LOCAL SEARCH ALGORITHMS FOR GEOMETRIC OBJECT RECOGNITION: OPTIMAL CORRESPONDENCE AND POSE

J. Ross Beveridge

Tech. Report. CS 93-??

ACKNOWLEDGMENTS

There is little, besides programing, that I enjoy doing alone. I find ideas are best shared and developed with others, and therefore the genesis of this thesis lies in discussions with more people than I dare list; to all who have played a part, I say thank you.

I owe a special debt to my thesis committee. Ed Riseman's and Al Hanson's guidance and patience have given me the skills to do research. Their driving commitment to computer vision, their curiosity about computer vision, and their open minds are sources of inspiration. I am indebted to Robbie Moll, for he introduced me to local search. Finally, Don Geman's command of probability theory, and his refreshing common sense, make his advice invaluable.

Rich Weiss helped to frame the basic research problem addressed in this thesis, and I am in his debt. Teddy Kumar deserves special thanks, for without his 3D pose algorithm, and his patience in explaining it to me, the extensions to full-perspective matching would have been impossible. Through countless joyful discussions with Bruce Draper and Bob Collins I have defended, refined, and changed my ideas about computer vision. Finally, I wish to acknowledge Scott Anderson, Brian Burns, John Brolio, Martin Herbordt, Manmatha, John Oliensis, Harpreet Sawhney, and Lance Williams, for they have all helped me.

It is with gratitude that I thank the agencies whose financial support made this work possible. This work has been supported in part by the Advanced Research Projects Agency(via TACOM) under contract DAAE07-91-C-RO35, and contract DACA76-92-C-0041 (via U.S. Army Topographic Engineering Center). This work has also been supported by the National Science Foundation under grant number CDA-8922572.

My wife, Adele Howe, and I entered graduate school together, and we have supported each other through the creation of two theses: hers and mine. Adele's insight and unfaltering support has meant more to me than I can say. My life personally, and as a researcher, is immeasurably richer for the benefit of her companionship.

ABSTRACT

LOCAL SEARCH ALGORITHMS FOR
GEOMETRIC OBJECT RECOGNITION:
OPTIMAL CORRESPONDENCE AND POSE

May 1993

J. Ross Beveridge

B.S., University of California, San Diego

M.S., University of Massachusetts

Ph.D., University of Massachusetts

Directed by: Professor Edward M. Riseman

Recognizing an object by its shape is a fundamental problem in computer vision, and typically involves finding a discrete correspondence between object model and image features as well as the pose - position and orientation - of the camera relative to the object. This thesis presents new algorithms for finding the optimal correspondence and pose of a rigid 3D object. They utilize new techniques for evaluating geometric matches and for searching the combinatorial space of possible matches. An efficient closed-form technique for computing pose under weak-perspective (four parameter 2D affine) is presented, and an iterative non-linear 3D pose algorithm is used to support matching under full 3D perspective.

A match error ranks matches by summing a *fit error*, which measures the quality of the spatial fit between corresponding line segments forming an object model and line segments extracted from an image, and an *omission error*, which penalizes matches which leave portions of the model omitted or unmatched. Inclusion of omission is crucial to success when matching to corrupted and partial image data.

New optimal matching algorithms use a form of combinatorial optimization called local search, which relies on iterative improvement and random sampling to probabilistically find globally optimal matches. A novel variant has been developed, *subset-convergent* local search finds optimal matches with high probability on problems known to be difficult for other techniques. Specifically, it does well on a test suite of highly fragmented and cluttered data, symmetric object models, and multiple model instances. Problem search spaces grows exponentially in the number of potentially paired features n, yet empirical performance suggests computation is bounded by n^2.

Using the 3D pose algorithm during matching, local search solves problems involving significant amounts of 3D perspective. No previous work on geometric matching has generalized in this way. Our hybrid algorithm combines the closed-form weak-perspective pose and iterative 3D pose algorithms to efficiently solve matching problems involving perspective. For robot navigation, this algorithm recognizes 3D landmarks, and thereby permits a mobile robot to successfully update its estimated pose relative to these landmarks.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

The shape of a three-dimensional (3D) object is frequently the most distinctive attribute used to identify that object in an image. For rigid objects, recognition[1] by shape involves two interrelated subproblems. The first is to find a discrete mapping between features of a geometric object model and the corresponding features in a two-dimensional (2D) image. The second is to find the geometric transformation between the object model and the camera such that projected object model features fit the corresponding image features. The first is the *correspondence problem* and the second the *pose problem*. The process of solving for both the correspondence and pose is called 'geometric matching', or for brevity in this thesis, simply 'matching'.

No single, general, robust and computationally tractable matching algorithm exists. Although the problem has been studied for at least 30 years [95], all the commonly used algorithms have limitations. One failing of many algorithms is an inability to deal effectively with cluttered and imperfect image features. A second failing is an inability to handle 3D perspective. The vast majority of the work on recognition has dealt with essentially 2D problems often analogous to recognizing flat objects on a table.

To overcome some of these failings this thesis introduces a new family of matching algorithms based upon local search. Local search [64, 76, 92] is known within the combinatorial optimization literature as an effective means of finding near optimal solutions to some difficult combinatorial optimization problems. Little use has been made of local search in computer vision. Other non-linear optimization techniques have been used for a variety of tasks. For example, stochastic relaxation has been used for image restoration [39] and boundary detection [38] while Hopfield networks have been used for building recognition [85].

In this thesis, for the first time, local search is adapted to geometric matching. The result is a family of algorithms which probabilistically find globally optimal matches. As will be shown, these algorithms excel at finding matches in highly cluttered and imperfect data. They also perform matching subject to full 3D perspective. These two strengths in particular set them apart from algorithms which have come before.

The remainder of this chapter motivates and introduces local search as a means of solving geometric matching problems. Section 1.1 describes the basics of local search in general and local search matching in particular. Section 1.2 outlines some of the broad research objectives which have motivated and driven the research. Section 1.3 discusses mathematical models

[1] The term 'recognition' here refers to finding a particular object expected to be present in an image. This is consistent with prior uses of the term [95, 80, 48], but should not be confused with the less constrained problem of selecting which of many different objects are visible in an image.

of how 3D objects appear in 2D images. Two key terms, *weak-perspective matching* and *full-perspective matching*, are defined here. Section 1.4 provides three concrete examples of matching problems solved with the algorithms developed in this thesis. Section 1.5 reviews the major contributions of the thesis and finally Section 1.6 provides a guide to the remainder of the thesis.

1.1 Local Search Matching

The simplicity of local search [92] is enticing. Local search exploits a synergy between iterative improvement and random sampling. Iterative improvement is used to repeatedly improve an initial solution, a match of a model in this work, until it is locally optimal. To increase the probability of obtaining a near optimal solution, multiple trials are run from randomly selected initial solutions, and the best solution from the set is retained.

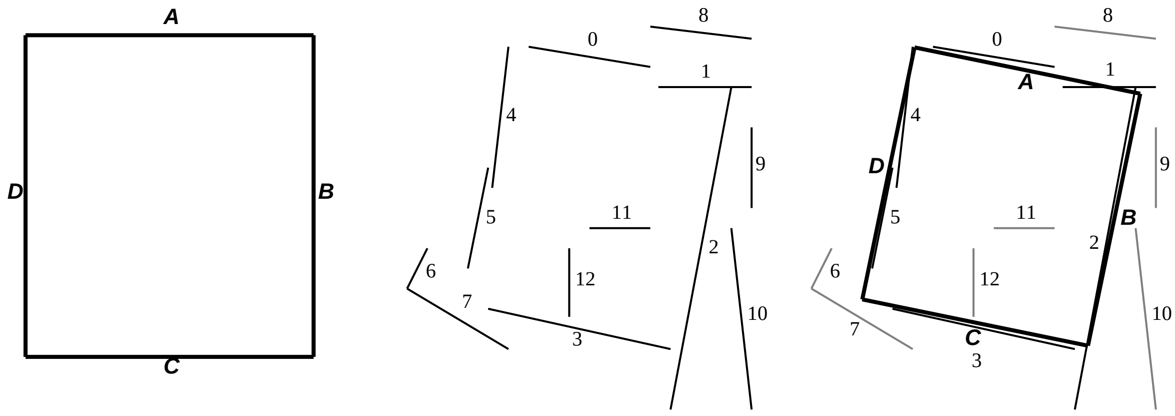
Local optimality means a match is comparable to or better than all its local neighbors. The neighborhood itself is a domain-specific construct, and designing good neighborhoods is an important part of applying local search to new problems. Local search shares its use of iterative improvement with other well known techniques, such as hill-climbing and gradient descent. Like hill-climbing, local search is typically used for discrete state spaces. Gradient descent, on the other hand, assumes a continuous space and a differentiable objective function.

The origins of local search as a technique for solving difficult combinatorial optimization problems are commonly traced to the work of Kernighan [64] and Lin [75, 76]. In these papers, Kernighan and Lin demonstrated that fairly simple iterative improvement algorithms could find near optimal solutions to such problems as traveling salesperson (TSP) and graph partitioning. Local search is one of the earliest and simplest of the general, heuristic [2] non-linear optimization techniques, and its variants remain the best way known of solving many of the well studied combinatorial optimization problems [60, 61].

Figure 1.1 illustrates geometric matching as a problem and how local search is used to solve it. The object model shown is a rectangle comprised of four straight line segments labelled A through D. Image features consisting of straight line segments are shown and numbered. These represent imperfect data such as might be produced by a straight line extraction algorithm. Matching involves finding a discrete *correspondence* between model and image segments as well as the *pose* of the model relative to the data. In this example, the pose is the 2D position, orientation and size of the rectangle relative to the image segments. More generally in this thesis, the pose will be the 3D position and orientation of a rigid object relative to a camera.

The best match in Figure 1.1 illustrates the rectangle model fit to corresponding image segments using a least-squares error criterion developed as part of this thesis. Placing the object model in this best-fit pose relative to the corresponding image data is the first step in evaluating match quality. Two fundamental and intuitive criterion go into evaluating the quality of a match. First, in a good match, model features *fit* the corresponding image data. Second, good matches do not *omit* significant portions of the model. The match error function formalizes and quantifies fit and omission. As should seem intuitive from

[2] Heuristic here means the method is not guaranteed to find the best, globally optimal solution



Rectangle Model *Imperfect Data* *Best Match*

		A				B				C			D			
Row	**Error**	0	1	8	11	2	9	10	12	3	7	11	4	5	6	12
1	3.52	●			●	●		●		●	●					●
2	1.52	●				●		●		●	●					●
3	0.81	●						●		●	●					●
4	0.60	●						●		●	●					
5	0.45	●						●		●	●		●			
6	0.38	●					●	●		●	●		●			
7	0.34	●					●	●		●			●			
8	0.31	●					●	●		●			●	●		
9	0.30	●	●				●	●		●			●	●		
10	0.28	●	●				●			●			●	●		
11	0.20	●	●			●	●			●			●	●		
12	0.10	●	●			●				●			●	●		

Single Exececution of Local Search

Figure 1.1 Illustrating local search matching. A model, imperfect data, and an optimal match are shown. In addition, successively better correspondence mappings show how local search might move from a poor match to the one which is optimal.

inspection, the match shown minimizes this match error. Often in this thesis it will be possible to determine the globally optimal match by inspection, and algorithms will be judged according to their ability to find these globally optimal matches.

The table in Figure 1.1 illustrates local search finding the best match. A black circle filling a square indicates that a specific pair of model-image segments correspond as part of the match. Successive rows indicate successively better matches. The match error is displayed on the left. Search is initiated starting with a fairly poor match and culminates in row 12 with the best match. Although the initial match is poor, local search determines a modest set of simple perturbations which transform this poor match into the best match.

The perturbations leading to better matches are determined by searching a neighborhood of possible changes to the current match. Neighborhood exploration is an explicit generate-and-test procedure in which the match error for each neighbor is explicitly computed. In this example, the neighborhood consists of all matches obtained by adding or removing a single pair of candidate model-image pairs. At each step in the search the match error for all neighbors (in this case 15) is computed and the neighbor with the lowest match error becomes the new best-match.

A single execution of local search may not arrive at the globally optimal match. There is a probability that local search initiated from a randomly selected initial match will arrive at the globally optimal match and changing the design of the neighborhood can dramatically alter this probability. This thesis presents neighborhoods for which this probability is high enough to enable the use of local search to solve geometric matching problems. However, even with the best neighborhoods, random sampling is still an important tool for further increasing the probability of finding the globally optimal match. The idea is common to many probabilistic algorithms: run many trials and return the best solution.

For a wide range of practical matching problems, neighborhoods presented in this thesis find the globally optimal match with probability better than 0.1. With such neighborhoods, finding the globally optimal match with 99% confidence requires no more than 50 independent trials of local search. Each trial searches a vanishingly small portion of the combinatorial correspondence space, and hence the procedure as a whole is computationally tractable.

There is a fundamental difference between geometric matching and most other combinatorial optimization problems. In problems such as graph partitioning and TSP, changes in the objective function for neighboring solutions are localized. For instance, with the TSP the change in the cost of a tour resulting from the swapping of two cities is localized to the links in the tour directly involved in the swap. Costs for links between cities not involved in the swap do not change.

In contrast, for geometric matching, pairs of segments are interrelated through the best-fit pose of the model. Adding or removing a single pair of model-image segments from a match changes the pose and this in turn changes the fit and omission error for *all* pairs of matching segments. This coupling of evaluation through the best-fit pose of the model complicates the evaluation of potential matches and generates subtle interactions. Geometric matching should not be confused with simpler 'matching' problems which lack such global interdependence.

1.2 Objectives

The impetus for local search as an approach to matching arises in part out of the following research objectives:

1) Avoid domain-specific and object-specific assumptions.

2) Deliver low-order polynomial, average-case computational complexity.

3) Separate the definition of a good match from the choice of matching algorithm.

4) Make no unrealistic assumptions about model form or data quality.

5) Solve harder problems with more computation, not additional code.

Some of these are more obvious then others. All are in large part satisfied by the algorithms presented in this thesis.

The first objective distinguishes this work from domain-specific or even object specific work. Algorithms fashioned from scratch to exploit specific object or domain knowledge will typically outperform more general algorithms. If the goal is to find rectangular objects, and only rectangular objects, then specialized algorithms can be designed for this task. However, the limitations inherent in such specialization need hardly be stated. There is one important correlate to this goal: the best algorithms ought to have both a general form and clearly identifiable mechanisms for incorporating additional domain constraints. For local search matching, a means of incorporating domain constraints is through the modification of the objective function, and this is described in Section 3.6.

The second goal, polynomial average-case complexity, is a practical necessity. General, exponential algorithms already exist for geometric recognition. They are of little practical use! Implicit in this goal is also the assumption that average case performance is a good indicator of how the algorithm will perform in practice. Hence, it is important that algorithms do not run away in an uncontrollable fashion, even when they encounter unusually difficult problems.

The third goal, that of separating the criterion for success from the design of an individual algorithm, is less common but equally important. Formalizing matching as combinatorial optimization achieves this goal. A match error defines the goal of the process to be the match with the lowest error. Failure of the system to perform as desired may be due to: 1) the match error improperly favoring an undesired result, or 2) the search algorithm failing to find the optimal match. The means of remedying these two situations are completely different, and hence the ability to distinguish between them is critical. A formal definition of optimality also facilitates comparisons between alternative search algorithms.

The importance of handling realistic data and models cannot be overemphasized. As will shortly be illustrated with examples, image data is often fragmented, accreted (over grouped), cluttered, and partial. Failure to perform gracefully under all these conditions limits usefulness. Realistic assumptions regarding models are equally important. Polygonal models are limiting in at least two ways. First, they assume that in a partially modeled world all interesting objects have clearly delineated, visible, closed boundaries. Second, they essentially force an algorithm to ignore surface markings. Other limiting assumptions to be avoided include insisting that only one instance of a model be present or that models be asymmetric.

The final objective is that harder problems should require more computation but not new code. It is less common to see this objective stated , although it is clearly desirable. It is relatively easy to see that local search meets this objective. A single trial of local search improves upon an initial match until it is locally optimal. Starting from a randomly selected initial match, there is some probability of finding the globally optimal match. In a very practical sense this probability is a measure of problem difficulty. Finding the globally optimal match with the same level of confidence requires more trials for harder problems. It does not, however, require any modifications to the underlying matching software.

Understanding how to determine the number of trials to run is essential to understanding the practical use of the algorithms developed in this thesis. The approach advocated in this thesis is to identify a set of sample matching problems and to run local search many times on these problems. For each problem, the success/failure ratio is recorded, where success means local search found the optimal match. Because the independent trials of local search represent a binomial random process, it is possible to use these ratios to estimate the trials required to obtain success with say 95% or 99% confidence for each problem. The estimates for individual problems may be used to derive a conservative estimate of the number of trials required to solve problems from the sampled problem domain. This process is more fully described in Section 5.3.1.

1.3 Imaging

Where a set of points on a rigid 3D object model appear in an image depends upon two things. It depends upon the pose of the object relative to the camera. It also depends upon the projective mapping between 3D points in camera coordinates and 2D points on the camera image plane. Imaging is the combined effect of these two factors. The common 3D perspective imaging model assumes that arbitrarily placed rigid 3D objects are viewed through a pin-hole camera. Mathematically this is accomplished by rotating and translating 3D points on the object model from object coordinates into camera coordinates and then perspective projecting these 3D points onto the 2D image plane. If $\vec{P}_o$ is a 3D point on an object model in object coordinates and $\vec{p}_c$ is the 2D image of this point under full-perspective, then their relationship is defined as:

$$\vec{p}_c = \begin{vmatrix} x/z \\ y/z \end{vmatrix} \qquad \text{where} \qquad \begin{vmatrix} x \\ y \\ z \end{vmatrix} = \vec{P}_c = R\vec{P}_o + \vec{T}. \tag{1.1}$$

The pose of the object relative to the camera is expressed as a rotation R and translation T. Visibility checks, such as hidden line and hidden surface removal, are used to avoid imaging obstructed portions of an object. This full-perspective imaging is the basis of most current 3D computer graphics techniques [36].

If all model matching techniques assumed full-perspective, then there would be no need to say more about imaging. However, this is not the case. It appears that with the possible exception of Lowe [82] [3], the work presented in this thesis is the first to perform geometrically accurate matching under full-perspective. This is in part because this thesis takes advantage of the work of Kumar [71, 69] on computing 3D object pose under full perspective. All other previous work has in one manner or another simplified the problem by making restrictive assumptions about imaging. To understand the work presented here, some of which also makes restrictive assumptions about perspective, as well as the relationship between this work and that which has come before, it is necessary to describe some of the more common ways of restricting the imaging process.

One restricted form of imaging is obtained by replacing perspective projection with something called scaled orthograhpic projection, which is defined by replacing equation 1.1 with:

$$\vec{p}_c = s \begin{vmatrix} x \\ y \end{vmatrix}. \tag{1.2}$$

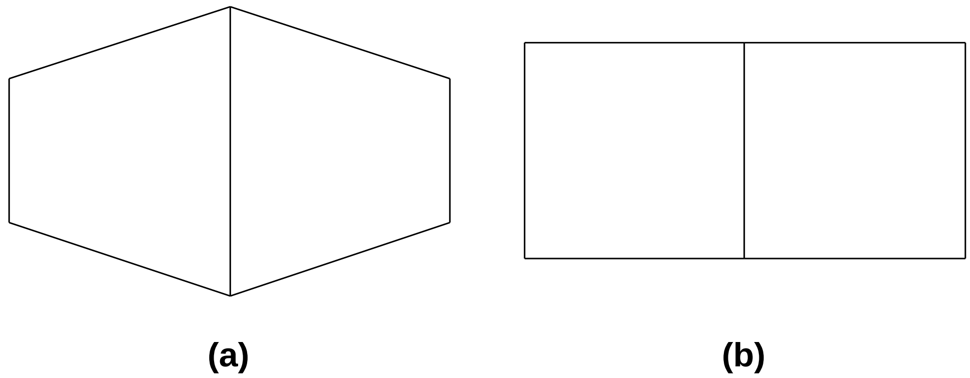


Figure 1.2 A full-perspective and weak-perspective-3D view of a cube. a) full-perspective and b) weak-perspective-3D. Perspective makes it possible to see the cube as 3 dimensional while with weak-perspective it looks flat and 2 dimensional.

Using scaled orthographic projection yields what will be called weak-perspective-3D. The '3D' is tagged onto the end to distinguish it from another weakened form of perspective imaging defined below. What is most characteristic about weak-perspective-3D imaging is that it does not induce vanishing points. What this means in practice is that it is an adequate approximation to full-perspective imaging for shallow objects: objects with little depth relative to their distance to the camera. However, it will increasingly distort objects for which some points on the object are significantly closer to the camera than others. The striking difference between a full-perspective view and a weak-perspective-3D view of a cube is illustrated in Figure 1.2.

Both full-perspective and weak-perspective-3D imaging map directly from the 3D object model to the 2D image plane. Other forms of imaging divide imaging into two steps. In the first step, the 3D object model is perspective projected into the image plane based upon a particular pose of the object relative to the camera. In the second step, this projection

[3] In this work Lowe solves limited correspondence problems in the context of motion tracking. However, he does so while fully accounting for 3D perspective.

becomes a 2D template subjected to 2D affine (similarity) transformations in the image plane. These 2D affine transformations partially account for changes in appearance associated with relative changes in object pose. In the context of matching, the first step is usually done only once prior to matching. Sometimes the 2D model may be generated directly by hand or extracted from an image. Figure 1.3 summarizes different types of imaging.

When a 2D similarity transform - rotation, translation and scaling - is used in place of a general affine transform, the result is a different form of weakened perspective imaging. This type of imaging shall here be called weak-perspective-2D. Weak-perspective-2D shares some attributes with weak-perspective-3D. It behaves similarly for planar objects viewed at roughly right angles to the camera. However, these are fundamentally different forms of imaging and should not be confused.

Weak-perspective-3D and weak-perspective-2D imaging have different strengths and weaknesses. The strengths of weak-perspective-3D imaging include the ability to image an object from any arbitrary viewpoint. In contrast, assuming a 2D projection generated from a particular initial viewpoint, weak-perspective-2D can only generate a subset of all the possible views of the object. In this respect, weak-perspective-3D is more general than weak-perspective-2D.

However, weak-perspective-2D is superior to weak-perspective-3D in the following respect. When viewing an object of modest depth relative to its distance from the camera, weak-perspective-3D cannot accurately express the appearance of the object as viewed through a pin-hole camera from *any* viewpoint. In contrast, for at least a subset of views related to an initial perspective projection of the 3D object, weak-perspective-2D will accurately express the appearance of the object.

This difference may be illustrated using Figure 1.2. There are a set of viewpoints from which the 2D projection in Figure 1.2a can be transformed via a 2D similarity transform so as to exactly match the actual 2D appearance of the cube as viewed through a pin-hole camera. For example, if the cube were moved away from the camera its image would shrink, and this could be accommodated by changing the scale factor of the 2D similarity transformation. In contrast, there is *no* viewpoint from which the cube would appear as pictured in Figure 1.2b.

