2D pose estimation from line segments. This is used for pose refinement and verification in Algorithm **RANKED_POSE_HYPS** (Alg. 10).

inputs: A list of 2D image line segments,

A list of 2D model line segments,

An initial estimate T_0 of the 2D affine transformation of the model.

outputs: The final 2D affine transformation T of the model,

The assignment matrix M.

initialize $T = T_0$, $\beta = \beta_0$, $\delta = \delta_0$, $\epsilon = \infty$, $k = 0$, $maxsteps = 30$.

while $k < maxsteps$ **and** $\epsilon > \epsilon_{\mathbf{max}}$ **do**

Initialize M^0 according to Equation 6.8.

Apply Sinkhorn's algorithm (**SINKHORN3**) to M^0 to produce M.

Compute T'_k using Equations 6.10-6.13.

Compute T_k as the inverse of T'_k.

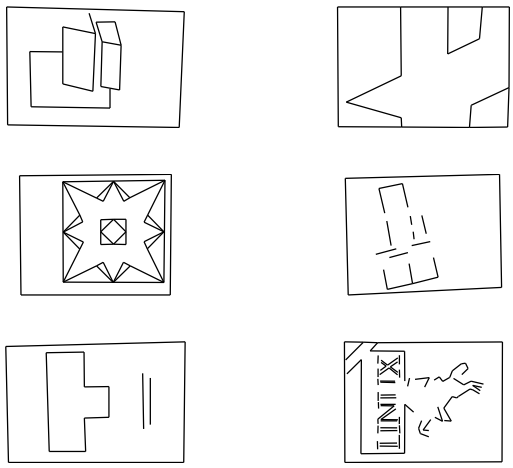
$\epsilon = \max |T_k - T_{k-1}|$.

$k = k + 1$.

$\delta = \delta - \left(\delta_{\text{final}} - \delta_0\right) / maxsteps$.

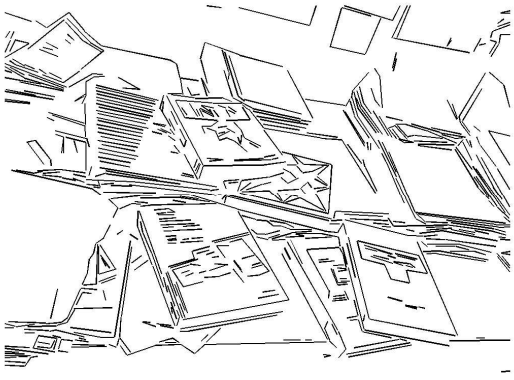
$\beta = \beta_{\text{update}} \times \beta$.

end



(a) Models of books.

(b) Test image.



(c) 731 detected lines.

(d) Recognized books.

Figure 6.10: Five books, some partially occluded, are recognized in a cluttered environment.

in the top 10 hypotheses of the sorted hypothesis list. One book ("Linux", shown in the lower right of Figure 6.10a) 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 6.10a 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

Figure 6.11: Image of a table of books taken by a camera tilted 50° from vertical.

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 6.11 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 6.12. 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 can achieve a probability of correct

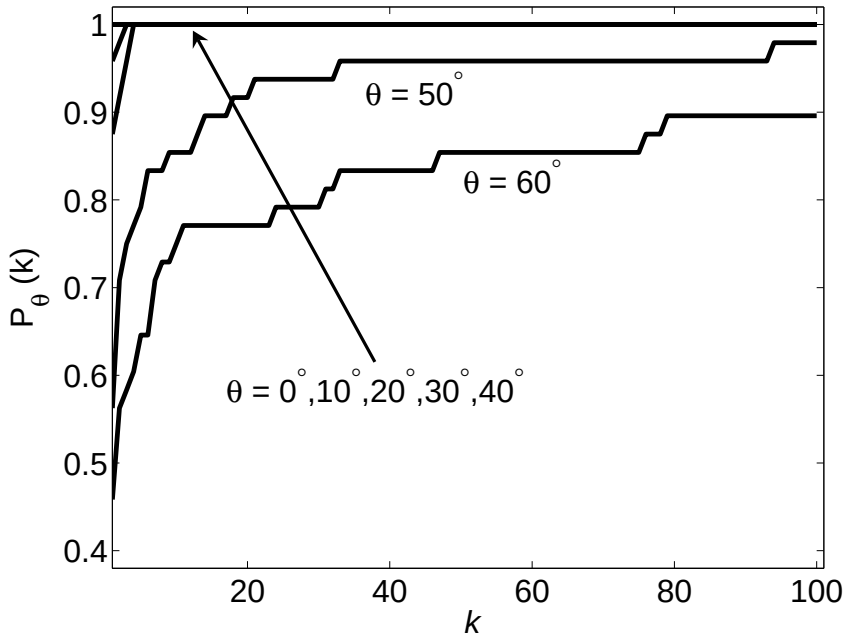


Figure 6.12: $P_\theta(k)$ is the probability that one of the first k sorted pose hypotheses for a model leads to a correct recognition for that model. θ is the difference in viewpoint elevation between the model and the object. For $\theta \leq 40°$, one of the four highest ranked pose hypotheses always leads to correct recognition. The curves for $\theta = 0°$ thru $40°$ are superposed for $k \geq 4$.

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 6.13 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.

We wish to estimate the maximum number of views (2D models) of a 2D or 3D object that our recognition algorithm must match to an image in order to ensure with high probability that the object will be correctly recognized if it is present in the image. Although our experiments involve only a small number of objects, we assume that these results are representative of a broader class of objects and therefore base our estimates on them. We ignore perspective effects and represent each object with

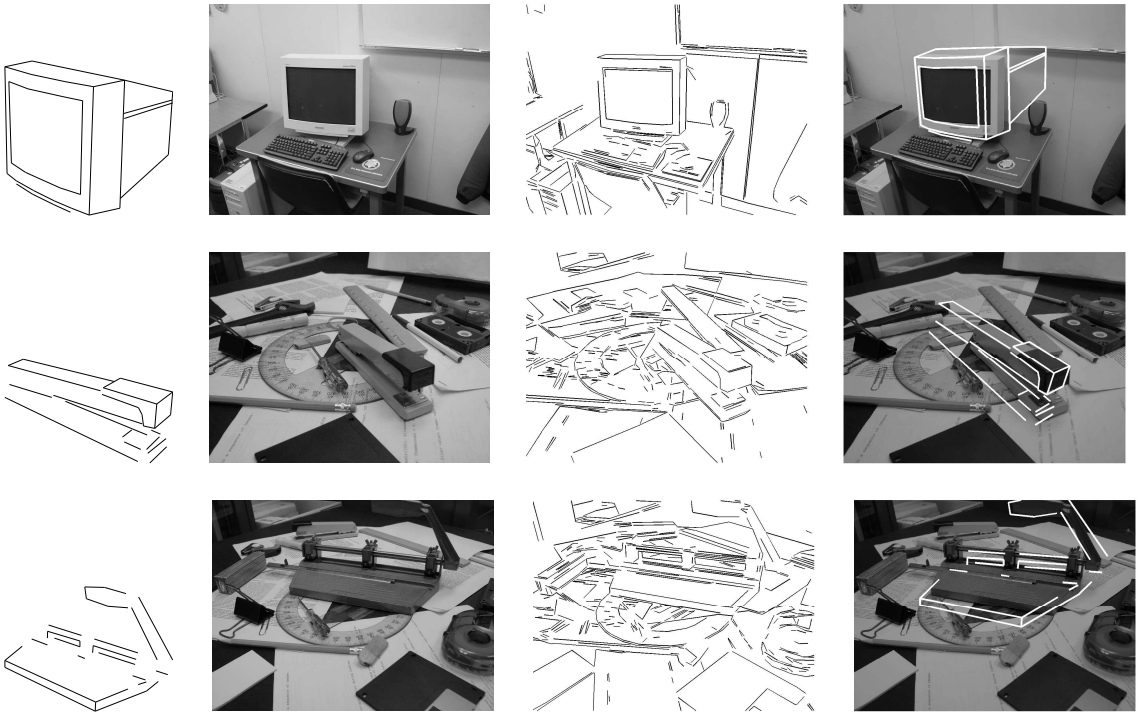
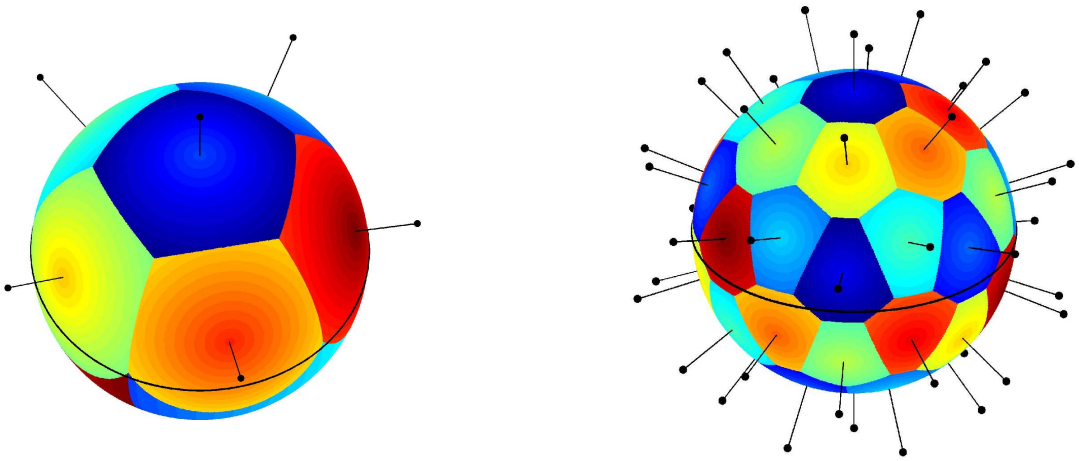


Figure 6.13: Recognition of 3D objects from viewpoint-dependent 2D models: computer monitor (top row), stapler (middle row), and hole punch (bottom row). Shown in each row, from left to right, is the 2D object model, original image, image lines, and model of recognized object overlaid on the original image. The modeled view of each object differs from the test view by 20° to 30°.

a set of models generated solely by changes in orientation; this should be sufficient provided the camera is not too close to the scene relative to the depth of the scene (perspective effects are dealt with in Chapter 7). Thus, the viewpoints of objects are represented as points on the surface of a unit sphere. To determine the number of models needed to represent an object, we must partition these viewpoints into a minimum number of nonoverlapping sets, which we call *view sets*, such that all viewpoints within a view set can be recognized by a single 2D model. The 2D model representing a view set will correspond to the center view of that set. How far a viewpoint in a view set can be from the center view is the experimentally determined maximum difference between the object's orientation and a modeled orientation that still allows the object to be recognized from that 2D model determines. Let this maximum difference between object and model orientation be denoted by Θ. To be conservative, based on results described above, we take $\Theta = 40°$ for 2D objects and $\Theta = 21°$ for 3D objects. (Taking $\Theta = 21°$ for 3D objects results in identical view sets as when $\Theta = 25°$.)

The minimum number of 2D models that are 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 (such as books on a table), 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 21° fully enclose the entire sphere. Thus,



(a) (b)

Figure 6.14: The view sets are shown for $\Theta = 40°$ (a) and $\Theta = 21°$ (b) that are needed to recognize 2D and 3D objects, respectively. Six view are needed to recognize 2D objects from any viewpoint above the plane of the object, and 44 views are needed to recognize 3D objects from any viewpoint. The upper and lower hemispheres are separated by a black line at the equator. Each colored patch represents the set of views that can be recognized using a 2D model generated for the center view in that set corresponding to the "pin". Each pin is an axis of a right circular cone whose vertex is located at the center of the sphere and whose angle is $\Theta = 40°$ in (a) and $\Theta = 21°$ in (b) . All viewpoints of an object that are within a given cone of viewpoints should be recognizable using the model generated from the viewpoint along the cones axis. This is a Voronoi diagram on the sphere for the set of center views; each colored patch is the set of all viewpoints that are closest to the center view of that patch.

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, and recognizing 3D objects from any viewpoint above the ground plane requires at most 22 2D models. The view sets for $\Theta = 40°$ and $\Theta = 21°$ are shown in Figure 6.14.

From the above experiments, it is apparent that a relatively 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 $\mathcal{O}(qmn \log(mn))$.

6.9 Conclusions

This chapter 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 the pose of an object 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.

Chapter 7

View-Based 3D Object Recognition

View-based object recognition algorithms model 3D objects with sets of 2D models of the objects in different poses. An object is recognized by matching one of the 2D models to the image. The advantage of this approach is that it allows a potentially simpler 2D algorithm to be used in place of a more complex 3D algorithm. Furthermore, when an image contains a large amount of clutter and occlusion, a 2D recognition algorithm should be less susceptible to being trapped in local minima because 2D poses are generally linear transformations with fewer degrees of freedom than the nonlinear perspective transformations required by 3D algorithms. The disadvantage of a view-based approach, however, is that because an object's appearance can vary greatly with its pose, many 2D models of each object might need to be examined in order to guarantee success.

We combine our recognition algorithm for 2D objects, **RANKED_POSE_HYPS**, described in Chapter 6, with either of our line-based 3D recognition algorithms, **SOFTPOSIT_LINE_ENDPOINTS** or **SOFTPOSIT_LINES**, described in Chapters 4 and 5, respectively, to produce a two-stage, view-based, 3D object recognition algorithm. This approach requires the availability of 3D CAD models of the objects to be recognized. These 3D models may be either wire-frame or solid models. The use of solid models allows hidden-line removal techniques to be used

when projecting 3D models into 2D models; 2D models with hidden lines removed are more accurate and less cluttered than 2D models generated from wire-frame models, and therefore they should provide more reliable matches in cluttered images.

The first stage of the view-based 3D recognition algorithm searches for a 3D camera orientation (i.e., viewing direction) for which the associated 2D model (the projection of the 3D model) provides a good match to the image. The camera orientations that must be examined were discussed in Section 6.8. For each camera orientation, the 3D model is positioned so that the camera axis passes through the center of the object and then the 3D model is projected into a 2D model. This 2D model is then matched to the image using **RANKED_POSE_HYPS**. The output of **RANKED_POSE_HYPS** is a 2D pose and an assignment matrix M between the image lines and the 2D model lines. When a 2D model is found that provides a good match to the image, the 3D algorithm **SOFTPOSIT_LINE_ENDPOINTS** (or **SOFTPOSIT_LINES**) is applied using the assignment matrix M to initialize the 3D pose of the 3D model. Because the 2D model is a projection of a 3D model, it is known which line segments in the 2D model correspond to which lines segments in the 3D model, so one can easily map the assignment matrix M between the image and 2D model lines to an assignment matrix M' between the image and the 3D model lines. M is not guaranteed to be perfect, nor does it need to be; the 2D model used to estimate M only approximates the expected appearance of the identified object. However, it is expected that the majority of the correspondences described in M will be correct, and therefore, a majority in M' will also be correct.

This should put the initial estimated 3D pose close enough to the true pose so that the 3D algorithm will avoid local minima and converge to the correct pose.

Algorithm 12 shows the pseudo-code for the 3D view-based object recognition algorithm. Figure 7.1 shows this algorithm applied to a robot image.

Algorithm 12 – 3D_OBJ_RECOGNITION: A view-based 3D object recognition algorithm.

inputs: List of 3D models $\mathcal{M}$, each consisting of a set of 3D line segments.

Input image $\mathcal{I}$, described by a set of 2D line segments.

outputs: List of model-pose pairs.

Create a range search data structure rs for the image lines $\mathcal{I}$.

for each 3D model $m_{3D} \in \mathcal{M}$ **do**

for each camera orientation $\boldsymbol{\theta}$ (as shown in Figure 6.14) **do**

Project m_{3D} to a 2D model m_{2D} using camera orientation $\boldsymbol{\theta}$.

Apply Algorithm **RANKED_POSE_HYPS** to rs and m_{2D}.

for each assignment matrix M returned by **RANKED_POSE_HYPS do**

if M describes a sufficient number of matches **then**

1. Map the 2D model assignments M to 3D model assignments M':

$$M'_{jk} = \begin{cases} M_{js} & \text{if } l_s \in m_{2D} \text{ is the projection of } l_k \in m_{3D}, \\ 0 & \text{otherwise.} \end{cases}$$

2. Apply Algorithm **SOFTPOSIT_LINE_ENDPOINTS** or **SOFTPOSIT_LINES** but use M' to compute the initial 3D rotation R and translation $\mathbf{T}$ as described in Section 5.3.1.

3. If a sufficient number of matches are found, then record this model and its 3D pose.

end

end

end

end

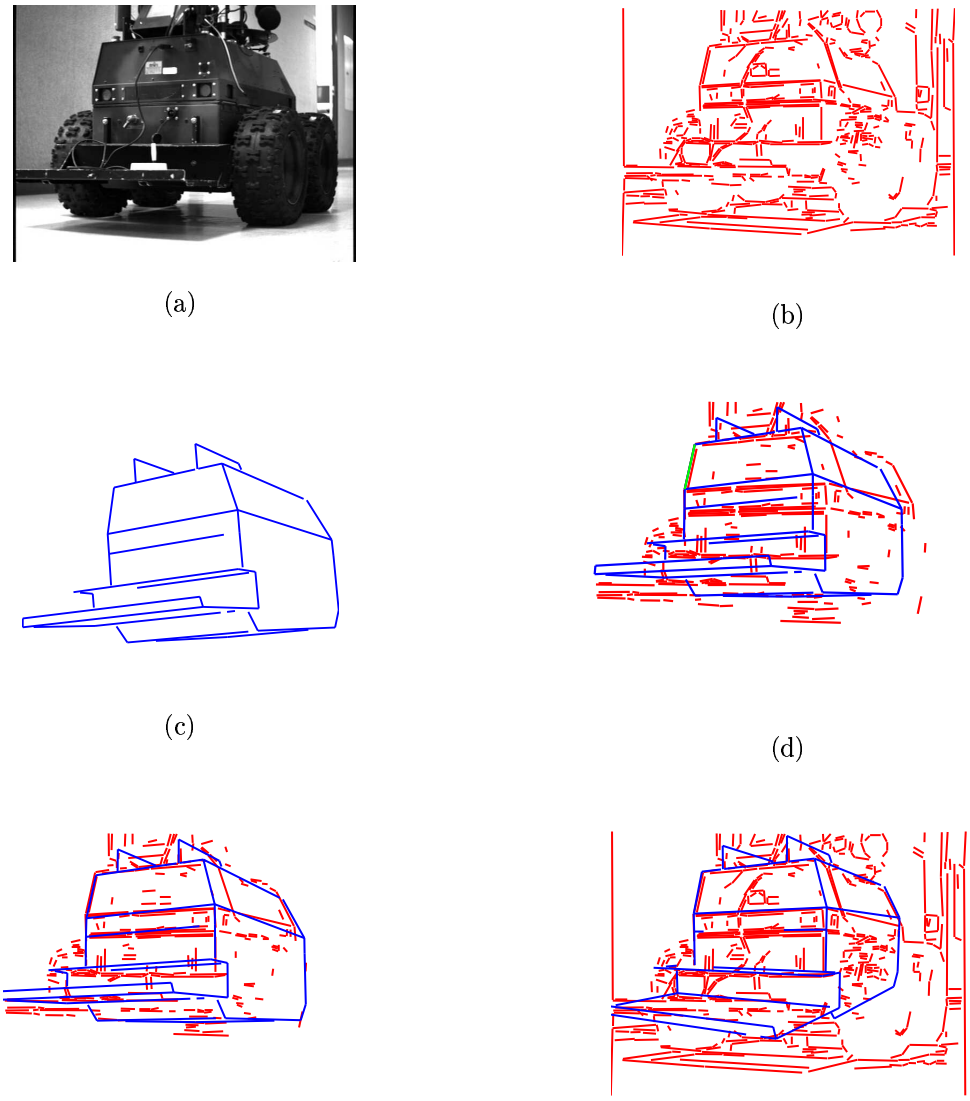


Figure 7.1: Example of view-based 3D object recognition. The 3D object recognition algorithm **SOFT_POSIT_LINES** is initialized using correspondences computed in the 2D object recognition algorithm **RANKED_POSE_HYPS**. (a) Original image. (b) Line segments detected in the image. (c) The 2D model used in the 2D algorithm; this was generated from a 3D CAD model using a hidden line removal algorithm. (d) The 2D affine pose hypothesis that leads to the highest-ranked 2D pose; the green line is the base correspondence. (e) The highest-ranked 2D pose after pose refinement. (f) The final 3D pose after initializing the pose using the 2D correspondences.

Chapter 8

Algorithm Comparison

In this chapter, we compare the performance of the three line-based object recognition algorithms described in Chapters 4, 5, and 6. Experiments are performed to determine how the success rate of the object recognition task is affected by the following four parameters: scene clutter rate, object feature detection rate, in-image-plane rotation of the object, and out-of-image-plane rotation of the object. Let m be the number of line segments in a 3D model of an object and n be the number of 2D line segments in an image of a scene that includes that object. Then, these four parameters are defined as follows.

1. The *scene clutter rate* is defined as $C = (n - m_1)/n$ where m_1 is the number of image line segments that are projections of some model line segment. The clutter rate is the percent of image line segments that are not the image of any model line segment.