There is a fourth even more restrictive form of imaging worthy of mention because it is one of the most commonly studied and easiest under which to perform geometric matching. This is 2D-rigid imaging, and it is similar to weak-perspective-2D imaging except that the 2D similarity transform is replaced with a 2D rigid - rotation and translation only - transform.

To provide a bit of intuition for each form of imaging, the following describes the physical circumstances which might give rise to each.

2D-rigid - Flat objects on a table perpendicular to the camera. The object can be translated and rotated on the table. The distance from the camera to the table is known.

Weak-perspective-2D - Any 3D object, but viewed from a restricted set of viewpoints defined relative to some nominal viewpoint. For example, a camera's precise orientation is not critical, nor is its distance from the object important. However, changes in relative orientation to the object can violate the weak-perspective-2D imaging model. Chapter 8 will clarify these viewpoint restrictions.

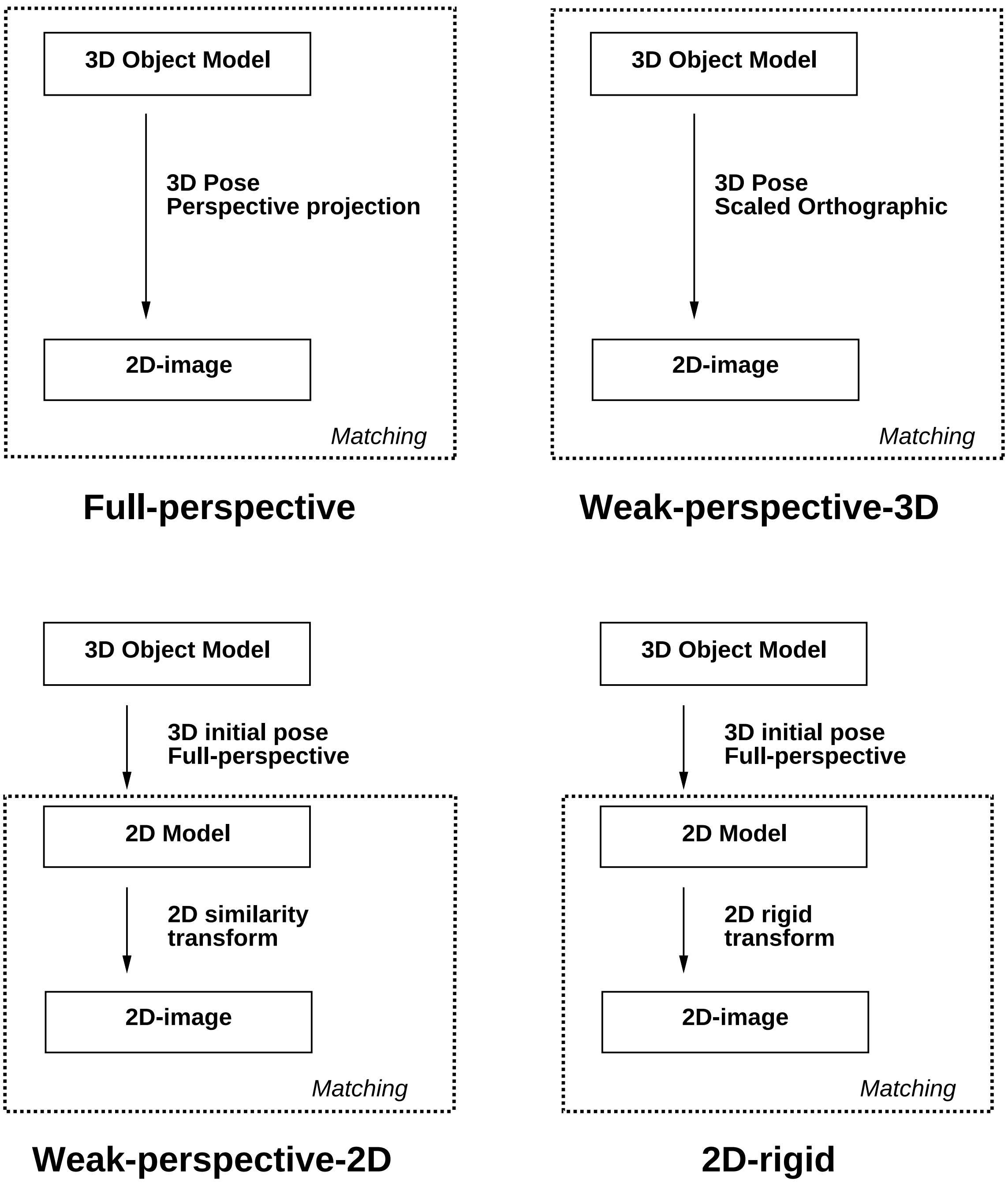


Figure 1.3 Four basic forms of imaging. The relationship between 3D model and the 2D image of the model is indicated in each case. The two step nature of weak-perspective-2D and 2D-rigid imaging are further indicated. The enclosing dashed boxes indicate the part of the imaging process dealt with during matching.

Weak-perspective-3D - Any 3D object which is shallow in depth compared to its distance from the camera. It can be viewed from any arbitrary viewpoint. However, to the extent that points on the object are not all at a constant depth relative to the camera, distortions will be induced.

Full-perspective - Any 3D object viewed from any arbitrary viewpoint. Full-perspective is an excellent first-order approximation for a standard camera. It neglects second-order effects such as radial distortion.

In this thesis local search matching will be performed assuming both weak-perspective-2D and full-perspective imaging. Matching under weak-perspective-2D imaging will in future be called *weak-perspective matching*. Matching under full-perspective imaging will be called *full-perspective matching*. These terms are well-defined within the context of this thesis, but it should be noted that the term weak-perspective matching is potentially confusing and may be used by others to refer to matching under weak-perspective-3D imaging. Section 2.4.8 of the literature review lists imaging forms assumed by previous works on geometric matching.

1.4 Examples

Each of the matches shown in this series of examples has been obtained with the algorithms developed in this thesis. The first example illustrates highly imperfect image data. The second example illustrates the role of global geometric consistency in disambiguating potentially confusing local structure. It also illustrates many-to-many mappings between model and image features. The third example illustrates the need to account for full-perspective.

The first example, Figure 1.4, is drawn from the aerial photo-interpretation domain. The specific problem is that of identifying a previously modeled building in the approximate area of the image in which it is expected to appear. Figure 1.4a shows a part of a larger image. Figure 1.4b shows line segment features extracted from this image using the Burns algorithm [20]. Figure 1.4c shows a relatively simple object model. The placement of this model represents an initial estimate of the object's position, orientation, and size. Figure 1.4d shows those data features determined to match the model. Segments shown in light grey were considered potential matches based upon the initial placement of the model as shown in Figure 1.4c, and forms the basis of the combinatorial search space in which the optimal match was found.

The relatively poor quality of the data in Figure 1.4 is indicative of what current feature extraction algorithms produce under these conditions, and makes this a challenging problem. The optimal match is pieced together from multiple fragments along each side of the model. It was also necessary to disregard the considerable 'clutter' present around the true match. These clutter segments mimic the local rectilinear structure of the model and greatly exacerbate the combinatorics of the correspondence problem.

Another application of matching arises when tracking a moving object. An example is presented in Figures 1.5 and 1.6. The car is moving directly toward the camera, and under these conditions it is possible to use a subset of the line segments extracted from the first image as a model to be found in the second two images. The subset of segments selected by

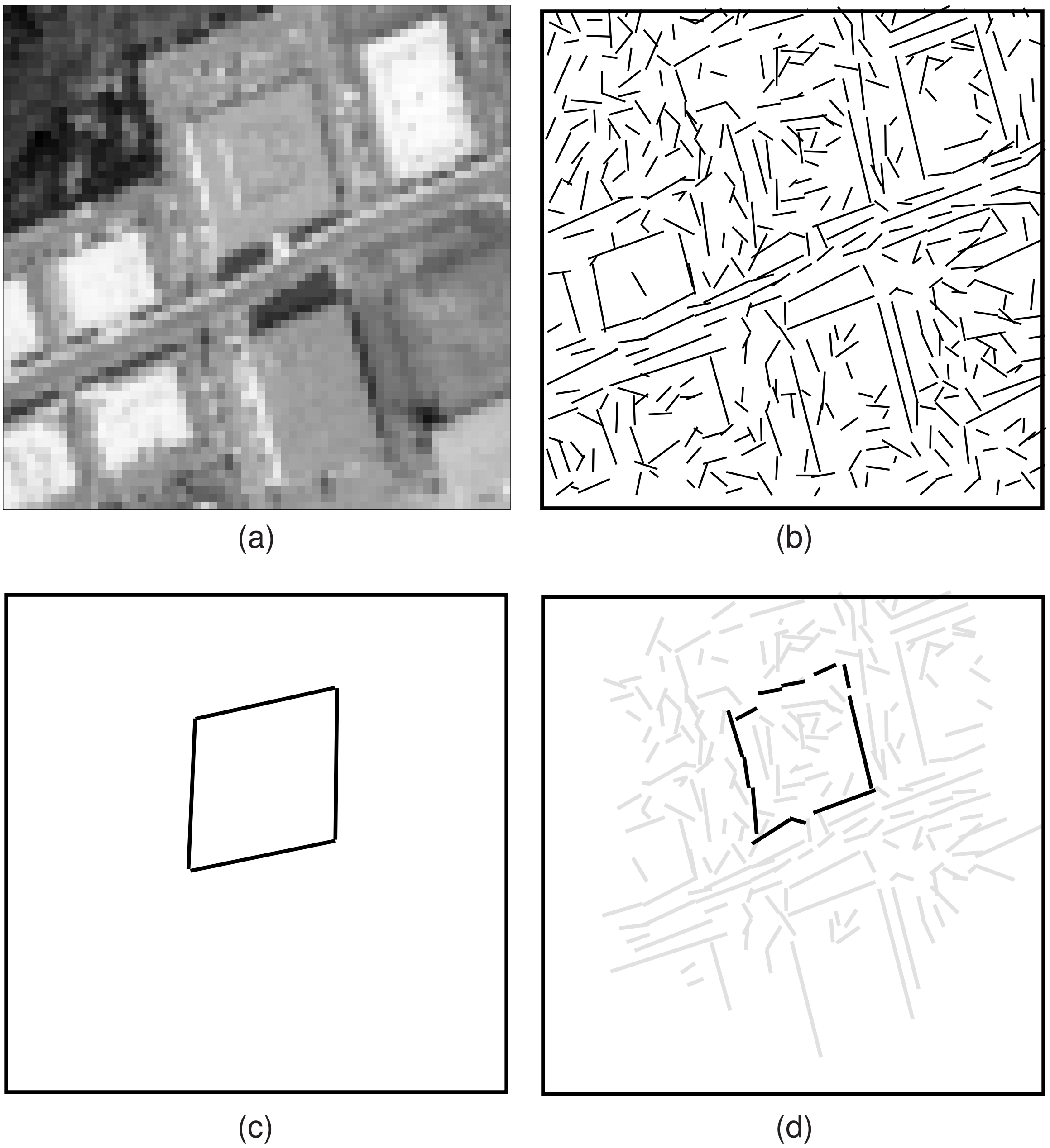


Figure 1.4 A matching example drawn from aerial image photo-interpretation: a) a small section of a larger aerial image, b) line segment features, c) a simple geometric object model denoting a building, d) the black data segments as a set match the building model while the grey segments do not.

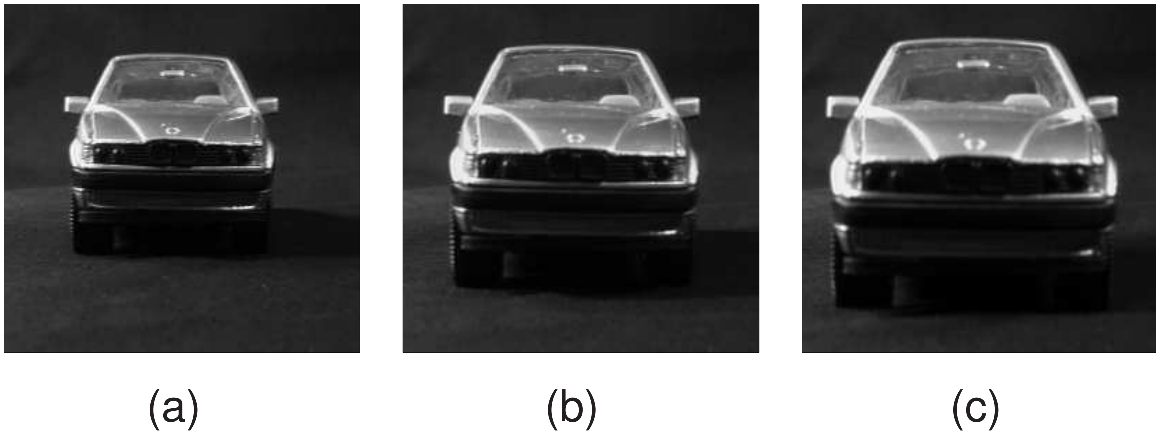


Figure 1.5 A sequence of moving car images. The car is moving toward the camera and hence it grows in size. The perspective remains essentially unchanged.

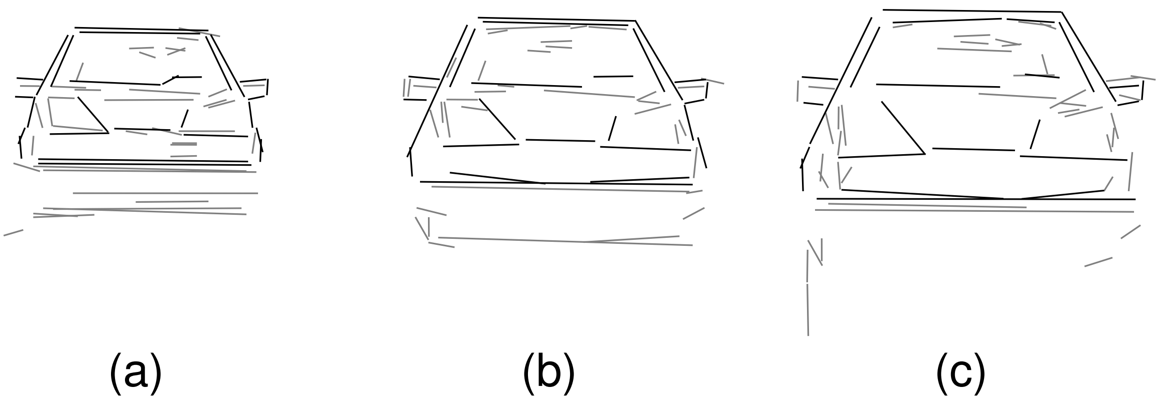


Figure 1.6 Tracking can be cast as a matching problem. Segments selected from the first image (a) are matched to features extracted from images (b) and (c). Black segments represent the model in (a) and the segments matched to it in (b) and (c).

hand to serve as a model of the car are shown in black in Figure 1.6a. The grey segments were not considered worth tracking. This model is then matched to the segments shown in Figure 1.6b and Figure 1.6c. Black segments match the model while grey segments do not. In this example, as in the previous one, the model is rotated, translated and scaled in the image plane so as to best-fit the corresponding data. Hence, these are both weak-perspective matching problems.

This tracking example illustrates several sources of potential difficulty. Observe that pairs of closely spaced lines border the windshield on the top and sides. The outer segments could locally match the inner segments as well as the outer segments, and the matching algorithm must overcome this local ambiguity. Fitting the object model as a whole effectively accomplishes this task. Although local search matching performs nearly flawlessly on this example, the reader should note there is a slight error in Figure 1.6c. The top of the right hand mirror is matched to a segment below the correct one. This slight mismatch is due to a slight distortion in the appearance of the model relative to what can be compensated for using only rotation, translation and scaling in the image plane. The match shown is optimal as measured by the match error function.

This match also provides an illustration of many-to-many mappings between object features and image features. Both this example and the last show the importance of matching one model line segment to one or more data fragments. This example also includes a case where two model segments properly match a single data segment. The outer left hand side of the car in the model consists of three adjoining segments. In the next image, Figure 1.6b, these three segments are extracted as a single overgrouped segment.

A third application of geometric object recognition arises in the context of mobile vehicle navigation. A vehicle in a partially modeled environment should be able to determine its position by recognizing known landmarks. Figure 1.7a shows an image acquired by a mobile robot in a hallway. Figure 1.7b shows data line segments extracted from the image. Figure 1.7c shows a projection of a 3D partial model of the hallway as it would appear from a position about 10 feet behind the robot's current position and with the robot looking directly down the hallway. Figure 1.7d shows the image segments found to match this 3D landmark. As in Figure 1.4, segments considered candidates but not included in the optimal match are shown in light grey.

In the first two examples, the matching problems were essentially 2D. To be precise, the models themselves consisted of 2D line segments which were rotated, translated and scaled in the image plane. However, in the last example from the hallway, the model consisted of 3D line segments which were rotated and translated in 3D, and then projected into the image plane using perspective projection. As will be exemplified in Chapter 8, in domains such as hallways, 3D perspective plays a significant role in how an object's appearance changes with small changes in camera position.

1.5 Contributions

As the examples just presented suggest, the local search algorithms presented in this thesis handle a wide variety of problems. As will be argued here, these algorithms overcome a variety of problems associated with previously proposed approaches to matching.

Previous work falls roughly into four categories: 1) Feature-focus [14], key-feature [80]

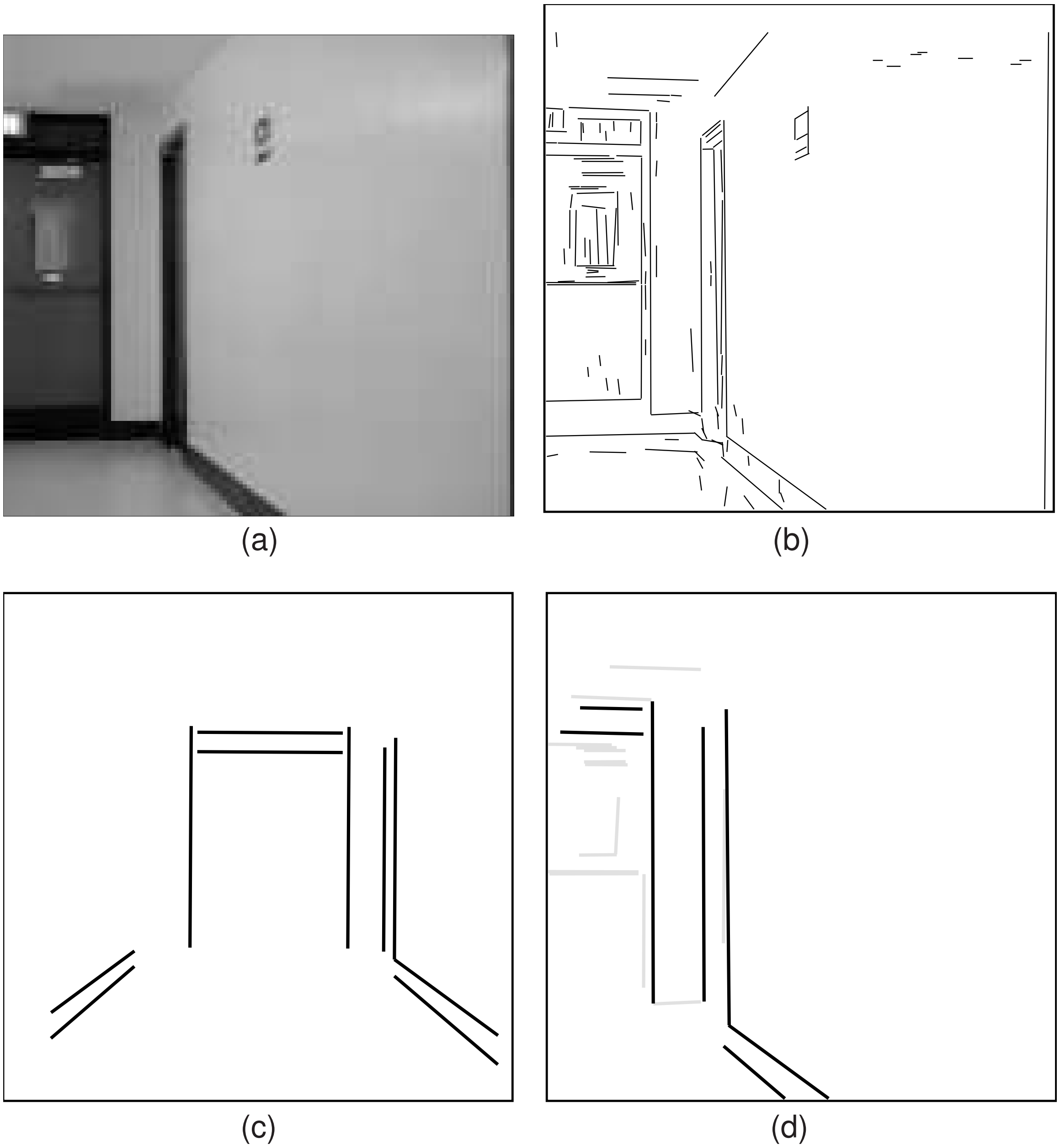


Figure 1.7 A landmark-based navigation example: a) an image obtained from a mobile robot, b) data line segments, c) a partial model of the hallway as seen from ten feet back and with the robot oriented directly toward the door at the end of the hall. d) Image features found to match landmark features. The match allows the true pose of the robot to be computed to within the accuracy of the pose algorithm: roughly 6 inches in this example.

and alignment [54] algorithms search for a distinctive feature indicating a match. 2) Generalized Hough transform [28, 5] and pose clustering [103] algorithms search for accumulated evidence of a match in the space of model-data transformations. 3) Geometric hashing [63, 72] and feature indexing [101] algorithms broaden the key-feature concept, using local clusters of features to predict both the identity and placement of an object. 4) Tree search [46, 48] and geometric consistency [4, 22] algorithms utilize geometric consistency constraints to prune a search tree. Chapter 2 provides a more comprehensive review of these four basic approaches to matching.

In at least four basic areas local search matching compares favorably to these previous approaches. First, local search matching handles imperfect image data better than these other approaches. Second, measured growth in required computation as a function of problem size for local search matching appears comparable or superior to predicted growth for these other approaches. Third, local search matching places fewer constraints upon allowable geometric configurations of models than do these other approaches. Finally, local search matching has extended naturally from weak-perspective to full-perspective matching. This is not true of any other approach reviewed in Chapter 2.