2. The *object feature detection rate* is defined as $D = m_2/m_3$. Here, m_2 is the number of 3D model line segments that have one or more corresponding image line segments. A model line segment can have multiple corresponding image line segments when an image line is fragmented by a line detection algorithm. m_3 is the number of model line segments that are not fully self-occluded (i.e., not *fully* hidden by other parts of the *same* object); this is the number of

model line segments that should be at least partially visible to the camera when there are no objects in the scene between the modeled object and the camera. When simple wire-frame models of objects are used, our system has no knowledge of the surfaces or volumes of solid objects, so m_3 will equal m, the total number of line segments in the 3D model. However, when a solid model is available, hidden surface removal is performed; in this case, the value of m_3 will usually be less than m and will be highly dependent on the object's pose. The quantity $1 - D$ is the percent of model lines that are not detected as image lines as a result of being occluded by other objects in the scene or because of deficiencies in the line detection algorithm. Object lines that are not detected because they are self-occluded do not affect the object detection rate.

3. The *in-image-plane rotation* of an object, denoted Θ_1, is the angle of rotation of the object about an axis parallel to the camera's optical axis.

4. The *out-of-image-plane rotation* of an object, denoted Θ_2, is the angle of rotation of an object about some axis perpendicular to the camera's optical axis. An arbitrary 3D rotation of an object may be decomposed into a one-degree-of-freedom rotation about a specified axis and a two-degree of freedom rotation about an axis perpendicular to the first. The in-image-plane rotation is the first rotation, which is about the optical axis, and the out-of-image-plane rotation is the second rotation, which is about any axis perpendicular to the optical axis. These rotations are shown in Figure 8.1. The angle ϕ shown in

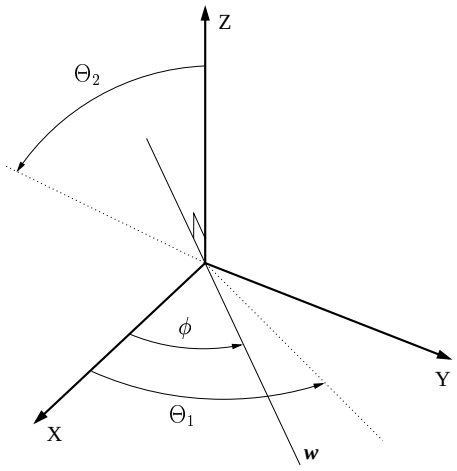


Figure 8.1: Θ_1 and Θ_2 are the rotational parameters of the experiments. The camera's optical axis is assumed to be aligned with the Z-axis in this figure. Θ_1 is the *in-image-plane rotation*, which is the rotation of the object about the Z-axis. Θ_2 is the *out-of-image-plane rotation*, which is the rotation of the object about the axis $\boldsymbol{w}$. $\boldsymbol{w}$ lies in the X-Y plane, so it's perpendicular to the Z-axis, and is at an arbitrary angle ϕ with respect to the X-axis.

this figure is not important to our experiments because, as will be described below, the 3D objects are random and therefore there will be no preferred angle ϕ that makes the recognition task easier.

These parameters determine the difficulty of an object recognition task. (Other parameters can also affect the difficulty, but we believe these are the most important parameters.) As the clutter rate C rises or the detection rate D drops, the problem generally becomes harder. Because our recognition algorithms always take $\Theta_1 = 0°$ and $\Theta_2 = 0°$ as an initial guess for an object's orientation, the larger Θ_1 or Θ_2, the more difficult the recognition task should become.

Simulated 3D objects and simulated images of these objects were used in our experiments so that the above parameters could be easily controlled. We performed two sets of experiments: one set in which the 3D models were generated by placing

the endpoints of the object line segments on a unit sphere, and the other set in which the objects were constructed of solid 3D boxes of random size and orientation (see below for more details). For every combination of the parameters, a number of independent trials were performed. In each of these trials, a single random 3D model was generated and an image of that model was created as determined by the the current parameter values: the object in the required pose was first projected into the image, some line segments on the image of the object were removed to simulate occlusion, and clutter line segments were added. Then, given the identical model and image, each of the three line-based object recognition algorithms were invoked to determine the pose of the object. Because the true pose of the object was known, the pose returned by each algorithm was judged correct if the computed 3D rotation and translation was sufficiently close to the true rotation and translation, respectively.

8.1 Random Lines on a Sphere

The first set of experiments were performed with random wire-frame objects constructed by placing the endpoints of the object's 3D line segments on the unit sphere. This is accomplished by first picking 36 random points on the unit sphere. Each point is then connected with a line segment to it's closest three neighbors. Finally, 36 of these line segments are randomly selected to form the 3D model. These are wire-frame objects with no volume or surface information, so hidden surface removal is not done during the simulated imaging process. Consequently,

there is no self-occlusion of these objects. Recognition of these wire-frame objects is likely more difficult than typically opaque real-world objects because it is possible for multiple "surfaces" of the wire-frame object to be seen simultaneously along a single line-of-sight, so that in effect, one or more surfaces of the object can be clutter for a different surface.

The parameters of the wire-frame experiments ranged over the following values:

$$C \in \{0.25,\ 0.5,\ 0.75\},$$

$$D \in \{0.4,\ 0.6,\ 0.8\},$$

$$\Theta_1 \in \{0°,\ 10°,\ 20°,\ 30°,\ 40°\},$$

$$\Theta_2 \in \{0°,\ 10°,\ 20°,\ 30°,\ 40°\}.$$

For every combination of these parameters, approximately seven independent trials were performed. Each trial consists of invoking the three object recognition algorithms on identical data. For each trail, a new random object and image was generated according to the current parameter values. The actual 3D orientation of the object was determined by Θ_1 and Θ_2 (with ϕ from Figure 8.1 set to zero), and the 3D translation was randomly selected to keep the projection of the object in the image. The initial guess used by the recognition algorithms for the pose of the object is always $\Theta_1 = 0$, $\Theta_2 = 0$, and $\phi = 0$, and a fixed translation centered at the mean translation. In addition to degrading the image due to the clutter and detection rate parameters as described above, the following processes are also performed in generating the image of an object:

1. The locations of the endpoints of the projected object line segments are perturbed by normally distributed noise with standard deviation equal to 0.25

percent of the image width.

2. The ends of the projected object line segments are chopped off to simulate that image line segments are typically not detected completely at junctions with other lines (i.e., at corners). The length chopped off the end of each line segment is uniformly distributed in the range of 0.4 to 0.6 percent of the width of the image.

3. When any two points on two image lines are closer than 0.25 percent of the width of the image, then all regions of the shorter line that are within this distance of the longer line are deleted.

4. Of the projected object line segments that remain (some are totally removed due to step 3 above, or due to the object feature detection rate), 5 percent of the internal regions (i.e., not connected to the endpoints) of these lines are deleted, simulating the fragmentation that commonly occurs with line detection algorithms.

Figure 8.2 shows a typical wire-frame object, its image, and the correctly recognized object.

The results of the experiments with the wire-frame objects are shown in Figure 8.3. One can see that the ranked pose hypothesis (RPH) algorithm performs significantly better than the two SoftPOSIT algorithms in terms of differences between the true and expected orientation of objects (as determined by Θ_1 and Θ_2), the clutter rate, and the object feature detection rate. From Figures 8.3 (a) and (c),

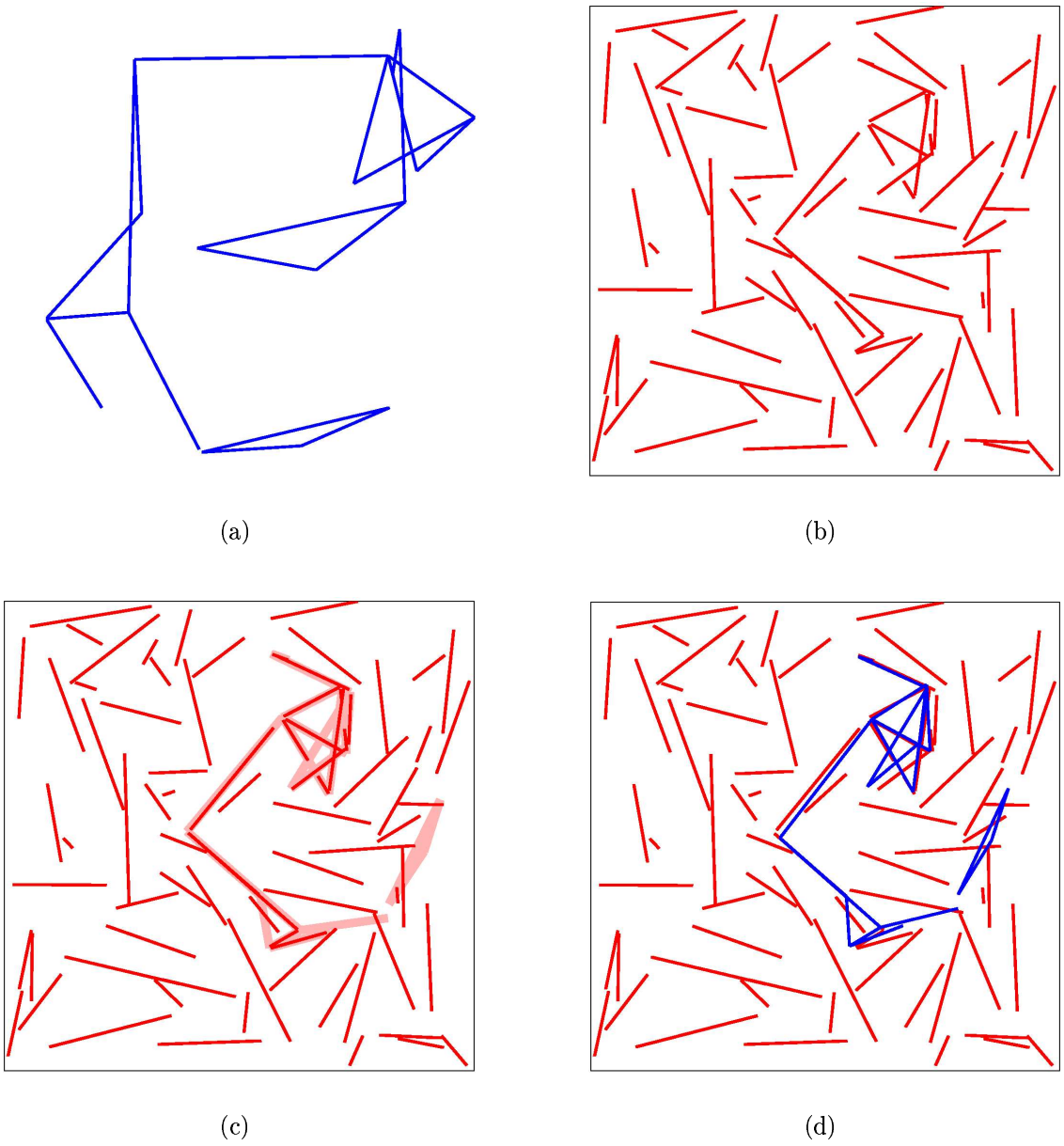


Figure 8.2: This is an example of the synthetic data where the endpoints of the object's line segments are constrained to lie on a unit sphere. (a) The projection of the model in a pose with no rotation, which is the initial guess for all of the recognition algorithms. (b) The image of the object with 50° of in-image-plane rotation, 20° of out-of-image-plane rotation, a clutter rate of 0.8, and an object feature detection rate of 0.7. (c) The thick pink lines show the projection of the model in its correct pose. Occluded object lines are indicated by thick pink lines that do not align with a red image line. (d) The projection of the model in a pose as determined by one of the recognition algorithms.

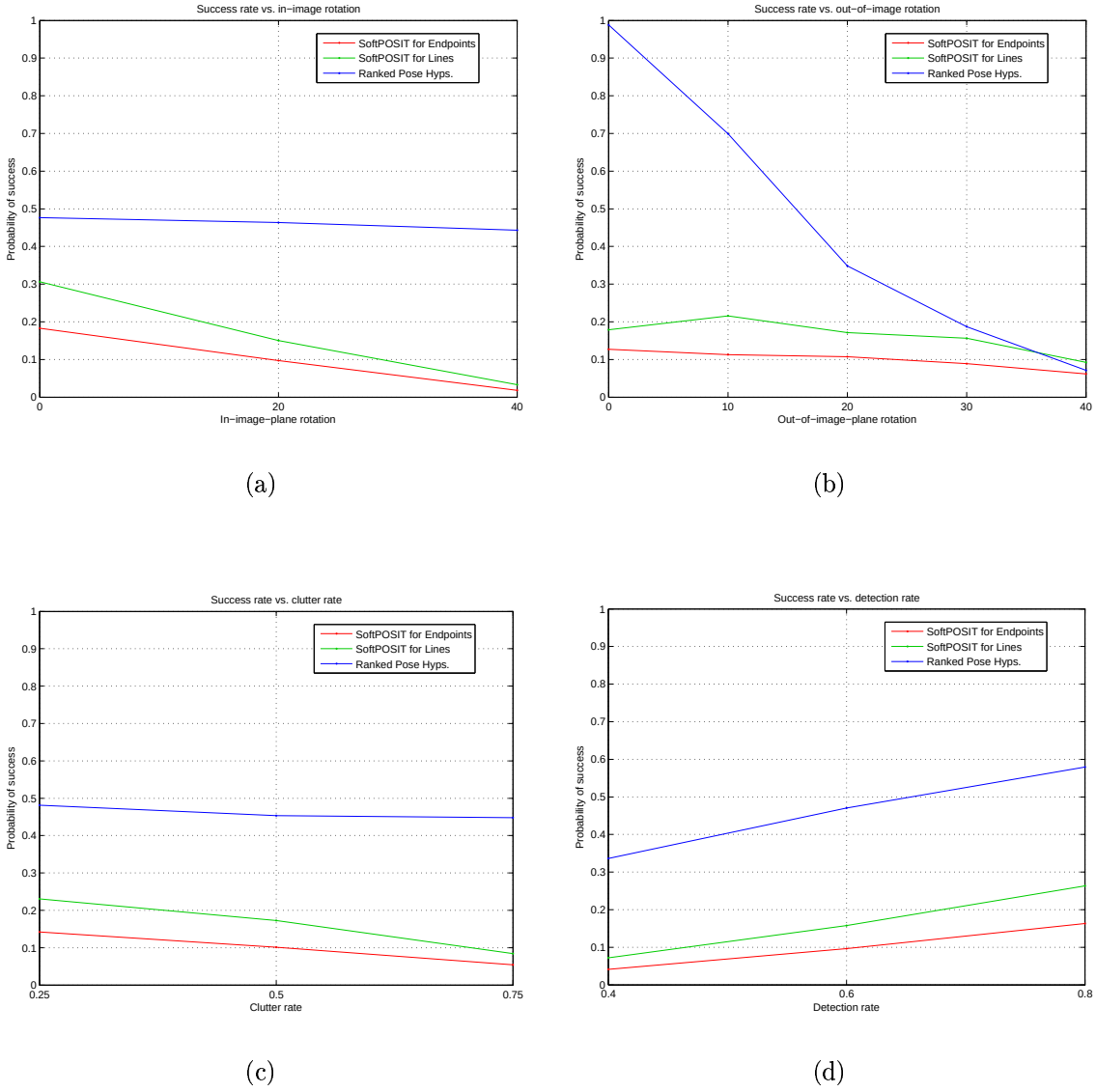


Figure 8.3: Rates of successful recognition for models constructed from random lines on a sphere. (a) Success rate as a function of Θ_1 marginalized over Θ_2, C, and D. (b) Success rate as a function of Θ_2 marginalized over Θ_1, C, and D. (c) Success rate as a function of C marginalized over Θ_1, Θ_2, and D. (d) Success rate as a function of D marginalized over Θ_1, Θ_2, and C.

we see that the performance of the RPH algorithm is largely unaffected by the in-image-plane rotation (Θ_1) of the object, and by the clutter rate (C), respectively. As expected, and as shown in Figure 8.3 (a), the two SoftPOSIT algorithms are affected by the in-image-plane rotation of the object; this is because, unlike the RPH algorithm, the model-to-image-feature distance measures that the SoftPOSIT algorithms employ are not invariant to image plane rotations. As shown in Figure 8.3 (b), The out-of-image-plane rotation has a large impact on the performance of the RPH algorithm, but smaller impact on the performance of SoftPOSIT algorithms. Finally, Figure 8.3 (d) shows that the performance of all of the algorithms is significantly affected by the object feature detection rate (D).

The probability of successful recognition in a single trial by the RPH algorithm is greater than 0.7 for all out-of-image-plane rotations up to 10°, even when marginalized over all values of Θ_1, C, and D. For out-of-image-plane rotations up to 20°, the probability of success is at least 0.35. To increase the probability of success in recognizing objects rotated by up to 20° out of the image plane and for any of the tested in-image-plane rotations, clutter rates, or object feature detection rates, the algorithm may be run multiple times using different initial guesses for the object's poses, where each of these poses is in the vicinity of the expected pose. For example, to ensure with probability 0.99 that at least one trial successfully recognizes an object when present, one would need to run the algorithm $\log(1-0.99)/\log(1-0.35) \approx 11$ times. The expected number of trials before an object is recognized, however, is only $1/0.35 \approx 3$. The two SoftPOSIT algorithms would not require a much larger number of initial guesses to recognize objects. For

example, the SoftPOSIT for Lines algorithm has a probability of success greater than 0.17 for all out-of-image-plane rotations up to 20°. Consequently, running this algorithm $\log(1 - 0.99)/\log(1 - 0.17) \approx 25$ times with different initial guesses for the pose of an object would ensure with probability 0.99 that that algorithm would recognize objects rotated by up to 20° out of the image plane and for any of the tested in-image-plane rotations, clutter rates, or object feature detection rates.

8.2 Random Boxes

The second set of experiments were performed with objects constructed from three solid 3D rectangular boxes (cuboids) of random size, position and orientation. The boxes are required to be close to each other, but are not allowed to intersect. The length, width, height, position, and orientation of each box is chosen randomly until three boxes are produced that don't intersect. Unlike the 3D objects generated from random line segments on a sphere (Section 8.1), the box objects are solid models with surface and volume information. This allows our simulated imaging process to do hidden surface removal. We feel that the box objects are more representative of man-made objects found in the real world: almost all real objects are opaque and solid, so that only one surface of the object can be seen along any line-of-sight; furthermore, many man-made objects can be accurately modeled using collections of rectangular boxes. The parameters for the box object experiments ranged over the following values:

$$C \in \{0.125,\ 0.25,\ 0.375,\ 0.5,\ 0.625,\ 0.75,\ 0.875\},$$

$$D \in \{0.4,\ 0.5,\ 0.6,\ 0.7,\ 0.8,\ 0.9\},$$

$$\Theta_1 \in \{0°,\ 10°,\ 20°,\ 30°,\ 40°\},$$

$$\Theta_2 \in \{0°,\ 5°,\ 10°,\ 15°,\ 20°,\ 25°,\ 30°,\ 35°,\ 40°\}.$$

The process of generating images of the random box objects is almost identical to the process described in Section 8.1 for random lines on a sphere. There are two differences, however. The first difference is that hidden surface removal is done when projecting the box objects into an image. The second difference is that, in addition to the individual random clutter lines that are placed into images, the images of random clutter boxes are also inserted into the image to create man-made-like clutter. If one is attempting to recognize man-made objects, then the background clutter in the scene is also likely man-made and should appear as such in images of those scenes. Figures 8.4 and 8.5 show typical box objects, their images, and the correctly recognized objects.

The results of the experiments with the box objects are shown in Figure 8.6. As with the previous experiments, one can see that the RPH algorithm performs significantly better than the two SoftPOSIT algorithms in terms of differences between the true and expected orientation of objects (as determined by Θ_1 and Θ_2), the clutter rate, and the object feature detection rate. When the out-of-image-plane rotation of a box object is no more than 10° from the initial guess, the probability that the RPH algorithm will successfully recognize that object in a single trial is 0.95, regardless of the in-image-plane rotation, clutter rate, or object feature detection rate. Comparing Figure 8.6 to Figure 8.3, one sees that the success rate of the