Expanding upon the issue of imperfect data, arguably the greatest problems are caused by fragmentation and accretion. To handle both of these conditions, an algorithm must support many-to-many mappings between object model and image features. However, most of the algorithms reviewed in Chapter 2 depend crucially upon an assumed one-to-one mapping. These algorithms are doomed when image data fails to meet this expectation. Some algorithms support mapping one model feature to many image features, and these are somewhat more robust. However, only the local search algorithms developed in this thesis fundamentally [4] support many-to-many mappings.

Empirical study of the average case computational complexity of local search matching suggests computation grows as roughly n^2, where n is the number of potentially matching features. This leads to a conjecture that the average case computational complexity of local search matching is order n^2. Of course, this conjecture is weak, since extrapolating from empirical performance over a finite range of problems is at best suggestive, and at worse false. Formal analysis would be better, but unfortunately formal analysis of local search algorithms is known to be difficult [62]. Therefore, this thesis offers the results of a systematic empirical study in the belief that it is better to present qualified empirical results than to say nothing at all.

The test suite used for the study includes problems with multiple model instances and highly symmetric models. The importance of this is twofold. First, both of these are common occurrences in various practical domains, and hence should be handled gracefully in order for an algorithm to be of practical use. Second, the class of geometric matching algorithms whose computational complexity has been most thoroughly studied are the tree search algorithms of Grimson [48], and while Grimson has analytically derived an average-case complexity bound of n^2 for restricted classes of problems, he has also shown tree search to be exponential if either multiple model instances or symmetric models are present. In this light, an n^2 average-case bound for local search matching would be better than that for any of the other general methods. It remains to be seen whether the conjectured n^2 bound for local search

[4] Some practical algorithms permit additional correspondences to be added in a post-processing step.

matching will be born out by further study.

The algorithms developed in this thesis assume very little about the geometric form of an object. This permits them to be used on a great range of different problems and goes a long way to insure that the approach is domain-independent. A model is simply *a collection of line segments.* Unlike the majority of the work reported in Chapter 2, there is no requirement that segments form closed contours. Hence, segments can denote surface markings as well as occluding contours. Unlike key-feature and geometric hashing approaches, the local search algorithms do not require key-features or feature indexing. Therefore, they are not dependent on either an automated means for selecting such features or, more fundamentally, upon such features existing in the first place. However, when key-features are reliably available they may be used to augment random sampling as a means of picking initial matches.

Finally, and perhaps most significantly, in the previous work there is an almost complete lack of matching under full-perspective. The algorithms developed in this thesis readily generalized from the commonly studied weak-perspective problems to that of matching under full-perspective. The same cannot be said for the other approaches reviewed in Chapter 2. Credit for this extension goes in part to Rakesh Kumar, for without the benefit of his work on full-perspective pose recovery [71, 69], this extension would not have been possible. In addition, the extension is made possible by the fact that the local search approach to matching developed in this thesis does not depend upon finding local collections of pose-distinctive or pose-consistent features. Virtually all previous approaches to matching are dependent upon the existence of such features, and this dependence is largely responsible for the failure of these other approaches to generalize to full-perspective matching. The existence of pose-distinctive local features under full-perspective becomes problematic and their identification combinatoric.

In contrast, local search matching has no such dependence. Instead, the dependence upon the geometric mapping between object and image features is encapsulated within the match error calculation. Computing the match error involves finding the optimal spatial fit between the model and corresponding (currently matched) image features. Under weak-perspective this involves solving for a best-fit 2D similarity transformation while under full-perspective it involves solving for the best-fit 3D pose of the object relative to the image.

Solving for the weak-perspective best-fit is less costly than solving for full-perspective best-fit: the first requires solution of a quadratic equation while the second involves iteration using a quasi-Newton method. Typically solving for full-perspective best-fit requires only a modest number of iterations, and hence the costs of computing weak-perspective versus full-perspective best-fit differ by a constant rather than an exponential factor. Consequently, the cost of weak-perspective matching versus full-perspective matching can be expected to differ by a constant factor. This is born out in practice, with full-perspective matching taking roughly ten times as long as weak-perspective matching.

It was discovered in the course of doing the work for this thesis that a judicious combination of weak-perspective and full-perspective fitting techniques leads to a hybrid form of matching algorithm that has significant advantages over either approach used alone. These algorithms use weak-perspective to rank alternatives during search, and full-perspective when actually adopting new matches. What this means geometrically is that the hybrid algorithm incrementally improves the correspondence mapping between 3D object line segments and 2D image line segments using the weak-perspective approximation for guidance. As it does this, the full-perspective fitting incrementally improves the 3D object pose estimates, and in so doing incrementally removes perspective distortion from the 2D projection of the 3D object. Results presented in this thesis show these hybrid algorithms solve problems involving full-perspective in amounts of time comparable to the simpler weak-perspective matching algorithms.

Several additional innovations and results presented in this thesis deserve special mention. As just stated, fitting 2D line segment models subject to rotation, translation and scaling is quite efficient, requiring the solution of a quadratic equation. Both the efficient fitting methods and the fit measure itself are original contributions of this thesis. The fit measure is the *integrated squared perpendicular distance* (ISPD) between the model line and corresponding image line segments. In two ways ISPD is superior to previously suggested measures [3]. First, perpendicular error is measured relative to the more stable *model line* rather than to the potentially fragmented and skewed image segments. Second, through integration, all points along an image segment are treated uniformly.

A fully general means of solving for the rotation, translation and scale of a model which minimizes ISPD is presented in this thesis. With the exception of underconstrained cases, the best-fit transformation is always uniquely determined by solving a quadratic equation. An interesting result also presented in this thesis is that minimizing ISPD subject to rotation and translation alone is more difficult. This 2D rigid case requires the solution of a quartic, rather than a quadratic, equation.

The development of novel local search neighborhoods and strategies has been important to the overall success of the work presented in this thesis. A new algorithm, named here *subset-convergent* local search, has been instrumental in reducing the average-case computational complexity to the roughly n^2 empirical estimate previously mentioned. The central idea behind the algorithm is that geometric matches which are locally (but not globally optimal) tend to result when different portions of a model become attached to incompatible image features. Subset-convergent local search takes such matches and disassociates from the match all but a subset of the corresponding image features. Doing this tends to allow local search to move beyond such undesired matches to those which are more globally consistent.

1.6 Overview

Chapter 2 reviews current approaches to recognition. The remaining chapters lay out the local search approach to geometric matching. The thesis builds through several stages, considering first weak-perspective problems and issues of computational complexity, and culminating with an algorithm for handling full 3D perspective. This algorithm, given a very rough estimate of an object's pose, solves simultaneously for the optimal correspondence and 3D pose.

Chapter 3 defines the discrete, combinatorial space of correspondences between object model and image features. It also defines, over this correspondence space, the match error function. The two fundamental terms of the match error, fit and omission, are described here. Optional terms capturing additional constraints are also presented. Addition of optional domain-dependent terms represents one way in which the general local search matching technique can be tailored to meet specific needs of particular domains.

Computing the match error depends upon obtaining the best spatial fit between the model and corresponding data. Chapter 4 fully derives the fitting algorithm for weak-perspective. The equivalent fitting procedure for full-perspective is presented in Chapter 7. Weak-perspective fitting minimizes an integrated-squared-perpendicular-distance (ISPD) measure between 2D model lines and 2D data line segments. The closed-form solutions for fitting a model to image data using this measure are presented. The case of fitting subject to rotation, translation and scaling is shown to be quadratic while the case of fitting subject to rotation and translation alone is quartic. A regularization term is also developed which minimizes squared distance between mid-points of corresponding model and image segments, and thereby resolves many otherwise underconstrained cases.

Chapter 5 introduces the use of local search for geometric matching. It reviews the ideas essential to local search, including local neighborhoods and random sampling. Specific neighborhoods are developed which are particularly well-suited to geometric matching. In particular, the *Hamming-distance-one* and *subset-convergent* neighborhoods are developed. Subset-convergent local search is shown to be very effective at breaking out of locally but not globally consistent geometric matches.

Chapter 6 takes up the question of how the computational demands of local search matching grow as a function of problem size. Formal analysis of local search complexity is known to be difficult [62], and empirical estimates are a common substitute. For a test suite of problems (also presented in Chapter 6) computation appears to grow as roughly n^2, where n is the number of potential pairings between model and data features. This rate of growth is not dramatically influenced by either the nature of clutter in an image or the structure of the object models.

Weak-perspective matching fails to capture the variability of full 3D perspective, and in some matching problems it is not possible to completely neglect full-perspective. Chapter 7 adapts the 3D pose algorithm developed by Kumar [70, 69] to support full-perspective matching. Several adaptations are described. One recasts the algorithm for computational purposes in terms of pose state vectors which are added and subtracted from a running sum when pairs of model-image features are added or removed from a match. Another adaptation involves adding a mid-point regularization term similar to that developed for weak-perspective fitting. This term uniquely determines the pose for many otherwise underdetermined sets of model-to-image feature correspondences.

Local search matching algorithms for full-perspective matching are developed in Chapter 8. These algorithms are demonstrated on problems which arise in the context of mobile robot navigation. The most conceptually straightforward extension substitutes *all* uses of weak-perspective fitting derived in Chapter 4 with full-perspective fitting derived in Chapter 7. Doing this yields a powerful algorithm, in that it readily solves problems involving perspective. However, due to the roughly order of magnitude increase in the cost of full-perspective fitting, this algorithms is slower than its weak-perspective cousin.

Through the development of a hybrid algorithm, Chapter 8 demonstrates that even sparing use of full-perspective yields an algorithm capable of solving matching problems that would are quite sensitive to perspective effects. The hybrid algorithm uses primarily the weak-perspective measures developed in Chapter 4 to guide the search process, and only uses full-perspective periodically. This hybrid algorithm combines the best attributes of both weak and full-perspective in order to solve perspective sensitive problems in little more time than that required by the cheaper weak-perspective only algorithm.

CHAPTER 2

Previous Work

2.1 Introduction

This chapter will emphasize the last ten years of work on recognition, focusing in particular on geometric model-based approaches which are neither object-specific nor domain-specific. Work on this general problem traces back at least to Roberts [95] in the mid 1960's, but the dominant approaches to solving geometric-based recognition tasks have all emerged since 1980. Although compartmentalizing a large body of work is risky, it is possible to group a significant portion of the work on geometric-based recognition into four broad categories: key-feature, generalized Hough, tree search and geometric hashing,. Briefly, the essential aspects of these four approaches are described below.

Key-feature: Key-feature algorithms search for highly distinctive local geometric features which indicate the placement of an object in a scene; the 'local-feature-focus' method of Bolles and Cain [14] is an early example.

Generalized Hough: Generalized Hough algorithms [28, 5], and more recently pose clustering [103] algorithms, emphasize search in the space of object model to image pose transformations.

Tree Search: Tree search algorithms [4, 46, 48, 16, 22] expand a tree of potential matches between model and image features using local geometric consistency constraints for pruning.

Geometric Hashing Geometric hashing and related approaches have arisen more recently [63, 72]. This work can be seen as a natural extension of the key-feature approach in which local geometric features are used to predict object identity as well as placement.

Before explaining the four common approaches to solving recognition tasks, Section 2.2 first reviews in broad terms the general statement of the geometric matching problem. Although there has yet to emerge a consensus upon the best way to solve recognition tasks, there is a developing nomenclature for describing the various aspects of the problem. It is important to understand these terms, and how they describe and constrain what is meant by recognition.

After reviewing the statement of the matching problem, Section 2.3 describes the key-feature, generalized Hough, tree search and geometric hashing approaches to matching. Each description summarizes the essential elements of the approach and cites both the earliest and most significant examples of its use. Where appropriate, some indication of the strengths and weaknesses of each approach will be given.

Section 2.4 summarizes local search matching as developed in this thesis and draws comparisons between this new approach to matching and work which has come before. One distinction that emerges is that local search matching combines random sampling and iterative improvement in a way never before used for geometric matching. Another distinction that local search, unlike the other common approaches, makes no clear distinction between hypothesizing and verifying a match. The other four approaches are all essentially techniques for hypothesizing either a correspondence or a pose. These hypotheses subsequently require verification. Finally, the imaging assumptions of previous works on matching are summarized, and this shows that with the possible exception of work by David Lowe [82], none of the work to come before has done quantitatively accurate matching under full-perspective.

2.2 Problem Overview

To better understand alternative ways of recognizing an object, it helps first to understand in more concrete terms what the problem entails. This can be done by reviewing the seminal work by Roberts [95]. Roberts restricted the general problem of recognition to the more constrained and correspondingly better defined problem of geometric matching, and it is largely from Roberts that we take the basic problem statement. Section 2.2.1 reviews the work of Roberts both to set a historical context and introduce fundamental themes which recur throughout the subsequent work on recognition.

2.2.1 Roberts Sets the Stage

What is most striking about the work of Roberts [95] is how he captured the essential elements of recognizing a geometrically modeled object. This includes the way in which he formulated the recognition problem, the representations and computational tools he thought would be important, and even some of the limitations he imposed upon the problem.

Roberts restricted the problem to that of recognizing a known, precisely modeled object. This led to a problem statement that, although difficult, is rich, well formulated, and ultimately solvable using geometry. This stands out in stark contrast to the more general problem of recognizing an instance drawn from a generic class of objects. To illustrate by example, Roberts limited the problem to that of recognizing *the chair*, not *a chair*, where 'the chair' is a specific chair with precisely known geometric dimensions. The more general problem of recognizing all things people call chairs is a classic problem in Artificial Intelligence, and is arguably unsolvable using only geometric constraints.

Roberts's work emphasized four things which have been recurrent themes in subsequent work, and which play a crucial role in the work of this thesis. These four things are:

Wire Frame Object Models An object was modeled using a rigid 3D wire frame made up of straight 3D line segments. A vast majority of the subsequent work on geometric based recognition has also used wire frame or variants upon wire frame models.

Image Line Segments Rather than look for evidence of an object directly in pixel intensity values, Roberts proposed that edges extracted from an image be grouped into locally straight segments, and that these segments be matched as symbolic entities to components of the wire frame model.

Least-Squares Pose Determination Since 2D segments in an image are projections of 3D segments in a wire frame model, it is possible to derives constraints upon the appearance of a model relative to the camera. Due to noise, these constraints will never be satisfied perfectly, but a least-squares technique ought to provide a means of estimating the object's pose.

Full-perspective Projection The appropriate model for how 3D model segments map to the 2D image plane is full-perspective projection.

This last point may seem obvious, but it deserves note when considered in light of the near complete failure of subsequent work to take account of perspective.

The details of Roberts's work look somewhat simple after nearly 30 years of research. For example, the object models he studied were comparatively rudimentary, based upon simple polygonal forms. However, his contribution in terms of circumscribing a rich class of problems, and his insight into the essential aspects of its solution, are worthy of note.

2.2.2 Terminology

In describing recognition problems and recognition algorithms it is important to be able to describe concisely different aspects of the problem. In the interests of doing this, here are a brief set of definitions.

Indexing: Determining which of possibly many object models are visible in an image. For example, consider all the possible pieces of furniture in a house, and then consider determining which are visible in a particular image.

Correspondence: The discrete mapping between features of an object model and features in an image. This mapping amounts to a statement that a 2D image feature is the projection of a corresponding object feature.

Pose: The spatial transformation relating the object model to the image or camera.

Projection: For 3D models, the process of mapping 3D points on an object model to 2D points in an image.

Imaging: The complete specification of the spatial mapping of 3D object features to the 2D image plane as defined by both the pose of the object with respect to the camera and the form of image projection.

Match: A correspondence and an associated pose.

2D-rigid Matching: Matching an object model to image features, assuming 2D-rigid imaging as described in Section 1.3.

Weak-perspective-2D Matching: Matching an object model to image features assuming weak-perspective-2D imaging as described in Section 1.3. Abbreviated to weak-perspective matching in this thesis.

Weak-perspective-3D Matching: Matching an object model to image features assuming weak-perspective-3D imaging as described in Section 1.3.

Full-perspective Matching: Matching and object model to image features assuming full-perspective imaging as described in Section 1.3.

The definitions of 'indexing', 'correspondence', 'pose' and 'projection' are consistent with common usage, including that of Grimson [48]. The term 'imaging' has been defined for this thesis in order to give a name for the complete process by which model features map to image features. The term 'match' is defined in a manner consistent with the algorithms developed in this thesis, and it is also consistent with most peoples intuition.

It is important to understand that indexing is ignored by most of the work on geometric recognition. Of the four basic approaches reviewed in this chapter, key-feature, generalized Hough, tree search and geometric hashing, only geometric hashing does indexing. As will become apparent, there are sound reasons for wanting, when possible, to avoid the indexing problem. Indexing adds one more layer of complexity to an already difficult problem. Like most other work on matching, the algorithms developed in this thesis will not deal with indexing.

Having removed, by assumption, indexing from the problem statement, a legitimate question is whether this restricted problem should be called recognition. The past standard for the field has been to say yes, and to conform with this standard, this thesis uses the term 'recognition' in its title and in more general descriptions of the problem. However, throughout most of the thesis the somewhat more precise term 'model matching', or simply 'matching', is favored.

The imaging process, which defines the spatial mapping between object features and image features, is an essential part of the overall description of a matching problem. The need to keep distinct the notion of pose and imaging can be overlooked when considering only 2D approaches to recognition. In these cases, 2D pose and imaging are synonymous. However, as Section 1.3 described, assumptions about object pose and 3D-to-2D projection can vary widely when working with 3D object models.

The way the space of correspondence mappings between model features and image features is defined says a lot about the robustness of an algorithm in the face of corrupted data. Despite obvious limitations, the majority of work on matching has assumed one-to-one correspondence mappings. The alternative mappings are reviewed in the next section.

2.2.3 Correspondence Mappings

It is critical to keep clear the distinction between the spatial mapping between features associated with imaging and the discrete correspondence mapping assigning image features to model features. The former is a continuous geometric mapping which transforms model features expressed in model coordinates to image features expressed in image coordinates. The later is a combinatoric discrete mapping associating model features with those image features presumed to be associated with, or caused by, the corresponding model feature. Just as previous work simplifies matching by assuming restricted forms of imaging, much of the previous work unrealistically simplifies matching by restricting allowable correspondence mappings.

If image feature extractions algorithms were perfect, then there would never be any need for a correspondence mapping which was not *one-to-one*: one feature on a model is manifested by a single corresponding feature in the image. This is not, of course, how real feature extraction works. The symbolic decomposition of an object model is often not reconstructed exactly by a feature extraction process, and it is not difficult to come up with situations in which it is important to map one or more model features to one image feature, and vice versa. For matching line segment models, the need for many-to-many mappings was already demonstrated in Section 1.4.

In this light, it is perhaps surprising that the vast majority of the work cited below presumes a match is a one-to-one correspondence between object model and image features ([63] [66], [105], [23], [26] [41], [56], [1] [55], and [22]). Since common line extraction algorithms [89, 20, 107] often fragment and sometimes accrete (over-group) segments, this one-to-one mapping assumption limits the applicability of these algorithms.

At a minimum, a robust matching algorithm must permit one-to-many mappings, so a single model feature may be mapped to several image fragments. Some common algorithms do permit such mappings, for instance [33, 98, 46]. However, none handle many-to-many mappings, and extending them to do so is problematic due to the immense increase in combinatorial possibilities. Only the algorithms developed in this thesis routinely assume many-to-many mappings between object model and image features.

2.2.4 Hypothesize then Verify

Essentially all the algorithms which fall into the key-feature, generalized Hough, tree search and geometric hashing categories generate either correspondence or pose hypothesis which must be subsequently verified. What this means in each case will become apparent when each approach is described below. However, a brief and simplified description can be given here.

Tree search algorithms hypothesize correspondence mappings which are potential matches, meaning there is likely but not necessarily a single pose transformation which aligns corresponding features. To determine which correspondences are good matches, it is necessary to run a separate verification procedure which tests for a single consistent pose. The key-feature, generalized Hough and geometric hashing approaches all use local features to hypothesize an object pose. It is then necessary to verify that there are in fact image features to correspond with each model feature in the hypothesized pose.

The reason for noting this common requirement is that it is not shared by the local search approach to geometric matching developed in this thesis. This point will be returned to in Section 2.4.7.

2.2.5 Object Specific Knowledge and Recognition

It would be a major omission if this review left the impression that there has not been work on more knowledge intensive and knowledge directed approaches to recognition. Approaches of this form are promising and interesting. The reason for their peripheral mention here is that they adopt a different set of generality assumptions than those made by the four classes of algorithms already mentioned and by the work developed in this thesis. Hence, they are not directly comparable.

To better see the different dimensions along which an algorithm can strive to be general, consider two essential elements of algorithms falling into the key-feature matching, generalized Hough, tree search and geometric hashing categories. First, they at least attempt to be general across a very wide range of geometric object models. What this means is they employ a single general procedure, regardless of the type of object being searched for. Second, drawing on the earlier discussion of the 'a chair' versus 'the chair' distinction, they all assume an essentially rigid geometric template as the object model, and hence will recognize 'the chair', but not 'a chair'.

There are other systems which employ more specialized knowledge and control strategies in order to perform recognition. Hence, these systems are more general in some ways, possibly recognizing objects drawn from more generically described object classes such as airplanes or buildings. However, they are more limited since their adaptation to recognizing different objects will require additional knowledge and modified control strategies.

A common knowledge-directed paradigm involves multiple levels of control based upon precompiled interpretation strategies. This is essentially the approach advocated by Brooks in ACRONYM [17]. This approach has been pushed further by Ikeuchi [58] in a system that automatically generates object recognition strategies. Brian Burns [19] has also worked to refine the multiple level of control approach through the development of what he calls 'view description networks'. Unlike geometric hashing, which has a distinct local feature identification stage followed by a table look-up stage, these algorithms embed geometric constraints into a multi-stage decision process.

Other largely geometric approaches draw heavily upon domain-specific knowledge. The Mosaic system developed by Herman and Kanade [84] uses fairly simple geometric constraints about buildings viewed from above to extract 3D structure from quite complex aerial images. The work of Heurtas [53] on automatic runway detection provides a quite different but equally compelling illustration of domain-specific and object-specific constraints.

Moving beyond shape, considerations such as color and context can be just as important and often more efficient sources of object hypotheses. For an excellent example of how color can be used to index object models, see the work of Swain [104]. Generally, heterogeneous sources of constraints are important, and a geometric matching algorithm should be thought of as one piece in a larger puzzle. Learning to put the puzzle together is the task of high-level vision research. For an excellent example of such integration see Draper [31].

2.3 Common Approaches

The four categories of geometric matching accurately reflect the major trends in the field. Obviously, the fit is not always perfect, and in practice individual algorithms sometimes exhibit attributes from more than one class. There is an approximate chronology underlying the order of presentation bellow. Key-feature matching [14, 80] covers a very broad range of specific algorithms, and is one of the more intuitive and early approaches suggested. The Generalized Hough approach [28, 5] emerged at essentially the same time, although pose clustering [103] was formulated later. The early contributions in tree search [4, 46, 48] came later than either key-feature or generalized Hough, and has yielded new and exciting improvements [16, 22]. Finally, geometric hashing [63, 72] is perhaps the most recent approach to be formulated, and as already noted, it is the only category that explicitly addresses model indexing as well as matching.