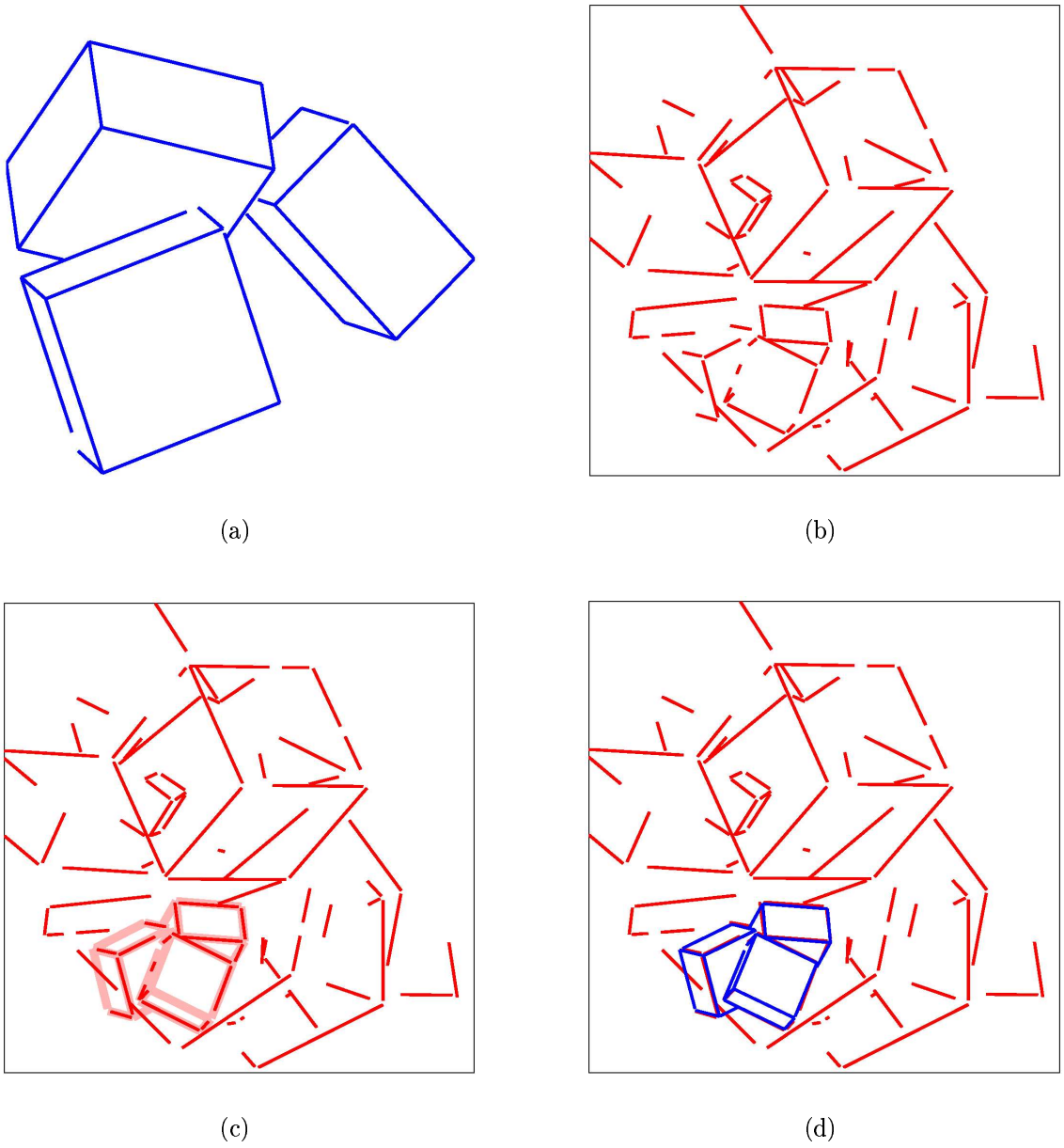
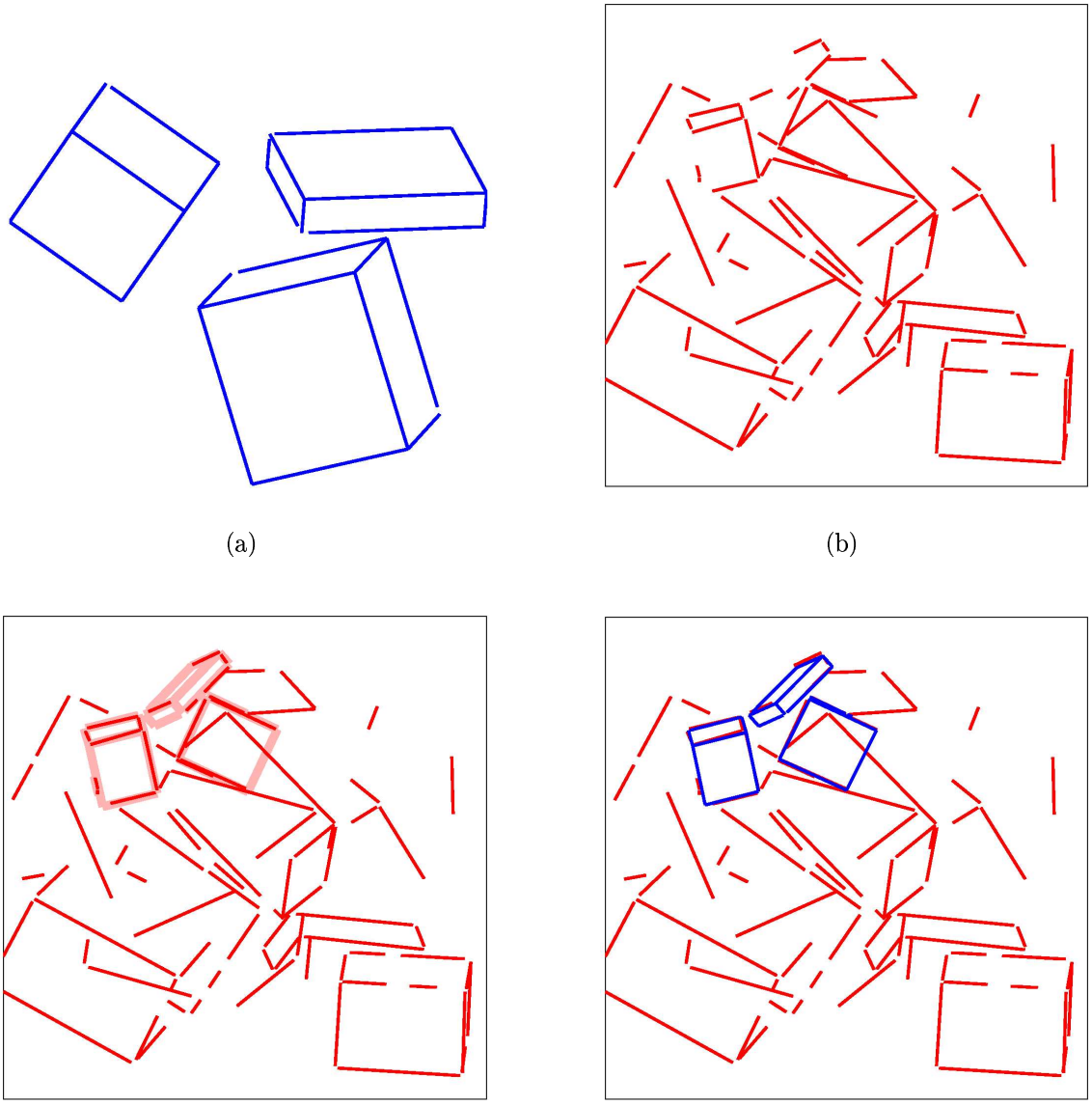


Figure 8.4: This is an example of the synthetic data where the objects are constructed from 3D boxes of random size and orientation. (a) The projection of the model in a pose with no rotation, which is the initial guess for all of the recognition algorithms. (b) The image of the object with 50° of in-image-plane rotation, 20° of out-of-image-plane rotation, a clutter rate of 0.8, and an object feature detection rate of 0.7. (c) The thick pink lines show the projection of the model in its correct pose. Occluded object lines are indicated by thick pink lines that do not align with a red image line. (d) The projection of the model in a pose as determined by one of the recognition algorithms.



(a) (b)

(c) (d)

Figure 8.5: This is another example of the synthetic random 3D box data. The parameters of this test and the figure subcaptions are the same as described in Figures 8.2 and 8.4. Notice that it isn't easy to determine where the object is in the image when comparing only (a) to (b).

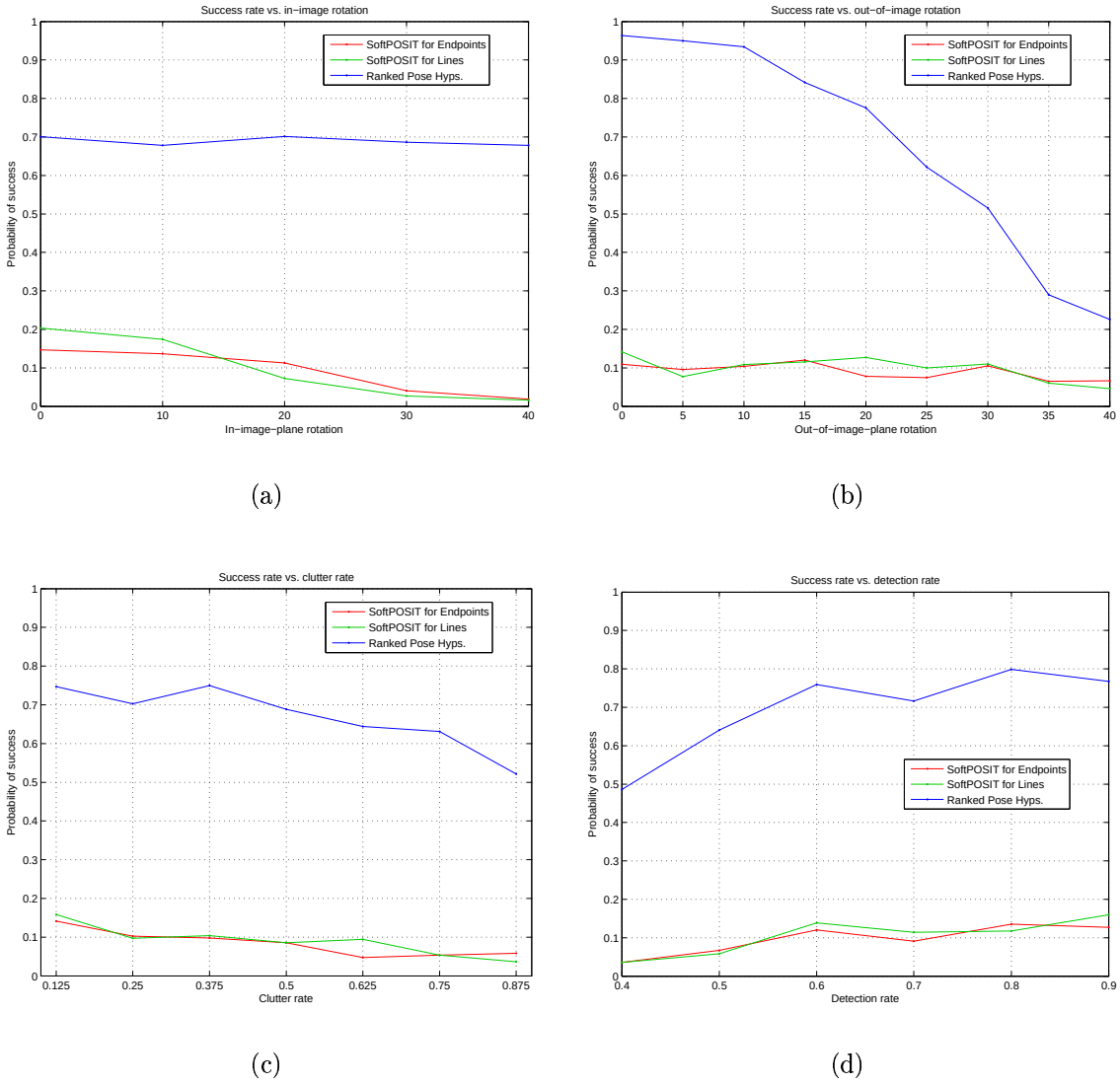


Figure 8.6: Rates of successful recognition for models constructed from random 3D boxes. (a) Success rate as a function of Θ_1 marginalized over Θ_2, C, and D. (a) Success rate as a function of Θ_2 marginalized over Θ_1, C, and D. (b) Success rate as a function of C marginalized over Θ_1, Θ_2, and D. (c) Success rate as a function of D marginalized over Θ_1, Θ_2, and C.