2.3.1 Key-feature Matching

They general idea behind the key-feature approach is this: when looking for an object in an image, look for something both simple to find and distinctive. This is in response to the fact that, as the number of features on a model grows, the number of possible correspondence mappings between model and image features explodes exponentially. The combinatorics of search are greatly reduced by considering only a small subset of model features. As this description suggests, key-feature algorithms divide matching into two stages. In the first stage, image data is searched for instances of key-features. The second step is to verify - either accepting or rejecting - each match hypothesized by an instance of a key-feature. An early example of this approach is the 'local feature focus' method proposed by Bolles and Cain [14]. David Lowe [79, 83, 80] in his work on perceptual organization and recognition extends and further motivates this approach.

There is a sense in which key-feature matching is recursive. The problem identifying a key-feature is essentially a problem of recognizing a model consisting of fewer parts. Since the key-feature has fewer components, the combinatorics associated with searching for them are correspondingly better. Additionally, if they truly are distinctive, finding a set of image features which correspond to a key-feature is tantamount to finding the object itself. However, there is a basic conflict between the need to keep the size of the key-features small and the need to define key-features which are distinctive.

When key-features get too simple they may become indistinctive, and hence are no longer 'key'. However, richly distinctive key-features tend to contain more parts, and the combinatorics of searching for them grows accordingly. Sitarman and Rosenfeld [99] have theorized that there is some 'optimum' size for key-features, and that this size may differ between domains. They provide a probabilistic analysis of cost in the context of relational graph matching. The intent is excellent and the underlying hypothesis almost certainly correct. However, Sitarman and Rosenfeld only consider 2D-rigid models translated in the plane. Hopefully, future work will extend their analysis to more interesting imaging models.

A simple use of the key-feature idea is evident in the work of Ayache and Faugeras [3]. In this work, 2D line segment models are matched to image segments. Long segments are assumed to be distinctive and are used as key-features. Under favorable conditions, such as looking at flat machine parts on a table under controlled lighting, this simple heuristic performs rather well. However, it is clearly a very weak heuristic for problems involving errorful image data or changes in 3D viewpoint.

A variety of key-feature algorithms have used local curvature properties to identify object silhouettes [66, 41, 1]. The silhouette recognition problem is simpler than recognition in grey-scale images. It essentially assumes a binary image from which the outline of one or more possibly overlapping objects is simply extracted. In addition, objects lie flat on a table top and are essentially 2 dimensional. Providing all these constraints are satisfied, then these technique are useful. Unfortunately, this work does not extend readily to more complex, non-binary, imagery. The combinatorics of constructing closed contours threatens to swamp that of recognition.

David Lowe's work is more general in several major ways. Lowe's SCERPO system [80, 81] was one of the earliest to recognize a 3D object based upon line segment features extracted from a moderately complex grey-scale image. Lowe did a particularly good job of quantitatively handling the problems associated with registering a 3D wire frame model to corresponding segments in an image. His early work [80, 81] assumed a weak-perspective-3D imaging, while his subsequent work on object tracking [82] handles full-perspective.

Lowe's contribution [80] to the key-feature approach is twofold. One contribution is his elegant argument for the centrality of perceptual organization in recognition. He argues that the human recognition system makes extensive use of feature grouping and abstraction processes, and that computer vision should do the same. In particular, Lowe argued that human recognition appears to depend upon the identification of distinctive structures. To illustrate his point, he uses a line drawing of a bicycle. Lowe observes that people see this bicycle easily so long as most of the line segments forming the front wheel are present. However, take away a few strategic pieces of this front wheel, and people have trouble recognizing the remaining segments as a bicycle. The wheel is serving as a key-feature and cueing the presence of the whole bicycle. Lowe's other contribution is to demonstrate with a working algorithm that key-features can play a significant role in recognizing 3D objects.

More recent work with 3D models and an approach somewhat like the key-feature approach is that of Huttenlocher [55]. In the algorithm developed by Huttenlocher, feature triples are used to hypothesize matches under weak-perspective-3D. The triples are ranked according to a distinctiveness heuristic and then searched in order. This approach resembles key-feature matching in that the small, hopefully distinctive features, are searched for first. It is more prudent, in that it works down the list of potential key-features as protection against failure. This approach has been demonstrated for a variety of 3D objects.

Perhaps the greatest failing associated with the key-feature approach is the lack of general algorithms for automatically selecting key-features. In Lowe's SCERPO system [80], the key-features for individual models were selected by hand. Clearly, in order to become a general method, this selection must be automated. Unfortunately, there appears to have been little progress to date in the area of automated key-feature selection.

One possible reason for this apparent lack of progress is that distinctiveness depends less upon the object itself than on the surrounding scene. Neither the geometric form of the object model, nor the aspects of the camera or sensor, matter as much as the nature of the other objects present in the world. A square, red, match box is easily seen on a white tablecloth, but not so easily seen on a red and white checked tablecloth. In controlled industrial settings it may be practical to catalogue and characterize the universe of possible objects and backgrounds. However, this is a daunting task for uncontrolled environments.

The ideas of key-feature matching have somewhat blended into more recent work on geometric hashing [63, 72, 73], which extends the key-feature idea in two important ways.

First, like the work of Huttenlocher [55], it relaxes the assumption that there exists a small number of key-features for each object model. Secondly, it suggests the use of local geometric features to select which object out of a possibly large set of objects is likely to be present in the scene. In keeping with the decision to review approaches roughly in order of their development, geometric hashing, which is the newest of the approaches, is discussed in Section 2.3.4.

2.3.2 Generalized Hough and Pose Clustering

The generalized Hough transform [28, 5, 27] algorithms and pose clustering [103] algorithms take an approach quite different from that of key-feature matching. Both shift the focus of search away from correspondence space and into pose space. Generalized Hough algorithms employ what are essentially voting schemes, while the more recently developed pose clustering algorithms identify tightly packed clusters of 'pose-distinctive' features. The meaning of 'pose-distinctive' feature will become clear shortly.

The intuition behind these approaches begins with the observation that partial matches between small numbers of model features and small numbers of image features often constrain the pose of the object model. To illustrate for 2D-rigid matching, one model line segment matched to one image segment constrains both the orientation, and to a lessor degree, the position. If an additional model segment is matched to an additional image segment, this will in general determine a specific orientation and position describing the pose of the model as a whole. These partial matches involving a small number of model and image features are pose-distinctive features.

When there is an instance of a model in the image data, then all the pose-distinctive features associated with this instance will map to essentially the same region of pose space. Typically, generalized Hough algorithms explicitly discretize the space into cells, and then each pose-distinctive feature adds its vote to the appropriate cell. The array of cells acts as an accumulator. A cell associated with the pose of a true match should receive as many votes as there are pose-distinctive features in a model. Conceptually it is a simple matter to search the pose space for cells with a sufficient votes to suggest a match. These cells, in turn, generate a hypothesized instances of the object with the indicated pose.

There have been a variety of algorithms using different variations of the generalized Hough transform. For a review of this work up to 1988 see Illingworth and Kittler [59]. For reasons to be discussed shortly, almost all the work on generalized Hough algorithms has focused on 2D matching problems. Some notable exceptions include the work of Silberberg, Harwood and Davis [98], who perform a restricted form of matching in which 3D objects are assumed to be on a stable ground plane and the camera is at a known height. As a consequence, their pose space is three dimensional. Thompson and Mundy [105] consider general unconstrained 3D views and then make judicious use of subspace projection. Specifically, votes are first projected from the six-dimensional space onto the three-dimensional subspace of rotations.

In the two cases just mentioned, either constraints upon pose or a subspace projection are used to reduce the dimensionality of the pose space which is searched for evidence of a match. The reason is quite simple, and is perhaps the major reason why the vast majority of generalized Hough work has avoided 3D matching. The problem is that explicit represenation of high dimensional, discretized, pose spaces is expensive and slow. Consider the memory required to represent a 2D-rigid pose space when each cell requires 1 byte of memory. If an image contains 512x512 pixels, and translation is discretized into 1 pixel cells, and if orientation is discretized into 1 degree increments, then nearly 95 mega-bytes of memory are required to represent the pose space. This is a lot of memory, but consider what happens if a fourth dimension is added in order to account for changes in scale. Presuming scale is discretized into 100 intervals, the pose space now requires nearly 10 giga-bytes of memory. Going beyond this four dimensional space of weak-perspective to six dimensions of 3D pose is intractable.

One of the advances associated with pose clustering over generalized Hough techniques is that it removes the need to explicitly represent the pose space. Stockman [103] was among the first to make the case for this variation of the basic generalized Hough idea. In pose clustering, pose-distinctive features are explicitly represented as points in N-dimensional pose space, and a clustering algorithm operating over this point set identifies tightly packed dense clusters. Each cluster is assumed to be the result of an object with the indicated pose. The advantage is that memory required for this algorithm is proportional to the dimensionality of the pose space times the number of points in the points set. A nice illustration of clustering local model-image feature matches based upon consistent pose hypothesis can be found in the work of Hwang [56].

Pose clustering, just like generalized Hough methods, falls victim to a second source of difficulty associated with higher dimensional pose spaces. As degrees of freedom are added to the pose transform, typically it is necessary to define new and more complicated pose-distinctive features. To illustrate this point with an example, consider 2D-rigid matching of straight line segment models. One model segment matched to one image segment constrains orientation, but because the image segment may be fragmented or possibly too long, the exact position or translation is not constrained. Typically, adding a second pair of matched segments constrains both rotation and translation, and is hence adequate for 2D-rigid matching. To account for changes in scale, it is typically necessary to add a third pair of matched segments.

The combinatorial complexity of finding pose-distinctive features in the image goes up exponentially in the number of individual object and image features required to constrain the pose. If pose-distinctive features include only 2 image segments, then testing all such possible features in an image is an order d^2 operation, where d is the number of line segments extracted from an image. This is the complexity associated with 2D-rigid matching. Allowing for changes in scale in order to perform weak-perspective-2D matching requires 3 segments, and the complexity jumps to d^3. Typically, to obtain reliable pose estimates under full-perspective requires 4 image segments, and the complexity increases to d^4.

Several efforts have been made to assess the strengths and weaknesses of generalized Hough matching. Brown [18] wrote a comparatively early assessment of noise sensitivity. More recently, Grimson and Huttenlocher [45] analyzed the sensitivity of the Hough Transform and conclude that false positive rates can become intolerably high in cluttered settings or when subspace projection is used. It seems, in general, that generalized Hough and pose clustering algorithms are quite useful for low-dimensional matching problems. For example, 2D-rigid and even weak-perspective matching. Their usefulness on higher-dimensional problems, such as full-perspective matching, is questionable.

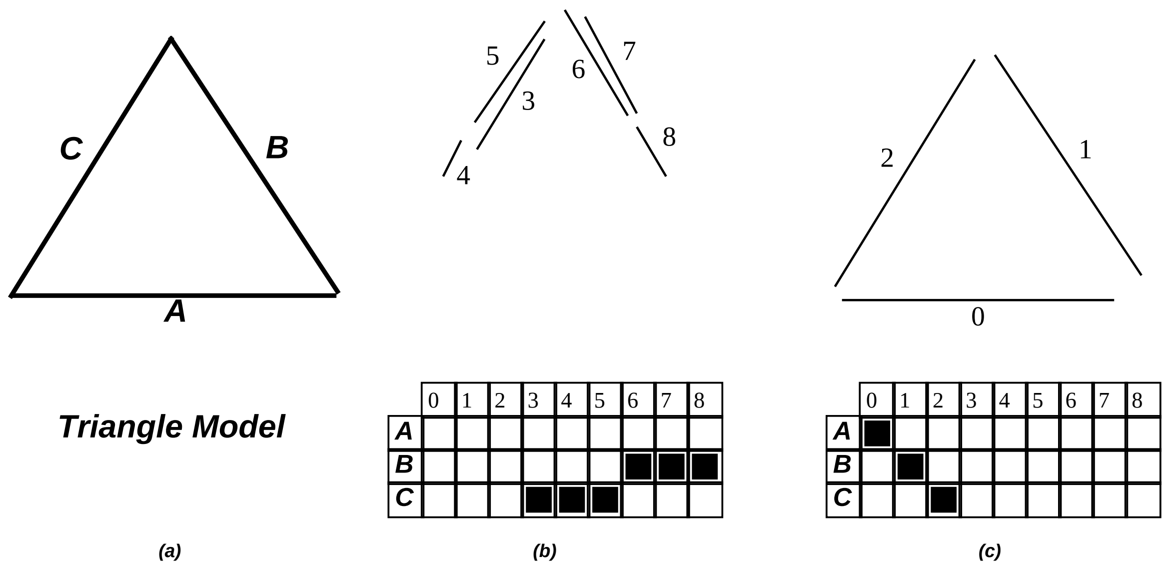


Figure 2.1 More pairs supporting a pose may not mean a better match.

Generalized Hough algorithms do not account for global consistency between object and image features. As illustrated in Figure 2.1, this leaves them open to another source of failure. Figure 2.1a shows three model segments forming a triangle, and Figures 2.1b and 2.1c show matches to two different sets of image segments. Filled in squares in the tables indicate corresponding pairs of model and image segments. The match in Figure 2.1c is the better, even thought there are more individual pairs of corresponding segments in the other match. A generalized Hough algorithm would incorrectly favor the match in Figure 2.1b because more pairs would vote for it than for the match in Figure 2.1c. This example illustrates why a subsequent verification step is so important.

2.3.3 Tree Search and Constraint Satisfaction

The 'tree' in tree search matching contains model-data correspondences, with the null correspondence at the root. The central idea is to expand the tree in order to find matches satisfying a set of geometric constraints. Tree search for matching geometric models to tactile sensor data was introduced by Gaston [37], and was employed by Baird [4] for 2D point matching. Grimson [46, 43, 42, 48] has done more than any other individual to promote tree search as a geometric matching technique, and his analytical studies [44, 47, 48] make tree search the best understood of the four approaches covered in this review. Recently, work by Beuler [16] and Cass [22] have extended Baird's ideas in some fundamental ways, leading to algorithms which intimately intertwine tree search with search in pose space.

The essential elements of tree search may be illustrated using the example in Figure 2.2, which shows a model consisting of 3 line segments, image data consisting of 4 line segments, and a fully expanded interpretation tree. The root, $\emptyset$, denotes the null match: no data segments are matched to model segments. Traversing down to level 1, each node represents a match between data segment 1 and either the indicated model segment or the null segment $\emptyset$. For example, the leftmost path down to level 1 represents a match between data segment

1 and model segment A. The rightmost path down to level 1 represents a match between data segment 1 and the null model segment $\emptyset$, the latter case implying no match.

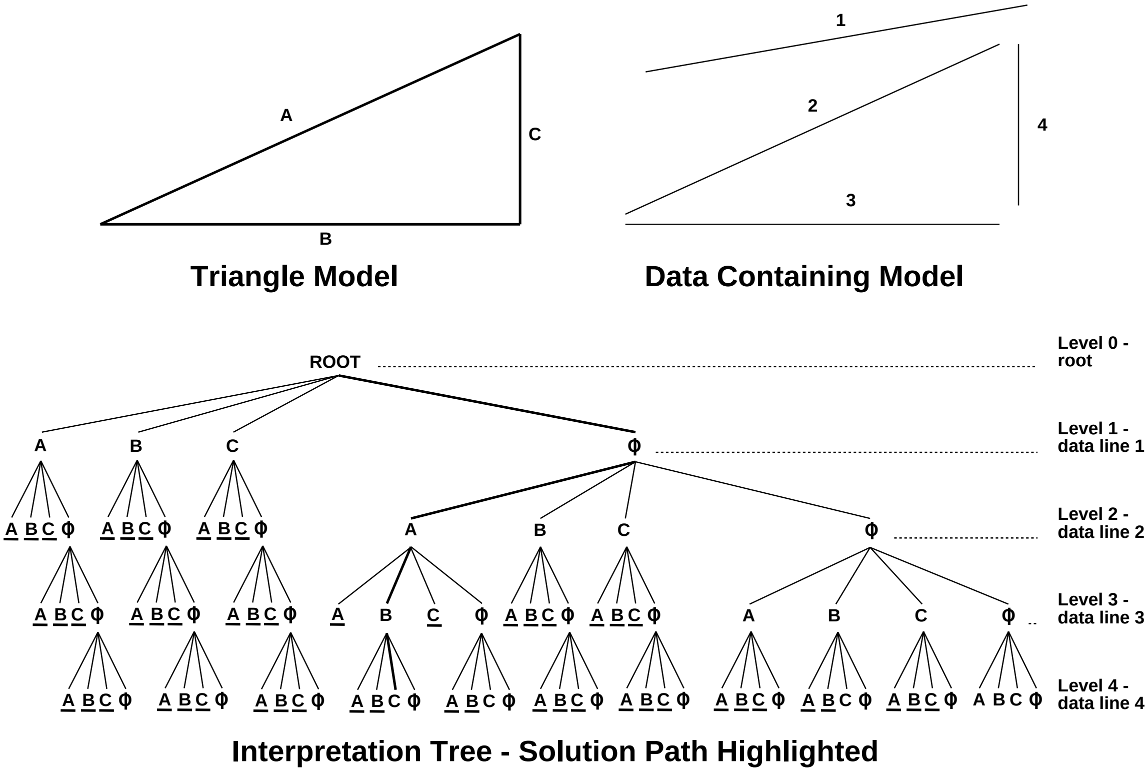


Figure 2.2 Illustration of tree search applied to geometric matching. Shown are a model consisting of 3 line segments, data consisting of 4 segments, and a fully expanded interpretation tree. Branches generating geometrically inconsistent matches are pruned.

The nodes on level 2 represents matches between data segment 2 and either the indicated model segment or the null segment $\emptyset$. At level 2, the first cases of pruning based upon geometric consistency can be seen. Underscored nodes indicate correspondence mappings which are geometrically inconsistent, and there is no reason to expand the tree further below these nodes. The solution path, fatter lines, represents the match $((1,\emptyset),(2,A),(3,B),(4,C))$. For the sake of this example, it has been assumed that only subsets of this match are geometrically consistent and pruning has been applied appropriately. As shown, this tree contains only 93 of a possible 341 nodes. Much of the analysis associated with tree search revolves around the question of how well geometric constraints prune the expanding interpretation tree.

The example just given illustrates tree search as formulated by Grimson and Lozano-Pérez [46] for recognizing rigid 2D objects in 2D image data, and 3D objects in 3D sensor data. In particular, Grimson introduced the use of the null segment $\emptyset$ to overcome the problem of spurious data. Earlier work, such as that of Gaston [37] and Baird [4], required each level of the tree to bind a data feature (or model if role of model and data are reversed) to a model feature. Baird was even more restrictive, requiring there to be exactly as many model points as data points, and defining a match as a complete one-to-one mapping between these two sets. In other words, a path down the tree was only considered a match if it reached the bottom.

A chronology of suggested strategies for searching the interpretation tree reflects the difficulty of problems being considered. In tactile sensing problems, the number of model and data features are typically small. Under these conditions, Gaston [37] found undirected breadth-first search to be acceptable. Certainly breadth-first search would be tractable for a problem of the size illustrated above. In the case of tactile sensing, breadth-first search can be improved upon by realizing that the acquisition of data is typically sequential and expensive in terms of time. Consequently, as shown in the work of Ellis [32], control strategies for tactile sensing can actively control the acquisition of sensor data based upon the degrees of freedom and possible ambiguities associated with the interpretation of the current tactile data.

Baird [4] and Grimson [46] favored depth-first over breadth-first search, no doubt in part because breadth-first search would have been extremely time consuming given the size of the problems they were studying. In addition to adopting depth-first search, Grimson cites two other factors as being important. The first is to rank choices so that the first acceptable interpretation is in some sense the 'best'. For example, data line segments can be ordered from longest to shortest, with the longest being matched first. A second important factor is to specify a stopping criterion which can be used to cut off search before a substantial portion of the interpretation tree has been expanded. An example of a stopping criterion is to call a halt to search once a consistent interpretation has been found involving data line segments whose total length exceeds a threshold.

Grimson's study of the behavior of tree search has been exemplary. These results are presented in two articles [44, 47] and more fully in his book [48]. He has determined the strength of pruning based upon local geometric consistency, the importance of a stopping criterion, and the associated problems of multiple model instances and symmetric models. To summarize these results, Grimson's analysis shows that for 2D-rigid matching with a stopping criterion the average-case computational complexity of tree search is $O(n^2)$, where n is the number of model features times the number of data features. He also shows that without the stopping criterion, tree search is exponential in the number of consistent interpretations.

The latter result, concerning performance without the stopping criterion, is interesting. The exponential performance is not the result of the algorithm wandering off into some portion of the search space unrelated to a consistent match. It is, instead, a consequence of the initial problem statement. The algorithm is designed to find *all* consistent matches between model and image features, and without a stopping criterion it dutifully enumerates them. In other words, when the largest consistent match contains k pairs of features, tree search enumerates all 2^k sets of pairs in the powerset of the largest consistent match.

Grimson shows that a principled stopping criterion can be derived from basic statistical assumptions about a problem domain, provided one assumes there is one and only one instance of the model present in the image data, and that the model does not have significant internal symmetries. However, he also highlights the two circumstances of multiple instances and symmetries, pointing out that they make specification of a stopping criterion problematic. This is an unfortunate limitation. Two or more objects of the same type often appear in the same image; consider telephone poles as one looks down a road. Also, both manufactured and biological objects frequently have internal symmetries.

Stepping back from Grimson's work, Baird's work is of little practical use in computer vision because of his overly restricted assumptions about the correspondence mapping. It is almost inconceivable that one could find a domain so well behaved that image data would be guaranteed to contain exactly one and only one extracted image feature for every model feature. However, Baird pioneered the use of linear programming as a means of testing the global geometric consistency of a match, and his formulation of matching is conceptually simple, general, and hence elegant. Formally, the problem Baird defines is to find all consistent bipartite mappings between two sets of points, where consistent means there exists a pose transformation placing each model point within a convex error polygon defined about its corresponding image point. Baird shows that testing whether a match is consistent via this definition is equivalent to testing whether there exists a solution to a set of linear inequalities. He therefore is able to take advantage of well known linear programming algorithms. Baird has shown the complexity of his algorithm to be $O(m^2)$, where m is the number of points in the model.

More recently, Beuler [16] and Cass [22] have overcome some of the difficulties encountered by Grimson's algorithms while utilizing Baird's consistency measure in cases involving extraneous and missing data. Cass has coined the term *pose equivalence analysis* for his work. His algorithm searches simultaneously in correspondence and pose space for maximal sets of pose consistent features. The algorithm is polynomial. However, for the case involving rotation, translation and scaling of 2D models, the complexity is $O(k^4 m^4 d^4)$ for m model features and d data features. Here k is the number of sides bounding a polygonal uncertainty region about each feature. As Cass readily notes, although the polynomial bound is of tremendous theoretical interest, it is too high for the algorithm to be put to direct practical use. Given the centrality of random sampling to the work in this thesis, it is interesting to observe that one means suggested by Cass for developing a more tractable algorithm is to use random sampling as the basis for a randomized version of the pose equivalence analysis algorithm.

To close this section, it is worth making some comments about match verification and global versus local geometric consistency. In Grimson's 1987 paper, he compared global versus local tests of geometric consistency between matched features. In this context, global consistency means that all pairs of matched features are consistent with a single global pose transformation. In contrast, local consistency means that matched features are pairwise consistent, but not necessarily globally consistent. Making up an example for the sake of illustration, say paired features $(A, 1)$ and $(B, 2)$ are consistent and imply a pose $\mathcal{F}_1$, and paired features $(B, 2)$ and $(C, 3)$ are consistent and imply a pose $\mathcal{F}_2$. For the triplet of features $(A, 1), (B, 2)(C, 3)$ to be globally consistent, $\mathcal{F}_1$ and $\mathcal{F}_2$ must be essentially equal. In contrast, under many tests of pairwise consistency, $\mathcal{F}_1$ and $\mathcal{F}_2$ may be different poses, in which case the triplet $(A, 1), (B, 2), (C, 3)$ is locally but not globally consistent.

Grimson concluded that testing for global consistency was costly, and didn't significantly improve the performance of tree search relative to the less costly use of local consistency checks. In his subsequent work, Grimson has been a proponent of local consistency checks. In contrast to Grimson, Baird, Beuler and Cass all utilize a global test of geometric consistency. Global consistency testing is more involved, and developing efficient tests for global consistency is a major facet of each of these individual research efforts.

Global consistency, although a strong constraint, is not equivalent to verification. Consider again the example in Figure 2.1 at the close of the section on generalized Hough transforms. In this example, both of the obvious matches are globally consistent, yet further verification is required to explicitly identify the difference between the two, and thereby determine the one with fewer pairs of matching features is better. Thus, tree search, like the generalized Hough transform, is a means of hypothesizing matches which require subsequent verification.

2.3.4 Geometric Hashing

Not unlike the key-feature approach, geometric hashing [63, 72, 73] seeks to identify features in an image that predict the presence of an object. The approach differs from key-feature matching in several important respects. Most notably, geometric hashing deals directly with multiple object models. Also, geometric hashing does not rely upon a few key-features, but instead attempts to exploit the predictive power of all subsets of image features.

There is an off-line and an on-line stage to geometric hashing. In an off-line operation, a database of geometric models is analyzed. The product of this analysis is typically a hash-table. The cost of this analysis is not part of the cost of recognition. The on-line part of the algorithm consists of selecting sets of features in image data, and looking-up the model or models to which these features correspond.

Examining one set of image features in order to determine the identity of an object would be adequate if : 1) all sets of features were uniquely predictive, 2) only one model were present in the image, and 3) the image were free from clutter. Unfortunately, not all sets of features are uniquely predictive and images often contain multiple model instances and clutter. Consequently, geometric hashing in practice looks at many sets of image features and accumulates votes in order to hypothesize the presence of an object.

In the earliest work typically thought of as geometric hashing, Kalvin [63] *et. al.* hash on a five-dimensional feature vector related to a measure of arc length versus turning-angle as measured over a contiguous subsection of an objects boundary. As emphasized in this work, the hashing metric must be invariant to the class of object pose transformations. The arc length versus turning-angle measure is invariant under 2D-rigid transformations. It is not, however, invariant with respect to variations in scale.

Lamdan [72, 73] *et. al.* extend and refine the geometric hashing idea. Their hashing is based upon a four-tuple of points, in which the first three points define the basis vectors of a 2D coordinate system. The hashing measure is the coordinates of the fourth point as measured with respect to the basis vectors, and this measure is invariant with respect to a full 6 parameter 2D affine transformations. For 2D similarity transformations, a measure based upon a three-tuple of points is suggested.

Geoemteric hashing suffers from two basic problems: sensitivity to noise and clutter. Grimson and Huttenlocher [49] suggest that Lamdan's choice of metrics is quite sensitive to noise, leading to a high false positive rate even for modest amounts of noise and clutter. One problem is that point coordinates are highly sensitive to noise when measured with respect to bases vectors defined by nearly parallel triples of points. Costa and Shapiro [25] similarly observed error sensitivity and have refined the geometric hashing approach using likelihood estimation.

Clutter is another source of problems. Imagine a data set in which 1 in 5 of the data points is a clutter point. Drawing sets of 4 points at random, the probability of all four points being associated with the model is only 1 in 5. For the other 4 out of 5 point sets, looking up a model in the hash table is a waist of time and a potential source of mistakes. Now imagine if 9 out of 10 points are clutter, in which case the odds of randomly selecting 4 good points would be 1 out of $10,000$.

Others who have pushed the the underlying ideas of geometric hashing have found it necessary to put considerable effort into designing algorithms to solve the problem of selecting a set of features all belonging to same object. For instance, Stein [101] has shown quite nice results using a hashed indexing scheme on 2D object models. To overcome the problem of selecting features from the same object, his approach uses boundary contour information. Stein defines super segments which are consecutive sides of a polygonal boundary. A feature vector defined in terms of the relative placement of consecutive segments in the super segment is used to index object models.

More recently, Stein has extended this work to the problem of matching 3D object models to 2D image features [102]. Stein emphasizes the importance of grouping control mechanisms to obtain a reasonable starting set of features. Interestingly, as one who has tried to extend geometric hashing to 3D full perspective matching, he argues that this is very difficult. He instead advocates the use of topological constraints between fairly complex image features.

Geometric hashing was a highly promising approach whose performance to date has fallen short of expectations. To its credit, it deals with indexing into databases containing multiple object models, and performs well for a class of modestly simple 2D recognition problems. However, no effective full-perspective geometric hashing scheme has been demonstrated, and given the known difficulties associated with defining invariants under full-perspective [21], such a generalization may be infeasible.

2.4 Local Search Matching in Relation to Previous Work

This section compares aspects of the previous work on geometric matching with the local search matching approach set forth in this thesis. It does this by first reviewing the origins of local search and then briefly describing how local search is applied to the problem of geometric matching. Included here is a brief discussion of the relationship between local search and both simulated annealing and genetic algorithms.

2.4.1 The Origins and Essence of Local Search

The origins of local search are commonly traced to the work of Kernighan [64] and Lin [75, 76]. In these papers, Kernighan and Lin demonstrated that fairly simple iterative improvement algorithms find near optimal solutions to such problems as the traveling salesperson problem (TSP) and the graph partitioning problem. A succinct description of local search is provided by Papadimitriou and Steiglitz [92] in their text on combinatorial optimization.

Kernighan and Lin [75, 76] define a discrete neighborhood about each feasible solution in the combinatorial search space. This neighborhood establishes the options available to local search. For TSP, they define a k-swap neighborhood in which the neighbors of a tour is all tours formed by swapping the positions of k cities. The simplest of these neighborhoods is the 2-swap neighborhood generated by all possible swaps of 2 cities. Local search for TSP starts from an initial tour and repeatedly tests neighborhoods for improvement. If a neighboring tour is found to be better, then the algorithms adopts this as a the best tour found so far. Local search repeats this process until the current best tour is better than all its neighbors. Such a tour is locally optimal.

A single execution of local search may become stuck on a local optima of little interest. One way to lessen the problems associated with local optima is to use alternative neighborhood definitions. Different neighborhoods can have profoundly different properties. In general terms, one would expect larger neighborhoods to decrease the problem of local optima while increasing the cost of search. For TSP, Kernighan and Lin studied 2-swap neighborhoods, 3-swap neighborhoods, and other variants. The neighborhoods for graph partitioning [64] are similar; a 2-swap neighborhood contains all graphs obtained by swapping a pair of nodes.

Random sampling may be used to increase the probability of finding the optimal or a near optimal solution. Multiple trials of local search are initiated from independently chosen random starting points and the best solution found is retained. Careful neighborhood design, coupled with random sampling, can make local search a powerful tool for solving complicated combinatorial optimization problems.

2.4.2 A Brief Mention of Alternate Optimization Techniques

Two other heuristic approaches to solving difficult combinatorial optimization problems deserve mention, and these are simulated annealing algorithms [65] and genetic algorithms [50, 94]. Simulated annealing, as its name suggests, is inspired by the model from physics of a material reaching a minimum energy state through a process of gradual cooling. When using simulated annealing to perform optimization, energy is typically equated to the objective function to be minimized. Simulated annealing is a type of stochastic relaxation.

In practical terms, a simulated annealing algorithm iteratively selects new states using a probabilistic update rule. Selection depends both upon the relative energy of the two states and the *temperature*. The higher the temperature, the more selection will be random, and the lower the temperature, the more the selection will be made in favor of the lower energy (lower error) state. In the limit, with temperature zero, stochastic relaxation will always select the lower energy state. At this extreme, simulated annealing behaves like local search.

Geman and Geman [39] provide a formal convergence proof for stochastic relaxation which can be briefly summarized as follows. Given a cooling schedule of appropriate form, in the limit as annealing time goes to infinity relaxation converges upon the globally optimal solution with probability one. In practice of course, cooling must involve a tractable number of iterations, and simulated annealing may converge to a locally optimal solution.

Genetic algorithms take their inspiration from biology rather than from physics. Typically, a population of solutions to an optimization problem both breed and mutate to create a new generation of solutions. This process is repeated many times. Survival pressures are applied which make better, more optimal, solutions more likely to reproduce.

Local search and genetic algorithms are connected through work on hybrid genetic algorithms [29] and parallel genetic algorithms. In a recent paper on parallel genetic algorithms, Mühlenbein [87] makes two relevant points: 1) in parallel genetic algorithms, individuals within a populations of solutions improve themselves using local hill-climbing, and 2) only solutions 'near' each other have the opportunity to breed. The hill-climbing operation is essentially local search, and in the limiting case a parallel genetic algorithm with no breeding is equivalent to random-starts of local search.

It is impossible to make one categorically correct, sweeping, judgement about how well local search compares to alternative approaches such as simulated annealing and genetic algorithms. Different heuristic methods excel under different circumstances on different problems, and predicting how a particular method will perform when adapted to a new problem domain remains an art. In one of the more thorough of the published comparisons, Johnson, Aragon, McGeoch and Schevon [60, 61] compare simulated annealing with other more traditional approaches, typically random-starts local search. The comparison is conducted for three problems: graph partitioning [60], graph coloring [61] and number partitioning [61].

In [60], Johnson et. al. set out to answer two questions: 1) how various design choices associated with simulated annealing affect performance, and 2) how well an optimized implementation of simulated annealing performed relative to a random-starts Kernighan and Lin style algorithm. To summarize their answer to the first question: details matter. Variations within a single general algorithm lead to major variations in performance. Their answer to the second question is captured in the following quotation from [61].

> For graph partitioning, the answer to the second question was mixed: simulated annealing tends to dominate traditional techniques (random-starts local search) on random graphs as the size and/or density of the graphs increases, but was roundly beaten on graphs with built-in geometric structure.

Of local search, simulated annealing, and genetic algorithms, local search is arguably the simplest. It is for this reason, more than any other, that this thesis adapts local search to the problem of geometric matching. Future work should consider adapting these other two forms of non-linear optimization to the geometric matching problem.

2.4.3 Adapting Local Search to Geometric Matching

Five things must be defined in order to adapt local search to a new domain: the search space, the objective function, the source of initial starting solutions, local neighborhoods, and search strategies. The search spaces considered in this thesis consist of many-to-many mappings between object model features and image features. As illustrated in the example from Figure 1.1 in Section 1.1, although many-to-many, the mapping may be restricted to consider only subsets of all possible pairings.

The objective function is a match error function defined over *all* correspondence mappings for which a unique best-fit pose of the model relative to the data can be computed, typically all correspondence mappings containing more than two pairs of corresponding features. The initial matches from which local search is initiated are randomly selected from the space of all possible correspondence mappings. Two local neighborhoods are developed, the simpler of the two consisting of all matches obtained by adding or removing a single pair of matched segments. Two of the simplest strategies for searching a local neighborhoods are considered in this thesis: first-improvement and steepest-descent. In first-improvement, the first neighbor found to be better is immediately made the current match. In steepest-descent, the entire neighborhood is searched, and the neighbor yielding the greatest improvement is made the current match. Steepest-descent is used in this thesis to pick which pair of model-image features is to be added or removed from a match.

This thesis develops a more sophisticated neighborhood and associated search strategy specifically suited to the geometric matching problem. The name *subset-convergent local search* is given here to this new technique. The underlying idea is to initiate search from subsets of a geometric match as a means of breaking out of local optima. What happens is that if a match is truly good in a geometric sense, search initiated from subsets (partial matches) simply returns to the good match. However, more importantly, if the match is locally optimal in terms of small changes, but not globally good, search from subsets tends to lead to better overall matches. The discovery of the subset-convergent algorithm was essential to the successful application of local search to geometric matching. The details of the algorithm are described in Chapter 5. Conclusions about the power of local search matching given here are typically for the subset-convergent algorithm.

2.4.4 The Issue of Acceptable versus Best Matches?

Most of the previous work cited formulates matching as a constraint satisfaction problem. The goal is an acceptable match. What constitutes acceptable can be rather implicit, where verifying a match simply involves finding additional features consistent with a presumed pose [55]. Acceptable can be more explicit, as in the work of Grimson [46] or Cass [22]. Very little of the work on matching explicitly defines the goal as that of optimization: find the best match. One notable exception is the work of Mohan [85], who uses a Hopfield network to identify optimal sets of image features matching a generalized building template. Another is recent work by Li [74] on optimal weak-perspective-2D line matching using relaxation labelling.

There is a superficial similarity between feature counting, which is common to many of the approaches mentioned above, and optimization based upon more substantive measures of match quality. Often algorithms count pairs of features that are consistent with a single object pose. This is the basis for the generalized Hough transform which looks for poses with the most votes. Algorithms emphasizing pairwise geometric consistency also sometimes seek maximal cliques in order to guarantee global geometric consistency [14, 15].

Grimson [46] counts the number of consistent features found during tree search and terminates when the count exceeds a threshold. Breuel [16] goes somewhat beyond this by stating the goal to be the largest set of consistent paired features. What should be noted about all of these uses of paired feature counts is they do not really draw upon underlying geometric similarity as the basis for ranking. Moreover, counting paired features unfortunately makes fragmented data look better, since fragments lead to a greater total number of consistent features. Recall the example of Figure 2.1 on page 29.

This thesis develops a match error function which measures geometric similarity between a model and corresponding image features. It does this by combining two simple notions. The first is that a model should fit the corresponding image data. As has been done before [80, 3], fit depends upon the residual error after a least-squares procedure is used to determine placement of the model relative to the data. The second notion is that corresponding data features should appear where a model feature projects into an image. Failure to find corresponding data is an an error of omission.

These two forms of error, fit and omission, have seldom been combined before in the context of model matching. A notable exception can be found in the discussion of matching metrics in Ballard and Brown [6]. They define a cost function for spring template matching that combines through summation a term related to what they call the spring cost and a term related to what they call missing. This is conceptually similar to the fit and omission error measures developed in this thesis.

The match error developed here differs in important ways from other metrics recently suggested in the literature. A recent paper [2] suggests a good metric for evaluating matches must be symmetric. In other words, it should return the same value regardless of which set of data is called the model and which is called the data. This is not appropriate in the context of model directed vision, where models known with moderately high precision are matched to far less perfect image data. Fragmentation of image data is qualitatively different from fragmentation in a stored object model. It does not make sense to equate the two by demanding a match error function be symmetric.

Wells [57] derives an evaluation function, a measure of Bayesian likelihood under a simple noise model. Of greatest relevance to this thesis, his measure sums a residual fit error with a penalty for *unmatched data* and does not take account of model features missing in the data. Due to the dependence upon data left unmatched, an identical set of matched features might have a good score in one image and a bad score in another. If the goal is to measure the quality of the match between object model and *corresponding* image features rather than the data as a whole, this indirect dependency upon unmatched features is a failing.

2.4.5 The Ancestry of Random Sampling

Fischler and Bolles [35] argued for an approach to matching called random sample consensus. If some small number of features, for instance three model segments to three data segments, typically determine an object's pose, then the space of possible triples may be sampled at random. In problems where a substantial number of the possible triples are correct - they belong to a true match - then this strategy will with high confidence find a true triple in a modest number of trials. Moreover, other correct triples will be in 'consensus': they will agree on the object's pose.

There is a nice relationship between Fischler and Bolles ideas and the use of median filtering to estimate an object's pose. Siegel [96] (alternatively [97]) and Benson provide a nice example in which median filtering is used for 2D point matching. Their problem is to compare biological shapes: for instance skulls of different animals. They show that median filtering, which seeks a majority subset of the possible paired features which minimizes the median squared error, removes pairs which throw off the overall fit. In this work, the median was computed exhaustively and there was no need for random sampling. However, this was made possible by assuming different components of the transformation were independent. This is not an ideal assumption.

Kumar [71], in his outlier detection algorithm, brings together the best of the random sampling and median filtering. For the 3D pose problem, prior attempts to treat transformation parameters as independent produced inferior results [71]. Therefore, Kumar's algorithm uses k-tuples of model-data pairs to generate a single estimate of the pose. In principle the algorithm would compute the pose for every possible k-tuple. For each pose it would determine squared error for all pairs of corresponding features and rank these in order to determine the median error. The pose with the lowest median error would be selected as best and those pairs in consensus with this pose are then taken to be the true match. Because the space of k-tuples is generally too large to do this exhaustively, Kumar randomly samples this space of k-tuples.

Siegel's algorithm and Kumar's algorithm can be properly called matching algorithms because they make a discrete choice as to which pairs to include and which pairs to exclude from a match. As said at the outset, they are only applicable to easy problems because they must either make overly simplistic assumptions about the transformation [96], or they must employ random sampling to generate a true kernel [71]. This latter step breaks down as the number of true model-data pairs begins to drop below 50 percent of the total number of pairs.

2.4.6 The Ancestry of Iterative Improvement

Some of the inspiration for local search matching as developed in this thesis can be found in the way key-feature algorithms *fill-out* an initial match. Once an initial match between a subset of model features and a set of key-features in an image has been established, key-feature algorithms typically verify the match by iteratively adding additional pairs of features. After each addition, the best-fit pose of the object with respect to the image is updated. Both the algorithm developed by Lowe [80] and the algorithm developed by Ayache and Faugeras [3] fill-out matches in this way. In filling-out a match, one can observe a compelling interplay between adding pairs of features to the match and updating the object pose estimate.

If the object model is displayed in registration with the image data, then what one sees is that often the initial fit of the model to the data is poor. Those features which are matched are aligned, but features not yet matched are frequently misaligned, and misalignment is worse for model features farther away from those already matched. However, as additional pairs of corresponding features are added to the match, one literally sees the model move into proper alignment with the data. The interplay between pose updating and adding pairs means that as the object's pose relative to the image improves, selecting correctly which pair of features to add next becomes easier.

The local search matching algorithm illustrated in Section 1.1 will perform much like the algorithms just described if initiated from a key-feature match. It will successively add pairs of features and refine the object pose estimate accordingly. However, unlike the algorithms of Lowe and Ayache, if one of the algorithms developed in this thesis thesis adds a pair of features which later appear not to match, it can *remove* them. This outwardly simple enhancement has far reaching consequences. It means the initial assignment of corresponding pairs need no longer be perfect in the sense of being a perfect subset of the true match. Moreover, it opens up the possibility of search moving about more freely through the space of possible correspondence.

2.4.7 Recognizing while Locating

Section 2.2.4 described the manner in which key-feature matching, generalized Hough matching, tree search and geometric hashing are all processes in which matches are hypothesized based upon local geometric evidence, and are subsequently tested for global geometric consistency. Local search matching in this thesis departs fundamentally from this model, and there are no such distinct phases of processing at work.

At all times during local search matching, the best-fit pose of the object model relative to the image is known, and the extent to which the corresponding features are globally consistent is expressed by the match error function. When a match of sufficiently low error is found, it is already known to be consistent. There is no need for verification of global consistency.

There is also nothing resembling a hypothesis generation step. Instead, there is an iterative movement from an initial, usually poor match, to one which is locally optimal. During this movement or search, there is rich interplay between the addition and removal of paired features from the correspondence mapping and the determination of the best-fit pose of the model relative to the image. Adding or removing a pair of features changes the best-fit pose of the model with respect to the image. This, in turn, changes the relative placement of model and image features, and thereby changes what pairs may next be added or removed from the match.

Faugeras and Herbert [33] coined a phrase in a paper on recognition of 3D objects in 3D sensor data that describes beautifully what local search matching is doing. It appears in a set of three general conclusions about matching drawn from their introduction.

> 1 Representations should be in terms of linear primitives, such as points, lines, and planes, even if at some intermediate level we deal with things like curved surface patches.
>
> 2 The fundamental constraint to be exploited is that of rigidity.
>
> 3 The basic paradigm to be used is that of recognizing while locating (or vice versa).

The details of their work differ from those developed in this thesis, but their conclusions fit this thesis as though they were written for it. In particular, the search carried out by the algorithms in this thesis can be succinctly described as 'recognizing while locating'.

2.4.8 Full-perspective Matching

Table 2.1 reiterates the four common types of imaging and lists researchers who have used each. The lists for 2D-rigid, weak-perspective-2D and weak-perspective-3D are by no means exhaustive. However, the work of Lowe [82] and the work presented in this thesis appear to be the only work to deal quantitatively with full 3D perspective during matching. One could argue for including also Stein's recent work [102], but it is omitted because Stein emphasizes qualitative over quantitative relationships when dealing with full-perspective.

Without going into any great detail, there are perhaps two primary reasons why full-perspective has been avoided. First, determining the pose of an object under full-perspective

Table 2.1 Partial lists of previous work broken out by imaging model. To help recall of imaging models, summaries from section 1.3 are repeated here.

Imaging	**Used by ...**
2D-rigid	
Flat objects perpendicular to the camera. The distance from the camera is known.	Kalvin [63] Grimson [46] Stockman [103] Hwang [56] Beveridge [11]
Weak-perspective-2D	
Flat objects perpendicular to the camera but at unknown distance and placement relative to the camera. Also useful for general 3D objects given further restrictions upon possible viewpoint.	Ayache [3] Gottschalk [41] Costa [25] Grimson [48] Beveridge [12] Cass [22]
Weak-perspective-3D	
Any 3D object shallow in depth compared to its distance from the camera and viewed from any arbitrary viewpoint. Some distortion induced for all but perfectly flat objects.	Lowe [80] Thompson [105] Huttenlocher [55]
Full-perspective	
Any 3D object viewed from any arbitrary viewpoint. Full-perspective is an excellent first order approximation for a standard camera. Work listed requires an initial approximate pose estimate.	Lowe [82] Beveridge [10]

is not trivial. Developing algorithms to do this in a robust and general fashion has itself been a major topic of research, and only in the past few years have Kumar [70, 69] and Lowe [82] developed fairly nice algorithms for pose determination.