RPH algorithm is about 20 percent better on the box objects when compared to its performance in recognizing random lines on spheres, for all four parameters of the experiment. Unfortunately, there is not a similar improvement in the performance of the two SoftPOSIT algorithms. This disparity is possibly due to the fact that the 2D models that the RPH algorithm initially matches to the image are generated by projecting the 3D models (in their initial guessed poses) using hidden surface removal; most of object line segments that are hidden in the image do not appear in these 2D models. The SoftPOSIT algorithms, on the other hand, do not do hidden surface removal, but work with the 3D wire-frame (transparent) representations of the objects. When compared to the RPH algorithm, the SoftPOSIT algorithms are attempting to match many more model line segments that do not exist in the image.

8.3 Conclusions

We have seen that the ranked pose hypothesis algorithm performs significantly better than the two SoftPOSIT algorithms in terms of differences between the true and expected orientation of objects (as determined by Θ_1 and Θ_2), the clutter rate, and the object feature detection rate. The success of the ranked pose hypothesis algorithm is due to its use of an affine-invariant matching metric to determine a number of approximate object poses prior to applying the SoftPOSIT deterministic annealing process to refine the 3D poses and correspondences. The initial guesses for the 3D poses (as determined from the approximate 2D poses) are often close enough to the true 3D pose to allow the deterministic annealing process to converge

to the true pose and correspondences. Without the initial approximate poses, the two SoftPOSIT algorithms must start with an initial guess that is often far (relative to the "landscape" of the objective functions) from the true pose (even for small Θ_1 and Θ_2) and are consequently often trapped in local minima.

Chapter 9

Conclusions

This dissertation explored the use of the deterministic annealing method for applications of correspondence, pose estimation, and object recognition, using simple geometric features. All other known algorithms that solve these difficult combinatorial optimization problems for perspective cameras and objects in general position have high-order run-time complexities. One goal of this work was therefore to determine if the continuation method of deterministic annealing could be applied to produce more efficient algorithms. To this end, we posed the correspondence and pose estimation problem in terms of the optimization of an energy function that depends on the 3D pose of a model and on the correspondences between model and image features. An annealing control parameter was also inserted in the energy function to provide a controlled level of smoothing of this function.

We started with an algorithm (SoftPOSIT) to register 3D model points to sets of image points generated by a perspective camera. This is a very difficult problem because no additional information was associated with either the 3D or 2D points, and all points were considered in isolation (with no shape context). Still, the SoftPOSIT algorithm was able to register models to images when the initial guesses for the orientations of the 3D models was off by up to 30° from the correct orientation. That the algorithm performed this well is somewhat surprising

considering that in many of these cases, the human operator had no idea how to manipulate the 3D model in order to bring it into alignment with the image: a single image of a cloud of 3D points provides little information about the structure of those 3D points. The run-time complexity of SoftPOSIT with random starts was empirically determined to be $\mathcal{O}(mn^2)$ for m model points and n image points. This is a factor of n better than the complexity of any published algorithm that solves the pose and correspondence problem for point features under a full perspective camera model.

Next, we extended the SoftPOSIT algorithm from matching point features to the case of matching line features: 3D model line segments were matched to image line segments 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. This algorithm used the SoftPOSIT algorithm for points to determine the pose and correspondences for a set of image and model lines. An iteration was performed where at each step the given 2D to 3D line correspondence problem was mapped to a new 2D to 3D point correspondence problem which depended on the current estimate of the camera pose: using the current pose estimate, the endpoints of the 3D line segments were first mapped to a set of virtual image points lying on image lines, and then SoftPOSIT for points was applied to the 3D endpoints and the virtual image points to improve the estimate of the camera pose.

The SoftPOSIT algorithm was extended a second time from matching point features to matching line features. This time, instead of mapping the line-based

problem into a point based-problem, the pose and correspondence problems were solved directly using the line features.

In these 3D pose and correspondence algorithms, we encountered some difficulties in using deterministic annealing to smooth the energy function: too little smoothing at the start of the annealing process would cause the model's pose to quickly move to a nonoptimal local minimum; conversely, too much smoothing at the start of the annealing process would cause the model's pose to undergo random changes early in the annealing process, enough so that the model's pose was no longer in the basin of attraction of the optimal solution (the correct pose) when the annealing temperature dropped enough to start enforcing the matching constraints. In general, we found that a good initial guess for the object's pose and a judiciously selected annealing schedule were essential to finding a correct pose.

We surmised that the energy function associated with a 2D correspondence and pose algorithm using a 2D linear transformation would have fewer local optima to become trapped in than a 3D algorithm using 3D perspective projection. We therefore developed a 2D pose and correspondence algorithm for line features. In this approach, corresponding line features were determined by a three-stage process. The first stage generated 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 were evaluated and ranked by comparing local image neighborhoods to the corresponding local model neighborhoods. Fast nearest neighbor and range search algorithms were used to implement a distance measure that was unaffected by clutter and partial occlusion. The ranking of pose hypothe-

ses is invariant to changes in image scale, orientation, and partially invariant to affine distortion. Finally, deterministic annealing was applied for refinement and verification, starting from the few best approximate poses produced by the previous stages.

To get back to recognizing 3D objects from perspective images, we integrated the 2D pose algorithm with our line-based 3D algorithms to produce a view-based 3D recognition algorithm. Rough correspondences were obtained from matching 2D projections of models to the images. We found that using a coarse identification of an object using an approximate 2D model provided correspondences that usually were accurate enough to enable a good initialization of the 3D pose algorithms, and then a good 3D pose could be determined relatively efficiently: since we had a good initial guess for the 3D pose, the annealing temperature could be more easily set to prevent the solution from diverging from the correct pose. The runtime complexity of this recognition algorithm for a single model consisting of m line segments and an image with n line segments was empirically determined to be $\mathcal{O}(mn \log(mn))$. This is a factor of n better than the complexity of any published algorithm that solves the pose and correspondence problem for line features under a full perspective camera model.

All of our algorithms used simple geometric point and line features. These features are not very descriptive, but they are easy to locate in images. Many real applications may have nongeometric information associated with these types of features (e.g., color, texture properties, etc.) which may be used to eliminate from consideration many outlier correspondences. This type of information would be easy

to integrate into our algorithms, and would provide a significant performance boost.

Appendix A

Complexity of RANSAC

The asymptotic complexity of the RANSAC algorithm [49] to model-to-image registration for a 3D model and a perspective camera is derived in this appendix. We first define a few parameters. Let:

- m be the number of 3D model points,
- n be the number of image points,
- f be the fraction of model points that are present (non-occluded) in the image,
- r be the desired probability of success (i.e., of finding a good pose).

Given a set of data with outlier rate w, it is well known [49] that the number, k, of random samples of size d of that data that must be examined in order to ensure with probability z that at least one of those samples is outlier-free is

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

We need to determine how this number of samples depends on m, n, f, and r for the hypothesize-and-test algorithm for large values of m and n.

Because we assume that the RANSAC algorithm has no *a priori* information about which correspondences are correct, correspondences are formed from randomly chosen model and image points. We assume that three correspondences are used to

estimate the object's pose. Let $s = fm$ be the number of detected (non-occluded) model points in the image. For a correspondence to be correct, two conditions must be satisfied: the object point must be non-occluded, and the image point must correspond to the object point. The probability that the i^{th} $(i = 1, 2, 3)$ randomly chosen correspondence is correct given that all previously chosen correspondences are also correct is the probability that these two conditions are satisfied, which is

$$\frac{s - i + 1}{m - i + 1} \cdot \frac{1}{n - i + 1}.$$

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

$$\frac{s(s-1)(s-2)}{m(m-1)(m-2)n(n-1)(n-2)} \approx \frac{s^3}{m^3 n^3} = \left(\frac{f}{n}\right)^3.$$

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

$$\left(1 - \left(\frac{f}{n}\right)^3\right)^T.$$

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

$$1 - \left(1 - \left(\frac{f}{n}\right)^3\right)^T \geq r.$$

Solving for T, we get

$$T \geq \frac{\log(1-r)}{\log\left(1-\left(\frac{f}{n}\right)^3\right)}.$$

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

$$T \approx \left(\frac{n}{f}\right)^3 \log\left(\frac{1}{1-r}\right).$$

Since each sample requires $\mathcal{O}(m \log n)$ time for back-projection and verification (assuming an efficient point location algorithm is used to search for image points[1]), the complexity of general RANSAC algorithm is

$$\left(\frac{n}{f}\right)^3 \log\left(\frac{1}{1-r}\right) \times \mathcal{O}(m \log n) = \mathcal{O}(mn^3 \log n).$$

Appendix B

Scaled Orthographic Image Points

Here we give a geometric interpretation of the relation between perspective and scaled orthographic image points. Consider figure 3.1. A plane Π' parallel to the image plane Π is chosen to pass through the origin P_0 of the object coordinate system. This plane cuts the camera axis in H ($OH = T_z$). The point P projects on plane Π' in P', and the image of P' on the image plane Π is called p'.

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

image of P'' on the image plane Π is called p''.

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

Consider again the equations of perspective (equations (3.1, 3.2)):

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

with $w = \mathbf{R}_3 \cdot \boldsymbol{P_0P}/T_z + 1$. We can see that $\boldsymbol{cp'} = s(\mathbf{R}_1 \cdot \boldsymbol{P_0P} + T_x, \mathbf{R}_2 \cdot \boldsymbol{P_0P} + T_y)$. Indeed the terms in parentheses are the x and y camera coordinates of P and therefore also of P', and the scaling factor s scales down these coordinates to those of the image p' of P'. In other words, the column vector of the right-hand side of equation (3.4) represents the vector $\boldsymbol{cp'}$ in the image plane.

On the other hand, $\boldsymbol{cp''} = (wx, wy) = w\boldsymbol{cp}$. Indeed the z-coordinate of P in the camera coordinate system is $\mathbf{R}_3 \cdot \boldsymbol{P_0P} + T_z$, i.e. wT_z. It is also the z-coordinate of P_L. Therefore $\boldsymbol{OP_L} = wT_z\boldsymbol{Op}/f$. The x and y camera coordinates of P_L are also those of P'', and the scaling factor $s = f/T_z$ scales down these coordinates to those of the image p'' of P''. Thus $\boldsymbol{cp''} = w\boldsymbol{cp}$. In other words, the column vector of the left-hand side of equation (3.4) represents the vector $\boldsymbol{cp''}$ in the image plane. The image point p'' can be interpreted as a correction of the image point p from a perspective projection to a scaled orthographic projection of a point P_L located on the line of sight at the same distance as P.

BIBLIOGRAPHY

[1] S.T. Acton and A.C. Bovik, "Generalized Deterministic Annealing," *IEEE Trans. on Neural Networks*, vol. 7, no. 3, pp. 686–699, 1996.

[2] A. Ansar and K. Daniilidis, "Linear Pose Estimation from Points or Lines," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 25, no. 5, pp. 578–589, 2003.

[3] S. Arya, D.M. Mount, N.S. Netanyahu, R.Silverman, and A. Wu, "An Optimal Algorithm for Approximate Nearest Neighbor Searching," *Journal of the ACM,* vol. 45, no. 6, pp. 891–923, 1998.

[4] 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, 1986.