A second reason why full-perspective has been avoided is the common reliance on local geometric constraints. These constraints are defined relative to small sets of corresponding features. The number of features required grows, and the degree of constraint shrinks, the closer the imaging model comes to full-perspective. For example, in geometric hashing [72], 2 points are required to establish a basis under weak-perspective-2D imaging. For full-perspective the number of points required grows to 5. This dramatically increases the computational complexity of the approach.

2.4.9 Computational Complexity

It is critical to understand how the computational demands of the algorithms developed in this thesis compare to other known matching algorithms. However, making such a comparison is as difficult as it is important. There are unavoidable weaknesses in the comparison which follows, and these are noted. One obvious problem is that different matching algorithms often do not solve equivalent problems. All difficulties aside, there is value in attempting a comparison.

Here local search matching, as developed in this thesis, is compared with generalized Hough, tree search [48] and pose equivalence analysis [22]. Key-feature approaches are not considered. Assessing their average-case computational complexity is problematic because the computation depends fundamentally upon the extent to which the key-features are distinctive, and this is difficult to determine in a general manner. In contrast, the generalized Hough, tree search and pose equivalence analysis approaches have generic algorithmic forms.

Analytically determining complexity bounds for local search algorithms is very difficult [62, 67], and it should be remembered that the rate of growth in computational demand for local search matching reported in this thesis is empirically estimated. Over a test suite of varying sized problems, the average case computational demand grew as roughly n^2, where n is the number of possibly matching pairs of model and image features. Extrapolating from these tests, it is conjectured that the computational complexity of local search matching is $O(n^2)$. Clearly this conjecture is weak. However, it is offered as the best guess of how the performance scales with increasing problem size. It is worth noting that the test suite of problems studied include known hard cases, specifically symmetric models and multiple model instances. These are both conditions causing the average case computational demand of tree search [48] to become exponential.

Table 2.2 compares complexity estimates for local search, generalized Hough, tree search [48] and pose equivalence analysis [22] algorithms. The generalized Hough estimate assumes possibly fragmented line segments as the primitive features. For 2D-rigid models two pairs of matched segments establish the pose. There are $m^2d^2 = n^2$ pose distinctive features and n^2 is the complexity of the voting step of the algorithm. The average case complexity for tree search has been derived analytically by Grimson [48]. The two bounds represent tree search with and without termination. Without termination the algorithm enumerates all 2^l subsets of the maximally consistent match. The bound on pose equivalence analysis is derived by Cass [22].

Table 2.2 Complexity estimates for general approaches to geometric matching. n is the number of potentially matching pairs of model and image features. The 2^l case for tree search is for a consistent match of size l searched without a termination criterion. The k for pose equivalence analysis depends upon the number of sides on the convex error polygon in which corresponding features must lie. It must be stressed that the value given for local search is a weak conjecture based upon extrapolated empirical data.

Approach	Complexity
Generalized Hough Transform	
2D-Rigid	n^2
Weak-perspective (2D)	n^3
Tree Search	
2D-Rigid	n^2 or 2^l
Pose Equivalence Analysis	
Weak-perspective-2D	$k\,n^4$
Local Search	
Weak-perspective-2D	n^2

Unlike local search, tree search, generalized Hough and pose equivalence analysis do not determine globally consistent matches. Thus, with each of these methods there is some additional verification cost not included in these estimates. Unlike local search, which finds probabilistically optimal matches, the other three deterministically find locally consistent matches. Pose equivalence analysis goes somewhat beyond this by finding maximal sets of pairwise consistent pose distinctive features. However, pose equivalence analysis assumes that the n pairs of features are pose distinctive. A model line segment paired with a potentially fragmented image segment is not pose distinctive.

Even accounting for the differences in problem statement and the means of estimating complexity bounds, the local search approach to matching compares favorably to these other general methods. Add this to the advantage that the local search approach is demonstrated as extending to full-perspective matching, and the advantage that it handles many-to-many correspondence mappings, and it should be clear why local search matching is a promising alternative to these other techniques. The remainder of this thesis will proceed to explain the various aspects of local search matching.

CHAPTER 3

Matching as Combinatorial Optimization

3.1 Introduction

To formulate geometric matching as optimization the following must be specified:

1) The model representation.

2) The data representation.

3) The discrete space of possible model-data mappings.

4) The match error.

Of these four elements, the design of the match error function is by far the most involved. This introduction will motivate some of the factors which go into evaluating the relative merits of a particular match. Formalizing and quantifying these factors will be the principle focus of this chapter.

Straight line segments are used to represent models and data in this thesis. Other geometric primitives, such as points, could be used. Most of what is developed in this thesis would be simpler with points than with line segments. However, mature algorithms exist for extracting straight line segments [89, 20], and using segments it is possible to represent a wide variety of objects.

The discrete space of possible matches is described by a set of possible correspondence mappings between model and data line segments. In this thesis, the mapping is defined in the least restrictive way possible: any number of model segments may map to (i.e. match) a data segment, and any number of data segments may map to a model segment. It is possible to preclude some pairs of segments from consideration if there is reason to believe such pairings are impossible. Section 3.2 more formally defines the space of correspondence mappings and motivates the use of many-to-many mappings.

As mentioned, the key issue in this chapter is: 'What makes one match better than another?' In tackling this question, it is instructive to step through a fairly simple example. Figure 3.1a shows a model consisting of 3 line segments and an associated set of 7 data line segments. It should be clear at a glance that there are loosely speaking two matches in this data. The first is $((A,1),(B,2),(C,3),(C4))$ and the second $((A,5),(B,6),(C,7))$. In this illustration, the size (i.e. scale) of the model is assumed fixed.

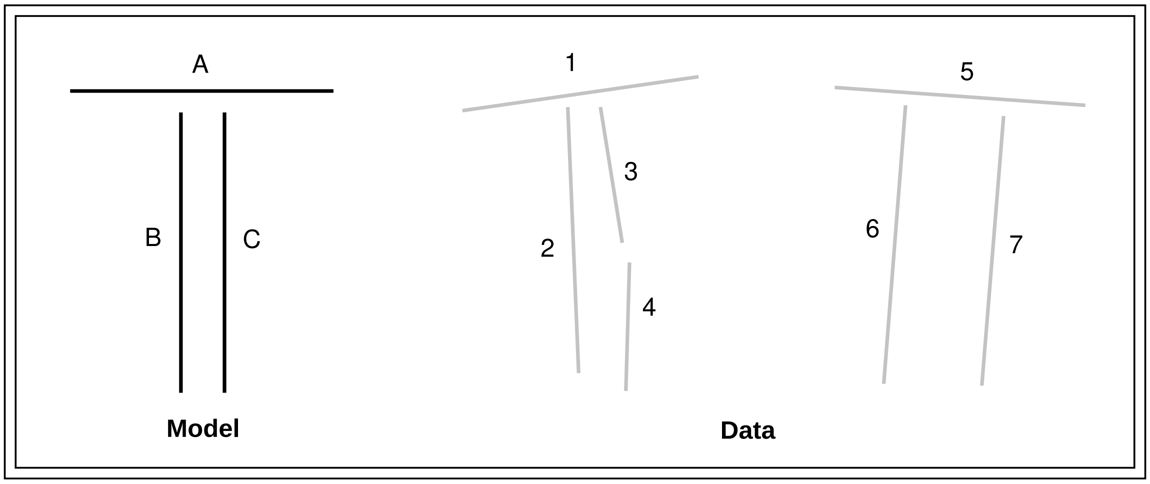


Figure 3.1 A simple model and data for illustrating match evaluation. Model segments are black and labeled with letters. Data segments are grey and labeled with numbers.

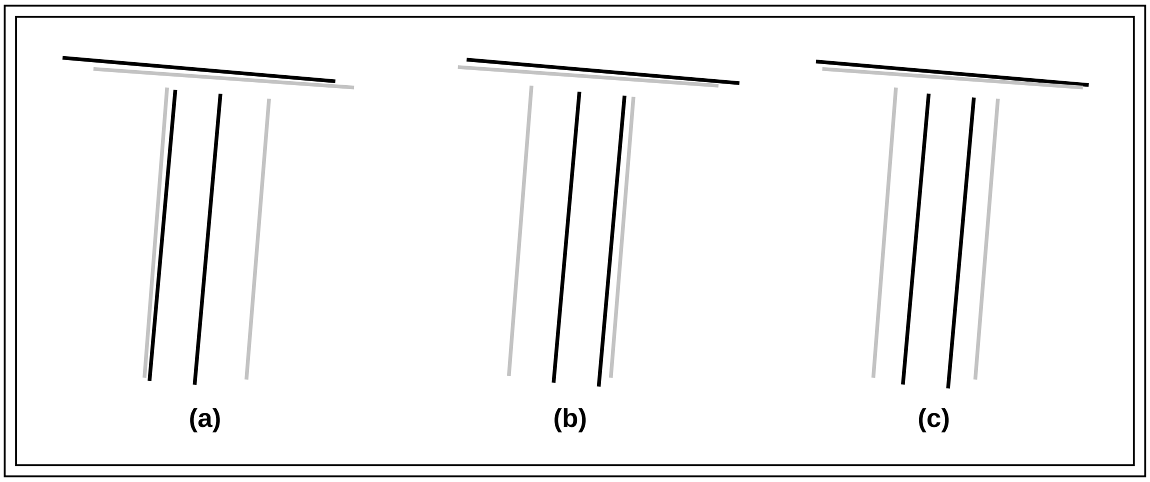


Figure 3.2 Plausible fits of the model to the data: a) favors left hand vertical segment, b) favors the right hand vertical segment, c) splits the difference.

It is virtually impossible to judge one better than the other without inferring something about the pose of the model implied by each. Qualitatively speaking, for correspondence $((A,1),(B,2),(C,3),(C4))$ there seems intuitively to be only one real option. However, correspondence $((A,5),(B,6),(C,7))$ is ambiguous, and Figure 3.2 shows three arguably consistent placements of the model relative to the data. Of these three, the one in Figure 3.2c represents a compromise between exactly matching any of the three pairs of segments. In contrast, the first two favor closeness of fit between some pairs at the expense of others. This third placement is indicative of what is produced by least-squares fitting, and is the approach favored in this thesis. The problem with favoring some correspondence pairs at the expense of others, as exemplified in Figures 3.2a and 3.2b, is that it is essentially equivalent to dropping pairs from a match. Consequently, it no longer respects the initial presumption that all corresponding pairs participate in the match. For example, the fit shown in Figure 3.2a is more appropriately associated with the correspondence $((A,5),(B,6))$ than with $((A,5),(B,6),(C,7))$.

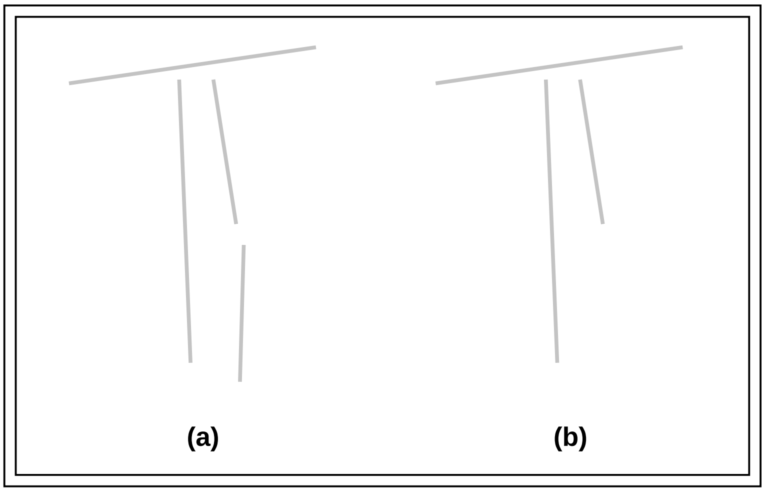


Figure 3.3 Two alternate matches illustrating omission: a) Four segments account for most of the model, b) dropping data segment 4 leaves a portion of the model unaccounted for. The loss of the segment makes this a poorer match.

As this example illustrates, fitting based upon a hypothesized correspondence is the necessary first step in evaluating the quality of a match. How well a model fits data is clearly one factor used to rank alternative correspondences. What may be less obvious, is that fit alone is an inadequate measure of match quality. Figure 3.3 shows the data for two almost identical matches, except in Figure 3.3b one data segment is omitted. The fit, in a least-squares sense, will be better in Figure 3.3b than in Figure 3.3a. What makes the data in Figure 3.3a a better match is that it better accounts for the entire model.

At a minimum, in order for a match error function to enforce the preferences expressed in these examples, it must account for both *fit* and *omission*. The exact manner in which each is formalized will differ given different assumptions, but the underlying reason for why each is needed should be apparent. The most obvious way to combine these two forms of error is to sum them. This leads to the essential form of match error used in this thesis:

$$E_{\text{match}} = \left(\frac{1}{\sigma^2}\right) E_{\text{fit}} + E_{\text{om}} \tag{3.1}$$

with the weighting coefficient σ controlling the relative importance of these factors.

3.2 A Space of Possible Matches

The least restrictive space of mappings is many-to-many: any number of data line segments may correspond to a model line, and any number of model line segments may correspond to a data line segment. A variety of constraints (e.g. approximate knowledge of a model's orientation) might preclude some pairs of model and data lines from matching, thus limiting the space of potentially paired segments. Such constraints come into play prior to, not during, matching.

Formally, let M be the set of model line segments and D be the set of data line segments. Let S be the subset of MxD containing pairs of model and data line segments which are candidate matches. Without prior constraints $S = MxD$. The possible mappings between model and data segments c belong to the space of correspondence mappings C:

$$C = 2^S \quad S \subseteq MxD. \tag{3.2}$$

Figure 3.4 shows how approximate knowledge of the model's placement might limit the search space. For example, data line segments $0, 1, 8$ and 11 are roughly parallel to model line segment A and might be considered close enough to be potential matches. Consequently, in this illustration the space of possible matches to model line segment A is restricted to pairs involving data segments $\{0, 1, 8, 11\}$.

There are several equivalent but distinct ways to denote a correspondence $c \in C$. One is to write out explicitly each pair s belonging to c. This form is assumed in equation 3.2. Another is to associate a unique bitstring with each $c \in C$. This may be done by defining C as a mapping from pairs S onto values $\{0, 1\}$:

$$C : S \rightarrow \{0, 1\}. \tag{3.3}$$

This bitstring interpretation is illustrated in Figure 3.4. In particular, the bitstring for the best match is shown. Finally, a tabular form such as shown in Figure 3.4 is convenient and will often be used in illustrations.

Data line segments are presumed to fragment often, as illustrated in Figure 3.4. Therefore, correspondence mappings must be one-to-many: one model line segment mapping to two or more data line segments. Examples in Chapter 1 illustrated that the converse, mapping one data line segment to two model line segments, is also important. Figure 3.5 provides another example. A curve is approximated by a sequence of straight line segments. Typically, the polygonal approximation used to describe the model will not be identical to that produced by a piecewise linear data extraction process; the exact points at which the curve is broken into successive segments are almost certain to differ. Hence, as is shown in Figure 3.5, one model line correctly maps to two data line segments, just as one data line segment also correctly maps to two model line segments.

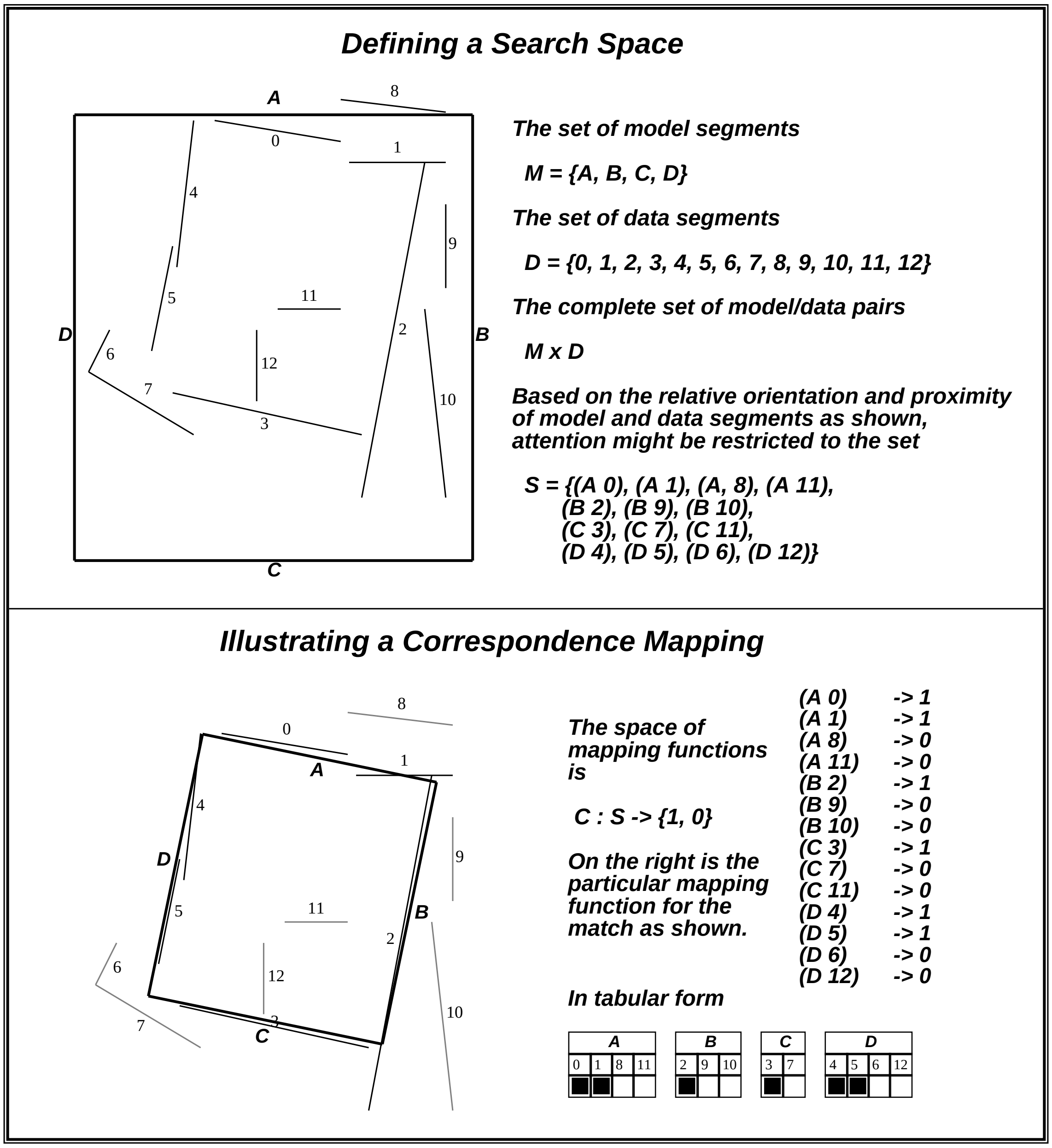


Figure 3.4 Illustrating search spaces and correspondence functions. Orientation and proximity can constrain the set of potentially matching pairs of model and image segments. The top figure shows the model in a pose expected to be roughly correct. This pose constrains the set of potentially matching pairs contained in the set S. The bottom figure illustrates the correspondence mapping for the best match.

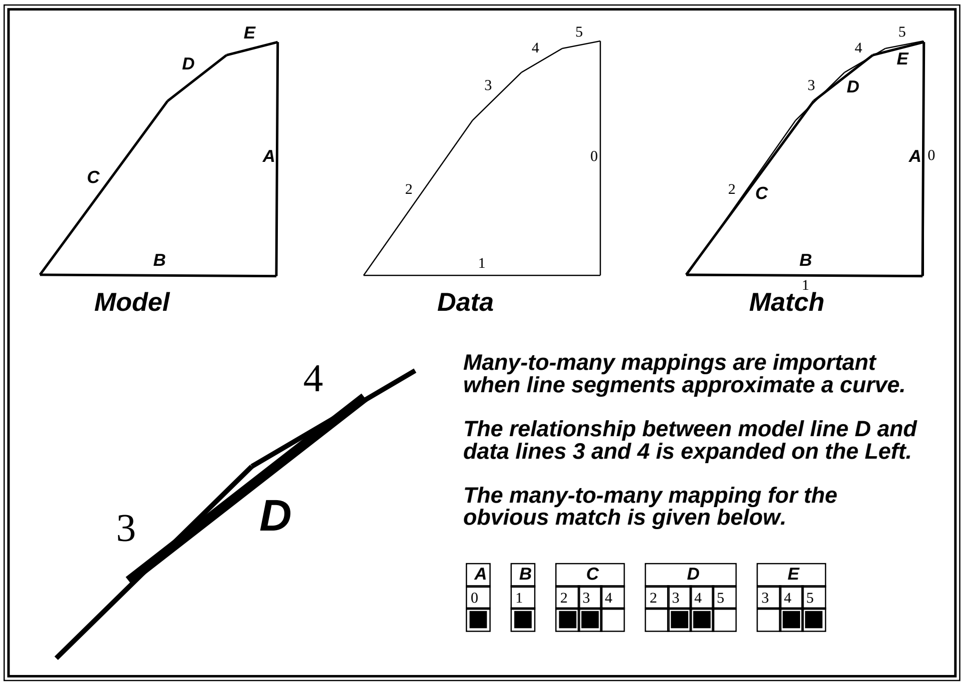


Figure 3.5 True many-to-many mappings do arise. In particular, when a curve is approximated with a series of straight line segments, the resulting mapping is likely to be many-to-many.

3.3 Fit Error and Fitting

Perhaps the most obvious way one might think of fitting a model to corresponding data would be to minimize a sum of squared distances between corresponding points on the model and the data. However, in part because data line segments fragment, the use of such a point-to-point measure for fitting is inappropriate.

To get around the problem of not having unique and reliably placed points, others have realized the importance of using point-to-line (2D) or point-to-plane (3D) measures [80, 3, 70] to accomplish fitting. This chapter will focus on the 2D case, measuring squared distance from points to lines in the 2D image plane. The 3D case is left for Chapter 7. For 2D fitting, Ayache [3] proposed the best-fit between a model and corresponding data segments should minimize the sum of squared perpendicular distance between model segment midpoints and infinitely extended data lines.

This thesis introduces a new fitting criterion, the *integrated squared perpendicular distance* (ISPD) between data segments and infinitely extended model lines. As will be explained in Chapter 4, this measure more reliably recovers the true pose of a model given fragmented data segments. As the name implies, ISPD is obtained by taking the measure of squared perpendicular points on a data segment and a corresponding model line and integrating this measure between the endpoints of the data segment. The details of how models are fit to data using ISPD are reserved for Chapter 4. This chapter will formally define ISPD and show how ISPD may be normalized with respect to the size of a model in order to produce the fit error function E_{fit}.

3.3.1 Integrated Squared Perpendicular Distance (ISPD)

The perpendicular distance v between a model line A and data segment 1 is illustrated in Figure 3.6. The reader should note that the model segment A has been extended into an infinite line, and that perpendicular distance is measured with respect to this infinite line. This is done for reasons of computational expediency, since it facilitates the development of general analytic closed-form solutions to the problem of finding the best-fit model pose.

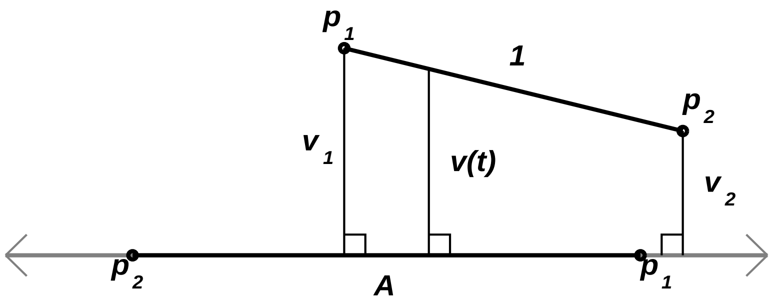


Figure 3.6 Points on data segment 1 project perpendicularly onto model line A. The perpendicular distance at any point along 1 may be written as a function $v(t)$.

The first step in deriving the ISPD is to write the perpendicular distance v as a function of position along the segment. This may be done by defining a parametric from $v(t)$:

$$v(t) \;=\; v_1 + (v_2 - v_1)\,\frac{t}{\ell} \qquad\qquad 0 \leq t \leq \ell. \tag{3.4}$$

The parameter t is the position on the data segment, v_1 and v_2 are the perpendicular distances from endpoints 1 and 2 respectively, and ℓ is the length of the data segment.

The definite integral of $v(t)^2$ over the length of the data segment has a relatively simple form:

$$\mathrm{ISPD} \;=\; \int_0^{\ell} v^2\,(t)\,dt \;=\; \frac{\ell}{3}\left(v_1^2 + v_1 v_2 + v_2^2\right). \tag{3.5}$$

Figure 3.7 illustrates fitting using ISPD in order to provide some intuition for how a model appears when fit to data, and how the residual perpendicular error appears visually as a perpendicular displacement between matched segments. Model segments are drawn in black. Data segments matched to model segments are next darkest, followed by extensions indicating the infinite line associated with each model segments. Perpendiculars dropped from endpoints of data segments to corresponding model lines are drawn in still a lighter shape of grey. Finally, unmatched data segment are drawn in the lightest shade of grey.

The correspondence in Figure 3.7a is the correct or best correspondence. Consequently, the fit is quite good. It is in fact somewhat difficult to see the grey line segments indicating perpendicular displacement between data segments and corresponding model lines because the model is lying essentially on top of the corresponding data segments. The perpendicular components are more easily seen in Figure 3.7b, in which an incorrect or sub-optimal pairing has been added to the match. The vertical data segment in the lower right hand corner has been matched to the right side of the rectangle, causing the rectangle to rotate relative to

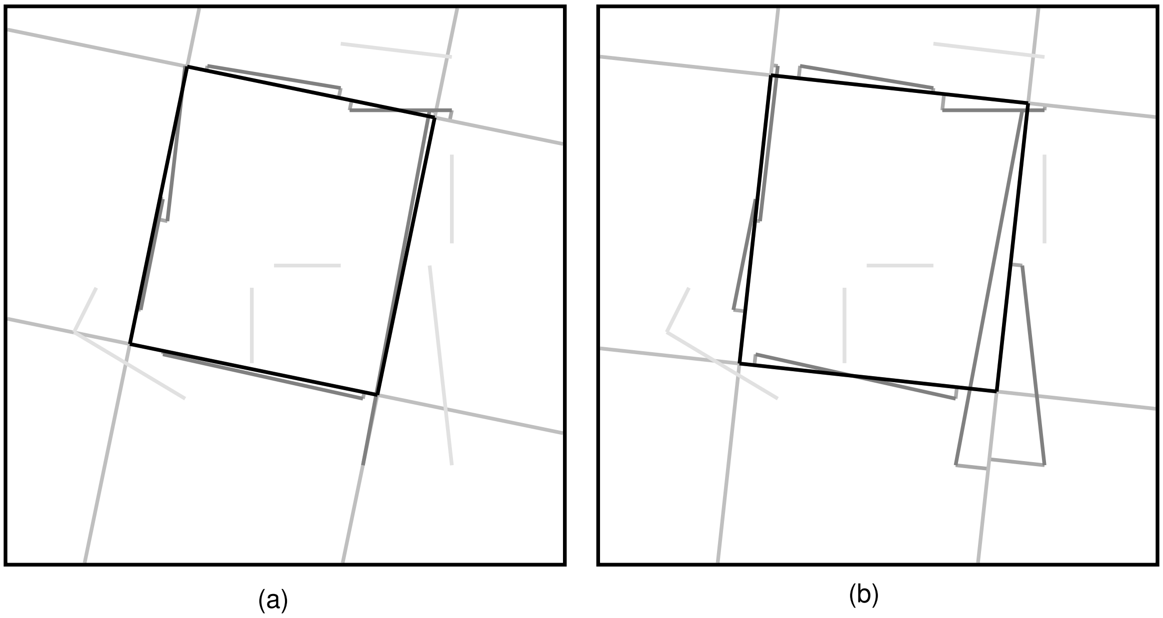


Figure 3.7 Illustrating how perpendicular error is used to measure fit. The error is indicated by lines drawn from endpoints of data segments to extended model lines. Observe how in (b) the addition of one 'incorrect' pairing changes the model's pose.

the pose in Figure 3.7a. Additionally, the rectangle has changed position and size slightly, but these changes are not visually obvious.

There is an alternative to ISPD which should be mentioned, and it is the weighted squared perpendicular distance from data segment endpoints. It is important to understand that this weighted endpoint perpendicular distance (WEPD) measure offers little advantage in terms of algebraic simplicity and that it introduces an undesirable endpoint bias. Consider the algebraic form of these two measures:

$$\text{ISPD} \;=\; \frac{\ell}{3}\left(v_1^2 + v_1 v_2 + v_2^2\right), \qquad \text{and} \qquad \text{WEPD} \;=\; \frac{\ell}{2}\left(v_1^2 + v_2^2\right). \tag{3.6}$$

These two fit measures differ only with respect to the cross term $v_1 v_2$ and the associated division of the sum by 3 rather than 2.

The endpoint bias of the WEPD measure is due to the fact that it is not invariant under fragmentation of data segments. If a single data segment is broken into two contiguous segments, the union of the sets of points lying on each fragment equals the set of points lying on the original. Consequently, the sum of the ISPD over the two fragments is indistinguishable from the ISPD computed over the original segment. However, this is not true of the WEPD measure, which biases fit based upon specific endpoint placement. Because it is invariant with respect to fragmentation, ISPD is used in this thesis.

3.3.2 Fit Error: Normalized ISPD

In principle, the residual ISPD after fitting could be used directly as a fit error function E_{fit}. However, for reasons discussed in this section, it is desirable to normalize the ISPD measure with respect to model size. One reason is so the match error function will produce comparable values for models of different size. Another reason is that normalization will simplify the task of combining E_{fit} with the omission error E_{om}.

Consider ISPD summed over all pairs of corresponding line segments $s \in c$:

$$\text{ISPD}(c) \;=\; \sum_{s \in c} \text{ISPD}(s) \tag{3.7}$$

where ISPD(s) is defined by equation 3.5 for a pair of segments s. To define a fit error, ISPD(c) might be normalized either by the sum of the lengths of the model line segments or the sum of the lengths of the data line segments. The sum of the model segment lengths may be written as:

$$L_m \;=\; \sum_{m \in M} \ell(m) \tag{3.8}$$

where $\ell(m)$ is the length of the model segment m **after** the model M has been fit to the corresponding data. The sum of the lengths data segments may be written as:

$$L_d \;=\; \sum_{m \in M} \sum_{d \in s_m} \ell(d) \tag{3.9}$$

where $\ell(d)$ is the length of a data segment d, s_m is the subset of the correspondence mapping c including model segment m, and $d \in s_m$ is understood to mean the data segments included in the pairs s_m.

Model-normalized and data-normalized fit errors may be defined as:

$$E^m_{\text{fit}} \;=\; \frac{1}{L_m}\text{ISPD}(c) \qquad \text{and} \qquad E^d_{\text{fit}} \;=\; \frac{1}{L_d}\text{ISPD}(c). \tag{3.10}$$

The principle difference between model-normalized fit error E^m_{fit} and data-normalized fit error E^d_{fit} concerns what happens when too many data segments are included in the correspondence mapping. To illustrate, if the total length of data line segments L_d is five times larger than the total length of the model line segments L_m, then E^m_{fit} will be five times larger than E^d_{fit}. Consequently, E^m_{fit} is somewhat better at penalizing such matches, and this in turn assists local search in rejecting such matches.

The data-normalized error E^d_{fit} is potentially simpler to compute than E^m_{fit} because the sum of the lengths of the model line segments changes with the best-fit pose. When fitting subject to rotation, translation and scaling in the image plane, adjusting L_m turns out to be trivial: simply apply the scale change to the sum of model line lengths measure in model coordinates. However, for full-perspective matching developed later in this thesis, recomputing L_m for every new best-fit pose is more complicated, and the simpler data normalized form E^d_{fit} is used.

The differences between E^d_{fit} and E^m_{fit} are illustrated by example in Table 3.1. In this table, the exact values for the two forms of fit error are shown broken out by model line segment. The segment labels are the same as shown in Figure 3.4 (page 48). Tables 3.1a and 3.1b are for the matches shown in Figures 3.7a and 3.7b respectively.

The key thing to observe is that for the very good match, Figure 3.7a, the difference between the two measures is negligible. However, for the match in Figure 3.7b, the total

Table 3.1 Comparing the fit errors for correspondences in Figure 3.7. Both model-normalized E^m_{fit} and data-normalized E^d_{fit} fit errors are shown. In addition, model segment length $\ell(m)$, number of data segments matched to the model segment $|s_m|$, cumulative length of matched data segments $\sum_{d \in s_m} \ell(d)$, ISPD, and average perpendicular distance PD are shown. Average PD is the square root of ISPD divided by $\sum_{d \in s_m} \ell(d)$. a) For the correspondence shown in Figure 3.7a. b) For the correspondence shown in Figure 3.7b.

Correspondence Figure 3.7a							
m	E^d_{fit}	E^m_{fit}	$\ell(m)$	$\|s_m\|$	$\sum_{d \in s_m} \ell(d)$	ISPD	PD
A	0.34	0.34	45.48	2	42.73	65.52	1.2384
B	0.15	0.15	51.07	1	65.12	28.69	0.6639
C	0.16	0.16	45.48	1	36.88	30.71	0.9126
D	0.23	0.23	51.07	2	48.58	44.27	0.9547
totals	0.88	0.88	193.11	6	193.30	169.21	–

(a)

Correspondence Figure 3.7b							
m	E^d_{fit}	E^m_{fit}	$\ell(m)$	$\|s_m\|$	$\sum_{d \in s_m} \ell(d)$	$ISPD$	PD
A	0.60	0.70	46.57	2	42.73	138.05	1.7974
B	10.50	12.19	52.29	2	101.34	2409.72	4.8764
C	0.22	0.26	46.57	1	36.88	50.63	1.1718
D	0.28	0.32	52.29	2	48.58	63.43	1.1427
totals	11.60	13.46	197.71	7	229.52	2661.86	–

(b)

fit error for the data-normalized form is noticeably smaller than for the model-normalized form, and this is an example of the undesirable behavior mentioned above. This difference will become more pronounced if additional data segments are added to the correspondence.

Table 3.1 also serves to illustrate how fit error increases for sub-optimal matches. Observe that the total fit error, measured either way, jumps by more than an order of magnitude with the addition of the bad data segment, and that although the majority of this jump can be attributed directly to the portion computed for model segment B, because the model is fit to the data a whole, the error contribution from the other segments also increases.

3.4 Omission Error

The omission error for a particular model segment will be defined as a non-linear function of the percent of the model line unaccounted for by data. A point, or more precisely an interval on a model segment, is defined to be unaccounted for if no part of any corresponding data line segment projects onto it. To illustrate using Figure 3.6 (page 50), the right portion of model segment A is covered, or accounted for, by data segment 1. The covered portion is bounded on the left by the point at which data segment endpoint p_1 projects onto A, and on the right by endpoint p_2 of segment A.

The omission error will be non-linear because even under the best of circumstances, a small amount of omission is to be expected (e.g. the ends of lines are often difficult to extract). However, if large portions of a model segment are missing from the data, the estimated quality of the match should be substantially reduced. To illustrate, consider a perfect square. Four data line segments, each covering three quarters of a side, is preferable to three line segments completely covering three sides, even though the total amount of model uncovered is the same in the two cases.

For a single model segment m, let p be the fraction not covered by a point on a corresponding data segment. The non-linear function of p used to define omission error is:

$$E_{\mathrm{om}}(p) = \begin{cases} \frac{e^{\alpha p}-1}{e^{\alpha}-1} & \text{if} \quad \alpha \neq 0 \\ p & \text{otherwise} \end{cases} . \tag{3.11}$$

The fraction p lies in the range $[0, 1]$, and hence the omission error too lies in this range. The degree of non-linearity is controlled by α. However, the exact manner in which changing α changes the form of $E_{\mathrm{om}}(p)$ is less than obvious. It is helpful to introduce an auxilary parameter a, and then define α in terms of a.

$$\alpha = 2\ln\left(\frac{2}{a} - 1\right). \tag{3.12}$$

The parameter a may be thought of as *attenuation* because it attenuates the omission error relative to the linear case. Figure 3.8 illustrates how attenuation specifies the non-linearity of the relationship between p and E_{om}. The horizontal axis is p and the vertical axis E_{om}. In the special case where attenuation $a = 1.0$, E_{om} is a linear function of p. In fact, it is the identity function. As a drops below 1.0, E_{om} drops below the linear case. For instance, $a = 0.5$ attenuates E_{om} by 50% at the midpoint of the curve. This parameter allows the user to vary the effect of the degree of model omission for different task domains.

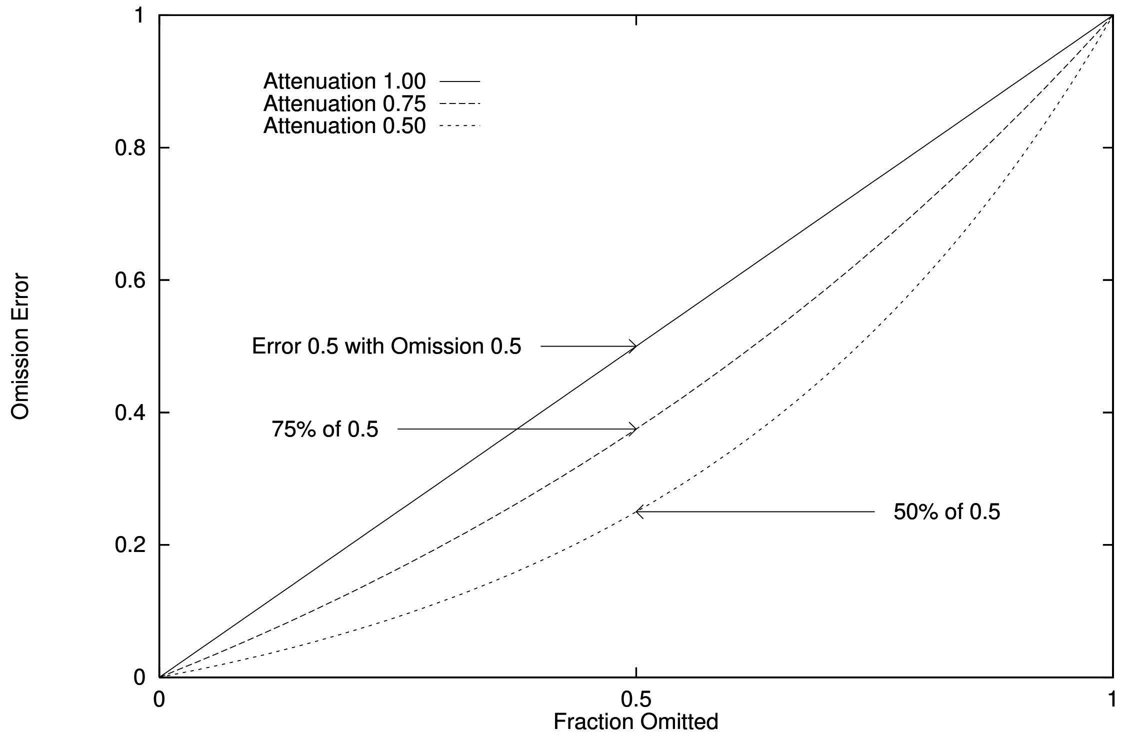


Figure 3.8 Different omission error curves. Attenuation a controls how the curve drops below the linear case. The three curves are for attenuation 1.00, 0.75 and 0.50. The corresponding values for α are 0.0, 1.0 and 2.2.

Thus far, $E_{\text{om}}(p)$ has only been defined for a single model segment m based upon the fraction of the segment omitted p. The omission error for a complete match is a weighted sum of the omission error for each of the model segments:

$$E_{\text{om}} = \sum_{m \in M} \left(\frac{\ell(m)}{L_m} \right) E_{\text{om}}(p_m). \tag{3.13}$$

Weighting the contribution from each model line segment by the relative length of the segment makes the omission error more directly comparable to the normalized fit error E_{fit}. However, it is possible to weight omission uniformly across model segments regardless of length. This emphasizes the importance of small segments, and an example where this is important is the telephone pole matching example in Section 5.4.3. Unless otherwise stated, the weighted by length form of the omission error will be used in this thesis.

3.5 Trading Off Fit Error Versus Omission Error

The balance struck between fit error and omission error defines the character of an optimal match. Recall from equation 3.1 that the relative importance of fit versus omission is controlled by the parameter σ. By weighting the relative importance of fit versus omission,

σ controls how far a data line segment may be displaced from a model segment and still be included in an optimal match. The fit error and omission error have been normalized in part to give σ an intuitive geometric interpretation: σ may be thought of as the *maximum-displacement* beyond which a data segment displaced relative to the model will be dropped from a match. The parameter σ may also be thought of as the standard deviation on an error process which displaces and skews data line segments relative to model segments. The effect is then to include in matches segments lying within one standard deviation of the model. Although this interpretation is approximately correct, making it rigorous is complicated, and it is not developed in this thesis.

The global fitting process used to register the model to the data complicates maximum-displacement interpretation for σ. Consider a set of data which is a perfect copy of a model. If a single component of the model is displaced by an amount σ, the best-fit pose of the model relative to the data will tend to distribute this error over all components of the model, and therefore the actual displacement will be less than σ. To better understand what is taking place, this section will study what happens when one component of a simple model is displaced relative to the rest of the model.

The relationship between fit and omission is better seen by substituting the model-normalized fit error E^m_{fit} from equation 3.10 and the omission error E_{om} from equation 3.13 into equation 3.1:

$$E_{\text{match}} = \left(\frac{1}{\sigma^2 L_m}\right) \sum_{s\in c} \text{ISPD}(s) + \sum_{m\in M} \left(\frac{\ell(m)}{L_m}\right) E_{\text{om}}(p_m) \tag{3.14}$$

$$= \left(\frac{1}{L_m}\right) \sum_{m\in M} \left(\left(\sum_{s\in s_m} \frac{\text{ISPD}(s)}{\sigma^2} \right) + \ell(m) E_{\text{om}}(p_m) \right). \tag{3.15}$$

To understand the consequences associated with matching or not matching a single model line segment, equation 3.15 breaks down the error by model segments $m \in M$.

Neglecting global fitting, the *break-even* point at which the cost of adding a pair $s' = (m, d)$ equals the penalty for omitting it is reached when the average perpendicular distance between d and m equals σ. To see this, consider the following scenario: there is one pair, $s' = (m, d)$, involving model segment m, and the data segment d completely covers model segment m. The contribution to E_{match} for model segment m with s' *included* in the correspondence consists solely of the fit error:

$$\frac{\text{ISPD}(s')}{\sigma^2}. \tag{3.16}$$

With s' *excluded* there is no fit error and $E_{\text{om}}(1.0) = 1.0$. Therefore, the contribution to E_{match} is

$$\ell(m). \tag{3.17}$$

The break-even point at which the match error is the same whether s' is included or excluded arises when equations 3.16 and 3.17 are equal, and this in turn implies that the break-even point is achieved when

$$\frac{ISPD(s')}{\ell(m)} = \sigma^2. \tag{3.18}$$

The expression on the left is the average squared perpendicular distance between m and d.

Therefore, taking the square root of each side shows that the break-even point is reached when the average perpendicular distance equals σ.

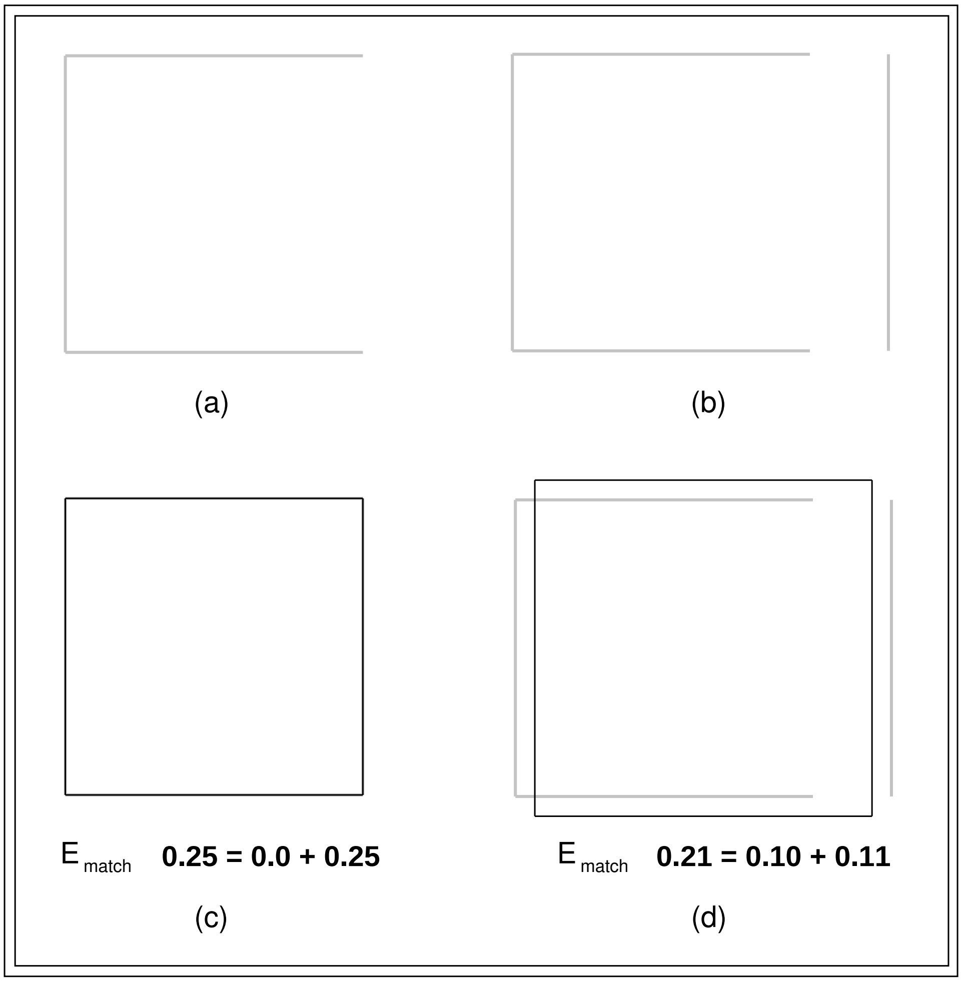


Figure 3.9 Global fitting modifies effect of displacing a single segment: a) three perfect sides of a square 30 units on a side, b) a fourth side displaced 8 units to the right, c) square matched data three sides, d) square matched to all four sides. The match error is written below the two matches: $E_{\text{match}} = \left(\frac{1}{\sigma^2}\right) E_{\text{fit}} + E_{\text{om}}$.