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

[6] D.H. Ballard, "Generalizing the Hough Transform to Detect Arbitrary Shapes," *Pattern Recognition,* vol. 13, no. 2, pp. 111–122, 1981.

[7] D.H. Ballard, "Parameter Networks: Towards a Theory of Low-Level Vision," in *Proc. 7th Int. Joint Conf. on Artificial Intelligence,* Vancouver, B.C, pp. 1068–1078, August 1981.

[8] I. Beichl and F. Sullivan, "The Metropolis Algorithm," IEEE Computing in Science & Engineering, vol. 2, no. 1, pp. 65–69, 2000.

[9] J.S. Beis and D.G. Lowe, "Indexing Without Invariants in 3D Object Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 10, pp. 1000–1015, 1999.

[10] B. Besserer, S. Estable, and B. Ulmer, "Multiple knowledge sources and evidential reasoning for shape recognition," *Proc. IEEE 4th Int. Conf. on Computer Vision,* pp. 624–631, May 1993.

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

[12] 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.

[13] P. Brand and R. Mohr, "Accuracy in Image Measure," *Proc. SPIE, Videometrics III,* pp. 218–228, October 31– November 4, 1994, Boston, U.S.A.

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

[15] L.G. Brown, "A Survey of Image Registration Techniques," *ACM Computing Surveys*, vol. 24, no. 4, pp. 325–376, 1992.

[16] J.B. Burns, R.S. Weiss, and E.M. Riseman, "View Variation of Point-Set and Line-Segment Features," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 15, no. 1, pp. 51–68, 1993.

[17] T.M. Breuel, "Model Based Recognition Using Pruned Correspondence Search," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 257–262, 1991.

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

[19] R.A. Brooks, "Symbolic Reasoning Among 3D Models and 2D Images". Artificial Intelligence, Vol. 17, pp. 285–348, 1981.

[20] J. Burns, A. Weiss, and E. Riseman, "Extracting Straight Lines," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 8, no. 4, pp. 425–456, 1986.

[21] J.F. Canny, "A Computational Approach to Edge Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 8, no. 6, pp. 34–43, 1986.

[22] 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, 2004.

[23] T.A. Cass. "Polynomial-Time Object Recognition in the Presence of Clutter, Occlusion, and Uncertainty," *Proc. 2nd European Conf. on Computer Vision,* pp. 834–842, Springer-Verlag, 1992.

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

[25] T.A. Cass. "Robust Affine Structure Matching for 3D Object Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 11, pp. 1265–1274, 1998.

[26] H.H. Chen, "Pose Determination from Line-to-Plane Correspondences: Existence Condition and Closed-Form Solutions," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 6, pp. 530–541, 1991.

[27] S. Christy and R. Horaud, "Iterative Pose Computation from Line Correspondences," *Computer Vision and Image Understanding,* vol. 73, no. 1, pp. 137–144, 1999.

[28] P. David, D.F. DeMenthon, R. Duraiswami, and H. Samet, "SoftPOSIT: Simultaneous Pose and Correspondence Determination," *Proc. of the European Conference on Computer Vision,* Copenhagen, Denmark, pp. 698-714, May 2002.

[29] P. David, D. DeMenthon, R. Duraiswami, and H. Samet. "Evaluation of the SoftPOSIT Model to Image Registration Algorithm," Center for Automation Research Technical Report CAR-TR-974, CS-TR-4340, July 2002.

[30] P. David, D. DeMenthon, and R. Duraiswami, "Simultaneous Pose and Correspondence Estimation for Autonomous Robotic Vehicles," *Proc. of the Army Science Conference*, Orlando, FL, December 2002.

[31] 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,* Madison WI, vol. 2, pp. II-424–II-431, June 2003.

[32] P. David, D. DeMenthon, R. Duraiswami, and H. Samet, "SoftPOSIT: Simultaneous Pose and Correspondence Determination," *Int. Journal of Computer Vision,* vol. 59, no. 3, pp. 259–284, September-October, 2004.

[33] P. David and D. DeMenthon, "Object Recognition by Deterministic Annealing of Ranked Affine Pose Hypotheses," University of Maryland Technical Report CS-TR-4731, July 2005.

[34] P. David and D. DeMenthon, "Object Recognition in High Clutter Images Using Line Features," *Proc. Tenth IEEE Int. Conf. on Computer Vision*, Beijing, China, pp. 1581–1588, October 2005.

[35] L.S. Davis, "Hierarchical Generalized Hough Transforms and Line-Segment Based Generalized Hough Transforms," *Pattern Recognition,* vol. 15, no. 4, pp. 277–285, 1982.

[36] F. Dellaert, S.M. Seitz, C.E. Thorpe, and S. Thrun, "EM, MCMC, and Chain Flipping for Structure from Motion with Unknown Correspondence", *Machine Learning*, vol. 50, no. 1-2, pp. 45–71, 2003.

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

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

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

[40] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society*, B, vol. 39, no. 1, pp. 1–38, 1977.

[41] 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.

[42] M. Dhome, M. Richetin, J.-T. Lapreste, and G. Rives, "Determination of the Attitude of 3D Objects from a Single Perspective View," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 11, no. 12, pp. 1265–1278, 1989.

[43] M.W.M.G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, M. Csorba, "A Solution to the Simultaneous Localization and Map Building (SLAM) Problem," *IEEE Trans. on Robotics and Automation*, vol. 17, no. 3, pp. 229-241, 2001.

[44] 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.

[45] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification, 2nd Ed.*, John Wiley & Sons, New York, NY, 2001.

[46] R.W. Ely, J.A. Digirolamo, and J.C. Lundgren, "Model Supported Positioning," *Integrating Photogrammetric Techniques with Scene Analysis and Machine Vision II, SPIE Aerospace Sensing and Dual Use Sensors and Controls,* Orlando, April 1995.

[47] C. Fennema, A. Hanson, E. Riseman, J.R. Beveridge, R. Kumar, "Model-Directed Mobile Robot Navigation," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 20, no. 6, pp. 1352–1369, 1990.

[48] P.D. Fiore. "Efficient Linear Solution of Exterior Orientation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 2 , pp. 140–148, February 2001.

[49] 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.

[50] D. Forsyth, J.L. Mundy, A. Zisserman, C. Coelho, A. Heller, and C. Rothwell, "Invariant Descriptors for 3D Object Recognition and Pose," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 10, pp. 971–991, 1991.

[51] D. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*, New Jersey, Prentice Hall, 2003.

[52] S. Ganapathy, "Decomposition of Transformation Matrices for Robot Vision," *Proc. 1984 IEEE Int. Conf. on Robotics and Automation,* vol. 1, pp. 130–139, March 1984.

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

[54] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721–741, 1984.

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

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

[57] C. Goodall, "Procrustes Methods in the Statistical Analysis of Shape," *Journal of the Royal Statistical Society B,* vol. 53, no. 2, pp. 285–339, 1991.

[58] W.E.L. Grimson and T. Lozano-Perez, "Localizing Overlapping Parts by Searching the Interpretation Tree," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 9, no. 4, pp. 468–482, 1987.

[59] W.E.L. Grimson, *Object Recognition by Computer: The Role of Geometric Constraint,* MIT Press, Cambridge, MA, 1990.

[60] W.E.L. Grimson and D.P. Huttenlocher. On the Sensitivity of the Hough Transform for Object Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 3, pp. 255-274, 1990.

[61] W.E.L. Grimson and D.P. Huttenlocher, "On the Verification of Hypothesized Matches in Model-Based Recognition", *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 12, pp. 1201–1213, December 1991.

[62] W.E.L. Grimson, D.P. Huttenlocher, and T.D. Alter, "Recognizing 3D Objects from 2D Images: An Error Analysis," *Proc. 1992 IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 316–321, June 1992.

[63] M. Grotschel and L. Lovasz, "Combinatorial Optimization: A Survey," *Handbook of Combinatorics*, North-Holland, 1993.

[64] R.M. Haralick and L.G. Shapiro, "The Consistent Labeling Problem: Part 1," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 1, no. 2, pp. 173–184, 1979.

[65] R.M. Haralick, H. Joo, C. Lee, X. Zhuang, V.G. Vaidya, and M.B. Kim, "Pose Estimation from Corresponding Point Data," *IEEE Transactions on Systems, Man and Cybernetics,* vol. 19, no. 6, pp. 1426–1446, 1989.

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

[67] R.I. Hartley, "Minimizing Algebraic Error in Geometric Estimation Problems," *Proc. Sixth Int. Conf. on Computer Vision,* pp. 469–476, Jan. 1998.

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

[69] C.G. Harris and M.J. Stephens, "A Combined Corner and Edge Detector," *Proc. Fourth Alvey Vision Conference,* Manchester, pp. 147–151, 1988.

[70] Y.C. Hecker and R.M. Bolle, "On Geometric Hashing and the Generalized Hough Transform ," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1328–1338, 1994.

[71] A Hill and C. J. Taylor, "Model-Based Image Interpretation Using Genetic Algorithms," *Image and Vision Computing*, no. 5, pp. 295–300, 1992.

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

[73] B.K.P. Horn, *Robot Vision*, MIT Press, Cambridge, Massachusetts 1986.

[74] B.K.P. Horn, "Closed Form Solution of Absolute Orientation Using Unit Quaternions," *Journal of the Optical Soc. Am. A,* vol. 4, no. 4, pp. 629–642, 1987.

[75] B.K.P. Horn, H. M. Hilden, and S. Negahdaripour, "Closed Form Solution of Absolute Orientation using Orthonormal Matrices," *Journal of the Optical Soc. Am. A,* vol. 5, no. 7, pp. 1127–1135, 1988.

[76] B.K.P. Horn, "Relative Orientation," *Int. Journal of Computer Vision,* vol. 4, no. 1, pp. 59–78, 1990.

[77] P.V.C. Hough, "Method and means for recognizing complex patterns," U.S. Patent 3,069,654, December 1962.

[78] T.S. Huang and A.N. Netravali, "Motion and Structure from Feature Correspondences: A Review," Proc. of the IEEE, vol. 82, no. 2, pp. 252–268, 1994.

[79] D.P. Huttenlocher and S. Ullman, "Recognizing Solid Objects by Alignment with an Image," *Int. Journal of Computer Vision,* vol. 5, no. 2, pp. 195–212, 1990.

[80] D.P. Huttenlocher, G.A. Klanderman, and W.A. Rucklidge, "Comparing Images Using the Hausdorff Distance," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 850–863, 1993.

[81] J. Illingworth and J. Kittler, "A Survey of the Hough Transform," *Computer Vision, Graphics, and Image Processing,* vol. 44, pp. 87–116, 1988.

[82] L. Ingber, "Simulated Annealing: Practice Versus Theory", *Mathl. Comput. Modelling*, vol. 18, no. 11. pp. 29–57, 1993.

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

[84] D.W. Jacobs, "Matching 3-D Models to 2-D Images," *Int. Journal of Computer Vision*, vol. 21, no. 1/2, pp. 123–153, 1997.

[85] F. Jurie, "Solution of the Simultaneous Pose and Correspondence Problem Using Gaussian Error Model," *Computer Vision and Image Understanding,* vol. 73, no. 3, pp. 357–373, 1999.

[86] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[87] 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/>.

[88] R. Kumar and A. Hanson, "Robust Methods for Estimating Pose and a Sensitivity Analysis," *Computer Vision and Image Understanding,* vol. 60, no. 3, pp. 313–342, 1994.

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

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

[91] G. Lei, "Recognition of Planar Objects in 3-D Space from Single Perspective Views Using Cross Ratio," *IEEE Trans. Robotics and Automation*, vol. 6, no. 4, pp. 432–437, 1990.

[92] S. Linnainmaa, D. Harwood, and L.S. Davis, "Pose Determination of a Three-Dimensional Object Using Triangle Pairs," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 10, no. 5, pp. 634–647, Sept. 1988.

[93] Y. Liu, T.S. Huang, and O.D. Faugeras, "Determination of Camera Location from 2-D to 3-D Line and Point Correspondences," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 12, no. 1, pp. 28–37, 1990.

[94] Y. Liu, L. Li, and Y. Wang, "Free Form Shape Matching Using Deterministic Annealing and Softassign," *Proc. Int. Conf. on Pattern Recognition*, vol. 2, pp. 128–131, 2004.

[95] D.G. Lowe, "Three-Dimensional Object Recognition from Single Two-Dimensional Images," *Artificial Intelligence,* vol. 31, no. 3, pp. 355-395, 1987.

[96] D.G. Lowe, "Fitting Parameterized Three-Dimensional Models to Images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 5, pp. 441–450, 1991.

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

[98] C.-P. Lu, G.D. Hager, and E. Mjolsness, "Fast and Globally Convergent Pose Estimation from Video Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 6, pp. 610–622, June 2000.

[99] M. Luxen and W. Forstner, "Optimal Camera Orientation from Points and Straight Lines," *Proc. of the 23rd DAGM Symposium for Pattern Recognition,* vol. 2191, pp. 84–91, Munich, Germany, 2001.

[100] J. Maciel and J.P. Costeira, "A Global Solution to Sparse Correspondence Problems," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 187-199, Feb. 2003.

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

[102] E.M. Mikhail, J.S. Bethel, and J.C. McGlone, *Introduction to Modern Photogrammetry*, New York, John Wiley & Sons, 2001.

[103] K. Mikolajczyk, A. Zisserman, and C. Schmid, "Shape Recognition with Edge-Based Features," *Proc. British Machine Vision Conference*, vol. 2, pp. 779–788, September 2003.

[104] H. Moon, R. Chellappa, and A. Rosenfeld, "3D Object Tracking Using Shape-Encoded Particle Propagation," *Proc. 2001 IEEE Int. Conf. on Computer Vision*, vol. 2, pp. 307–314, July 2001.

[105] T.K. Moon, "The Expectation-Maximization Algorithm," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 47–60, 1996.

[106] H.P. Moravec, "Visual Mapping by a Robot Rover," *Proc. of the 5th Int. Joint Conf. on Artificial Intelligence*, pp. 584–600, Cambridge, MA, 1977.

[107] G. Mori, S. Belongie, and J. Malik, "Efficient Shape Matching Using Shape Contexts," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 11, pp. 1832–1837, 2005.

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

[109] J.L. Mundy and A. Zisserman, eds., *Geometric Invariance in Computer Vision*, Cambridge, MA, MIT Press, 1992.

[110] H. Murase and S.K. Nayar, "Visual Learning and Recognition of 3-D Objects from Appearance," *Int. Journal of Computer Vision,* vol. 14, no. 1, pp. 5–24, 1995.

[111] N. Navab and O.D. Faugeras, "Monocular Pose Determination From Lines: Critical Sets and Maximum Number of Solutions," *Proc. 1993 IEEE Conf. on Computer Vision and Pattern Recognition,* pp. 254–260, June 1993.

[112] S.K. Nayar, S.A. Nene, and H. Murase, "Real-Time 100 Object Recognition System," *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 2321–2325, 1996.

[113] D. Noll and W. von Seelen "Object Recognition by Deterministic Annealing," *Image and Vision Computing*, vol. 15, no. 11, pp. 855–860, 1997.

[114] C.F. Olson, "Efficient Pose Clustering Using a Randomized Algorithm," *Int. Journal of Computer Vision,* vol. 23, no. 2, pp. 131–147, 1997.

[115] C.F. Olson, "Improving the Generalized Hough Transform Through Imperfect Grouping," *Image and Vision Computing,* vol. 16, pp. 627–634, 1998.

[116] C.H. Papadimitriou, *Computational Complexity*, Addison Wesley, 1994.

[117] M. Petrou, M. Mirmehdi, and M. Coors, "Multilevel Probabilistic Relaxation," Proc. of the Eighth British Machine Vision Conference, pp. 60–69, 1997.

[118] T.Q. Phong, R. Horaud, A. Yassine, and D.T. Pham, "Object Pose from2-D to 3-D Point and Line Correspondences," *Int. Journal of Computer Vision,* vol. 15, no. 3, pp. 225–243, 1995.

[119] M. Pilu, "A Direct Method for Stereo Correspondence based on Singular Value Decomposition," *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, Puerto Rico, pp. 261–266, June 1997.

[120] J.D. Pinter, "Continuous Global Optimization Software: A Brief Review," *Optima*, vol. 52, pp. 1-8, 1996.

[121] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed., Cambridge: Cambridge University Press, 1992.

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

[123] L. Quan and Z. Lan, "Linear N-Point Camera Pose Determination," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 8, pp. 774–780, 1999.

[124] A. Rangarajan and E. Mjolsness, "A Lagrangian Relaxation Network for Graph Matching," *Proc. IEEE Int. Conf. on Neural Networks,* vol. 7, pp. 4629–4634, 1994.

[125] S. Richter and R. de Carlo, "Continuation Methods: Theory and Applications," *IEEE Trans. on Circuits and Systems*, vol. 30, no. 6, pp. 347–352, 1983.

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

[127] K. Rose, "Deterministic Annealing for Clustering, Compression, Classification, Regression, and Related Optimization Problems," *Proc. of the IEEE*, vol. 86, no.11, pp. 2210-2239, 1998.

[128] P.J. Rousseeuw and A.M. Leroy, *Robust Regression and Outlier Detection,* John Wiley and Sons, New York, 1987.

[129] S. Sarkar and K.L. Boyer, "Perceptual Organization in Computer Vision: A Review and a Proposal for a Classificatory Structure," *IEEE Trans. on Systems, Man and Cybernetics,* vol. 23, no. 2, pp. 382–399, 1993.

[130] 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.

[131] C.Schmid, R.Mohr, and C.Bauckhage, "Comparing and Evaluating Interest Points," *Int. Conf. on Computer Vision*, Bombay, pp. 230-235, January 1998.

[132] G. Scott and C. Longuet-Higgins, "An Algorithm for Associating the Features of Two Images," *Proc. Royal Society of London B*, pp. 21–26, 1991.

[133] L.S. Shapiro and J. Brady, "Feature-Based Correspondence: An Eigenvector Approach," *Image and Vision Computing*, pp. 283–288, June 1992.

[134] T.M. Silberberg, D.A. Harwood, and L.S. Davis, "An Iterative Hough Procedure For Three-Dimensional Object Recognition," *Pattern Recognition,* vol. 17, no. 6, pp. 621-629, 1984.

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

[136] S.M. Smith and J.M. Brady. "SUSAN - A New Approach to Low Level Image Processing," *Int. Journal of Computer Vision*, vol. 23, no. 1, pp. 45-78, May 1997.

[137] J.P.P. Starinka and Eric Backera, "Finding Point Correspondences Using Simulated Annealing," *Pattern Recognition*, vol. 28, no. 2, pp. 231-240, February 1995.

[138] G. Stockman, S. Kopstein, and S. Benett, "Matching Images to Models for Registration and Object Detection via Clustering," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 4, no. 3, pp. 229-242, 1982.

[139] G. Stockman and J.C. Esteva, 3-D Object Pose from Clustering with Multiple Views, *Pattern Recognition Letters,* vol. 3, pp. 279-286, 1985.

[140] G. Stockman, "Object Recognition and Localization via Pose Clustering," *Computer Vision, Graphics, and Image Processing,* vol. 40, no. 3, pp. 361–387, 1987.

[141] T.M. Strat and M.A. Fischler, "Context-Based Vision: Recognizing Objects Using Information from both 2D and 3D Imagery," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 10, pp. 1050–1065, October 1991.

[142] P. Suetens, P. Fua, and A.J. Hanson, "Computational Strategies for Object Recognition," ACM Computing Surveys, Vol. 24, No. 1, pp. 5–61, March 1992.

[143] R. Talluri, and J.K. Aggarwal, "Mobile Robot Self-Location using Model-Image Feature Correspondence," *IEEE Trans. on Robotics and Automation*, vol. 12, no. 1, pp. 63-77, Feb. 1996.

[144] D.W. Thompson and J.L. Mundy, "Three-Dimensional Model Matching From an Unconstrained Viewpoint," *Proc. IEEE Int. Conf. on Robotics and Automation,* pp. 208–220, 1987.

[145] C. Tomasi and T. Kanade, "Detection and Tracking of Point Features," Carnegie Mellon University Technical Report CMU-CS-91-132, April 1991.

[146] P.H.S. Torr and D.W. Murray, "Outlier Detection and Motion Segmentation," *Proc. SPIE, Sensor Fusion VI,* vol. 2059, pp. 432–443, 1993.

[147] B. Triggs, "Camera Pose and Calibration from 4 or 5 Known 3D Points," *Proc. Int. Conf. Computer Vision,* pp. 278–284, 1999.

[148] R.Y. Tsai, "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses," *IEEE Journal of Robotics and Automation,* vol. 3, no. 4, pp. 323–344, 1987.

[149] P.W.M. Tsang, and A.T.S. Au, "A Genetic Algorithm for Projective Invariant Object Recognition," *Proc. IEEE TENCON, Digital Signal Processing Applications*, vol. 1, pp. 58–63, Nov. 1996.

[150] M. Turk and A. Pentland, "Eigenfaces for Recognition," *Cognitive Neuroscience*, vol. 13, no. 1, pp. 71– 96, 1991.

[151] S. Ullman, *The Interpretation of Visual Motion*, Cambridge, MA, MIT Press, 1979.

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

[153] 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.

[154] J.-C. Yuan, "A General Photogrammetric Method for Determining Object Position and Orientation," *IEEE Trans. Robotics and Automation,* vol. 5, no. 2, pp. 129-142, 1989.

[155] Y. Zheng and D. Doermann, "Robust Point Matching for Non-Rigid Shapes: A Relaxation Labeling Approach," Technical report CS-TR-4633 and UMIACS-TR-2004-75, Institute for Advanced Computer Studies, University of Maryland, 2004.