Figure 3.9 explores how fitting the model to the data as a whole changes the behavior just predicted. For a square of 30 units on a side, one side is displaced 8 units from its true position in the data, and then the match error is computed with $\sigma = 8$ both with and without the displaced side included in the match. The illustration shows how the effect of global fitting is to distribute and reduce the expected fit error.

Figures 3.9a and 3.9b show the data to which the square is matched both with and without the displaced side included in the match. Figures 3.9c and 3.9d show the model matched to each set of data, and under each the match error is shown as the sum of the fit error divided by σ^2 plus the omission error. In Figure 3.9c, the fit error is zero: the model lies exactly over the three data segments, and all the error comes from the omission of the one side. Figure 3.9d shows the effect of matching to the displaced side: the fit error increases and the omission error decreases. However, because of the global fitting, the fit error is distributed over all four segments and does not increase as much as the omission error decreases. Consequently, the match error is less with the displaced side included in the match.

3.6 Collateral Knowledge added to Match Error

A variety of additional forms of error might be added to the match error function, each reflecting a different constraint. Two such additional errors are considered here. The first is a measure of pairwise compatibility between corresponding segments. The second is a measure of the extent to which the best-fit pose - the transformation fitting the object model to the data - is consistent with initial expectations. The match error with these additional constraints added is:

$$E_{\text{match}}(c) = \left(\frac{1}{\sigma^2}\right) E_{\text{fit}}(c) + E_{\text{om}}(c) + E_{\text{pw}}(c) + E_{\mathcal{F}}(c) \tag{3.19}$$

where E_{pw} is a pairwise error term expressing pairwise compatibility between matching segments and $E_{\mathcal{F}}$ is a transformation error term expressing how well the best-fit pose conforms to the expected pose.

There is an obvious and a less obvious reason for adding constraints to the match error. The obvious reason is to alter the ranking of the locally optimal matches. The additional constraints modifie what constitutes the best match. The less obvious reason is that additional constraints modify the way local search explores the combinatorial space of possible correspondences. This is an important consideration, and will be discussed further after local search is introduced in Chapter 5. It should be said here that including the pairwise error defined below has proven to be essential in order to solve a difficult matching problem involving a highly symmetric object model.

3.6.1 Pairwise Error

Pairwise error, E_{pw}, is defined relative to a pair of corresponding model and data segments and provides an opportunity to take into account compatibility constraints between two matching segments. The pairwise error for a correspondence c is defined to be the sum of the pairwise error associated with corresponding pairs of segments:

$$E_{\text{pw}}(c) = \sum_{s \in c} E_{\text{pw}}(s). \tag{3.20}$$

The following is an example of a pairwise error function that has proven to be useful. This error penalizes segments whose relative orientations differ by more than a preselected threshold. The difference in relative orientations θ is measured after the model has been fit to the data:

$$E_{\text{pw}}(s) = \begin{cases} 0 & \text{if} \quad |\theta| < \theta_l \\ \frac{(\sin\theta)^2 - (\sin\theta_l)^2}{(\sin\theta_u)^2 - (\sin\theta_l)^2} & \text{otherwise} \end{cases} . \tag{3.21}$$

The threshold θ_l establishes a minimum difference in orientation, perhaps 10 degrees, below which the error is 0. The upper bound, θ_l, sets the slope of the error such that if the difference in relative orientation reaches this upper bound, then the error equals 1.0. If the error exceeds this upper bound, then it grows larger than 1.0. Figure 3.10 illustrates the shape of E_{pw} for $\theta_l = 10$ degrees and $\theta_u = 30$ degrees.

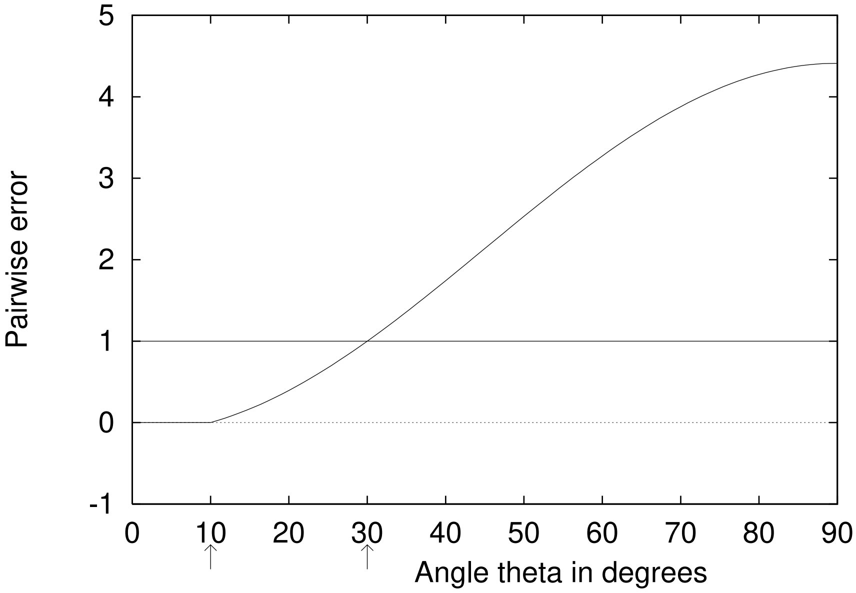


Figure 3.10 Example plot of relative orientation pairwise error. Error is zero up to $\theta_l = 10$ degrees, and then ramps up to 1.0 at $\theta_u = 30$ degrees. For θ greater than 30 degrees error continues to grow.

For reasons of computational expediency, this error is expressed in terms of sin and cos rather than directly in terms of the differences in angle. The squared sin of the angle between segments is readily computed by subtracting from 1.0 the square of the dot product between the normals of the two segments.

A variety of other pairwise constraints between potentially matching segments might be encoded using E_{pw}. For example, if appearance information were available for a domain, this might be encoded as a pairwise error; for example, a penalty might be exacted for matching a model segment expected to have a textured region on one side to an image segment with uniform regions on both sides.

3.6.2 Transformation Error

It has proven important when matching under weak-perspective to include an error term that discourages correspondences that imply near pathological transformations. In particular, the problem involves correspondences for which the change in scale becomes very large; for instance, a correspondence whose best-fit transformation shrinks the model by a factor of 10 or 100. In order to make these matches less desirable the following $E_{\mathcal{F}}(c)$ is defined:

$$E_{\mathcal{F}}(c) = \begin{cases} 1/s - r & \text{if} \quad s < 1/r \\ 0.0 & \text{if} \quad 1/r \leq s \leq r \\ s - r & \text{if} \quad s > r. \end{cases} \tag{3.22}$$

Here s is the scale change associated with the best-fit pose for correspondence c. The parameter r specifies an allowable range of scale changes s for which there is no penalty. If the scale change grows or shrinks beyond r then the error grows in proportion to the relative change in scale. For all the weak-perspective experiments presented in this thesis $r = 2$.

CHAPTER 4

Fitting Under Weak-Perspective

4.1 Introduction

Closed form equations for fitting 2D models to data are derived in this chapter. In particular, equations are developed for minimizing the integrated squared perpendicular distance (ISPD) measure introduced in the previous chapter. This approach to fitting 2D line segment models to 2D potentially fragmented 2D line segment data is superior to previous approaches to the problem.

Fitting 2D point models to corresponding 2D data points is simpler than fitting points to lines. For a rigid 2D point model fitting involves finding a rotation R^* and translation $\vec{T}^*$ in the image plane such that the sum of squared *Euclidean* distance between transformed model points and corresponding data points is minimized. If the 2D object model is permitted to change size, then an additional scale term s^* must also be determined. Solving for the best-fit rigid and similarity (variable scale) transforms for points is a simple and solved problem.

Unfortunately, when working with line segment models and potentially fragmented data line segments, it is unwise to use point-to-point fitting. It is, for example, difficult to determine corresponding points when a model line segment matches two data data line segments. As observed by others [80, 3], it is better to fit so as to minimize perpendicular point-to-line distance. Finding best-fit rigid and similarity transformations using point-to-line measures is a more difficult and less well studied problem.

When this thesis work began, the outwardly simpler problem of fitting models to data using 2D-rigid transformations was considered first. The best-fit rigid transformation was found to depend upon the roots of a quartic (fourth order) equation [11]. Subsequently, the variable scale problem was considered. In this problem the model is fit to the data using a 2D similarity transformation. For this problem it was found that the best-fit similarity transformation depends upon the roots of a quadratic equation. Therefore, fitting variable scale models in simpler than fitting rigid models, and the underlying reasons for why the rigid problem is more difficult are explored at the end of this chapter.

As already discussed in Chapter 1, the similarity transform is referred to here as weak-perspective because it approximates full-perspective for a subset of possible object views. For flat objects essentially facing the camera, the variability in appearance is essentially equivalent to taking a single projection from 3D-to-2D, and then applying 2D similarity transforms to this projection. The 2D similarity transform is also sometime called the 4-parameter affine transform.

A quadratic fitting result closely related to the one developed here was presented by Ayache [3] in 1986. This chapter begins by describing the fit measure and solution technique developed by Ayache [3]. Ayache measured error from midpoints of model segments to infinitely extended data lines, but as will be shown, it is better to measure from infinitely extended model lines.

4.2 Ayache: Minimizing Model Midpoint to Data Line Distance

The HYPER system [3] developed by Ayache and Faugeras recognized 2D parts placed flat upon a table. The approach taken was essentially one of first finding simple key-features and then confirming or rejecting hypothesized matches based upon the fit of the 2D model to corresponding image data. In particular, Ayache's models were comprised of 2D straight line segments and these were matched to line segments extracted from images. It is the details of the line segment fitting technique which are of interest here.

Ayache originally proposed fitting midpoints of model segments to midpoints of corresponding data segments, but found that because of imperfections in the data segments this introduced problems. Specifically, the fits were not reliable and consequently matches which ought to have been accepted were rejected. In response to the weakness of the point-to-point fitting, Ayache developed a fitting technique which minimized the squared perpendicular distance from infinitely extended data lines to the midpoints of model line segments.

Ayache's error measure may be written as

$$E = \sum_{i=1}^{n} \left(\hat{N}_i \cdot \left(s_{MD} R_{MD} \vec{M}_i - \vec{T}_{MD} \right) - \hat{N}_i \cdot \vec{D}_i \right)^2 , \tag{4.1}$$

where $\hat{N}_i$ is the unit normal for the ith data line segment, $\vec{D}_i$ is any point on the data line segment, and $\vec{M}_i$ is the midpoint of the ith model line segment. The optimal pose is determined by the scale change s^*_{MD}, rotation R^*_{MD} and translation $\vec{T}^*_{MD}$ which together minimize E. The rotation matrix R_{MD} and translation vector $\vec{T}_{MD}$ have the standard form:

$$R_{MD} = \begin{vmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{vmatrix} \qquad \vec{T}_{MD} = \begin{vmatrix} t_x \\ t_y \end{vmatrix}$$

Ayache showed that equation 4.1 may be rewritten as a simple quadratic in terms of an unconstrained vector $\vec{V}$. To build up to this, observe first that equation 4.1 may be rewritten as

$$E = \sum_{i=1}^{n} \left(\hat{N}_i \cdot \left(M_i \left(s_{MD} \vec{\Phi}_{MD} \right) - \vec{T}_{MD} \right) - \hat{N}_i \cdot \vec{D}_i \right)^2 \tag{4.2}$$

, where

$$M_i = \begin{vmatrix} x_i & -y_i \\ y_i & x_i \end{vmatrix} \qquad \vec{\Phi}_{MD} = \begin{vmatrix} \cos\phi \\ \sin\phi \end{vmatrix}. \tag{4.3}$$

The product $s_{MD}\vec{\Phi}_{MD}$ is a vector with two elements and two degrees of freedom. Hence, it is possible to further reduce equation 4.2 to a simple quadratic:

$$E = \sum_{i=1}^{n} \left(\hat{N}_i^{\mathrm{T}} C_i \vec{V} - \hat{N}_i^{\mathrm{T}} \vec{D}_i \right)^2 , \tag{4.4}$$

where

$$C_i = \begin{vmatrix} x_i & -y_i & 1 & 0 \\ y_i & x_i & 0 & 1 \end{vmatrix} \qquad \vec{V} = \begin{vmatrix} s_{MD}\cos\phi & s_{MD}\sin\phi & t_x & t_y \end{vmatrix}^{\mathrm{T}}.$$

The simplicity with which the optimal pose may be found using equation 4.4 is attractive, but unfortunately the measure as defined has several drawbacks. First, measuring perpendicular distance only from the midpoint of a segment does not constrain model segment orientation relative to a data segment. Second, infinitely extending the potentially fragmented data segments accentuates errors associated with skewed fragments. Each of these failings, along with the associated fixes, are described in the following section.

4.3 Minimizing Data Endpoint to Model Line Distance

4.3.1 Defining the Measure

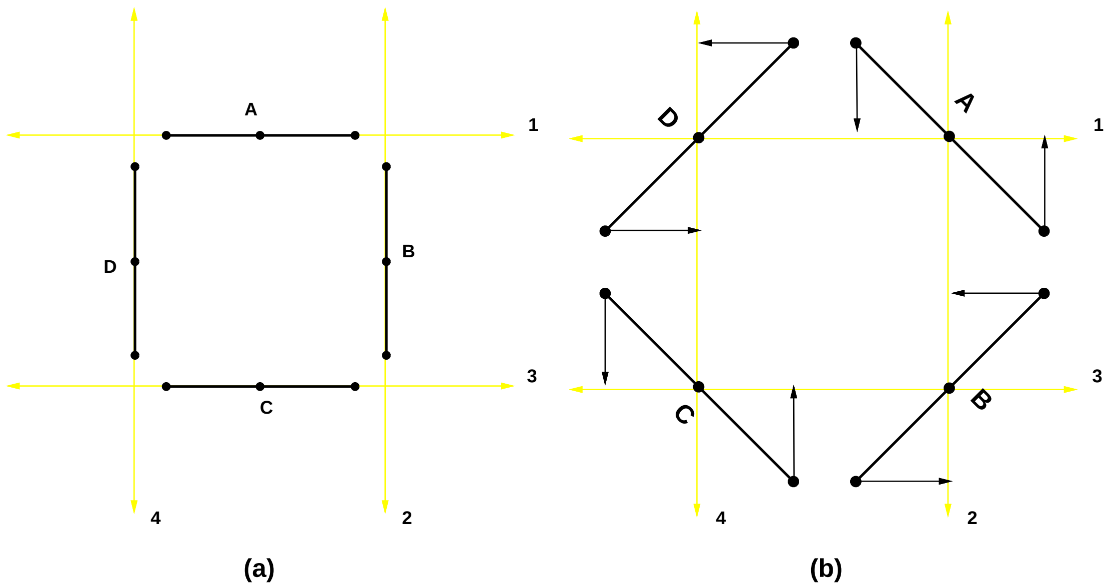


Figure 4.1 Measuring perpendicular distance from midpoints is inadequate. Measuring from endpoints is better. Four infinitely extended data lines are numbered 1 through 4. The model segments are lettered A through D. Model segment A matches to data line 1, segment B to 2, etc.

There are problems with Ayache's fitting based upon minimizing the perpendicular distance from model midpoints to infinitely extended data segments. A striking example of potential difficulties is presented in Figure 4.1, in which a model of a square is matched to perfect data. The model segments are shown in black, and infinitely extended data segments in grey. Using the midpoint measure proposed by Ayache, there are an infinite set of rotations and scalings that are equivalent. Figures 4.1a and 4.1b show two different fits, each with zero midpoint-to-midpoint error. Using perpendicular distance to endpoints rather than midpoints resolves this ambiguity.

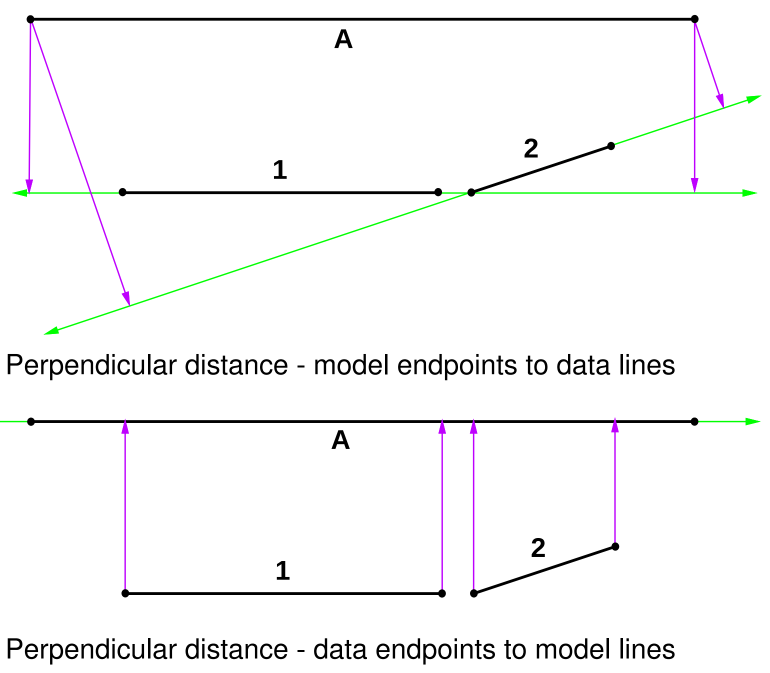


Figure 4.2 Perpendicular distance measured both from data and from model. A model line segment A and data line segments 1 and 2 are shown. Infinitely extending data line 2 exaggerates the perpendicular error between the left endpoint of A and the extended line. Extending model segments rather than data segments corrects this problem, since model segments do not fragment.

As illustrated in Figure 4.2, infinitely extending fragmented data line segments can bias the fit, and it is better to infinitely extend the model segments. Unlike data segments, model segments don't fragment. The resulting perpendicular data endpoint to model line fit measure is also shown in Figure 4.2. The skewing of segment 2 is not exaggerated using this more stable measure. A Monte Carlo study comparing these two measures confirmed that the data endpoint to model line measure more accurately recovers an object model's true pose when data segments are fragmented and skewed [11].

Given Ayache's derivation above, the most obvious way of writing the data endpoint to model line measure is to reverse the role of model and data segments in equation 4.1, and sum over both endpoints of the n corresponding segments:

$$E = \sum_{i=1}^{n} \sum_{j=1}^{2} \left(\hat{N}_i \cdot \left(s_{DM} R_{DM} \vec{D}_{ij} - \vec{T}_{DM} \right) - \hat{N}_i \cdot \vec{M}_{ij} \right)^2 . \quad (4.5)$$

$\hat{N}_i$ is now the unit normal to the ith *model* line segment, $\vec{M}_{ij}$ is the jth endpoint of the ith model line segment, and $\vec{D}_{ij}$ is the jth endpoint of the ith *data* line segment. Unfortunately, this measure computes this error subject to scaling, rotating and translating the data with respect to the model. To reflect this change, the subscript on the transformation has been changed from MD (Model-transformed-to-Data) to DM (Data-transformed-to-Model).

The transformation s^*_{DM}, R^*_{DM} and $\vec{T}^*_{DM}$ minimizing equation 4.5 best-fit registers the data to the model. However, in model matching, it is more natural to register the model to the data, and therefore the transformation s^*_{MD}, R^*_{MD} and $\vec{T}^*_{MD}$ transforming the model so as to minimize the perpendicular error measure is desired. Here is the same perpendicular error measure as in equation 4.5, but parameterized in terms of the model being transformed with respect to the data:

$$E = \sum_{i=1}^{n} \sum_{j=1}^{2} \left(\left(R_{MD} \hat{N}_i \right) \cdot \vec{D}_{ij} - \left(R_{MD} \hat{N}_i \right) \cdot \left(s_{MD} R_{MD} \vec{M}_{ij} + \vec{T}_{MD} \right) \right)^2 . \tag{4.6}$$

The perpendicular error is not invariant with respect to changes in scale, and therefore the transformation minimizing equation 4.6 is not the inverse of that which minimizes equation 4.5. The latter equation is appropriate for model matching.

By the construction already shown above, it should be apparent that minimizing equation 4.1, and hence equation 4.5, is straight forward. However, this same construction is not directly applicable to equation 4.6. A somewhat more involved construction will show that minimizing equation 4.6 reduces to the problem of solving a 2x2 eigenvector problem.

To see the reduction to an eigenvector problem, begin by noting that the inner product of vectors $\hat{N}_i$ and $\vec{M}_{ij}$ is invariant under rotation R_{MD}. Therefore

$$\left(R_{MD} \hat{N}_i \right) \cdot \left(s_{MD} R_{MD} \vec{M}_{ij} \right) = s_{MD} \left(\hat{N}_i \cdot \vec{M}_{ij} \right) = s_{MD} \rho_i$$

where ρ_i is the shortest distance from the ith model line to the origin.

It is worth emphasizing that from the standpoint of computing the best-fit transformation, the terms $\hat{N}_i$, $\vec{M}_{ij}$ and $\vec{D}_{ij}$ are constants defined by the initial geometry of the model and data segments. Hence, ρ_i depends only upon the initial placement of the model segment in the model's coordinate system.

Equation 4.6 may now be rewritten as:

$$E = \sum_{i=1}^{n} \sum_{j=1}^{2} \left(\left(R_{MD} \hat{N}_i \right) \cdot \left(\vec{D}_{ij} - \vec{T}_{MD} \right) - s_{MD} \rho_i \right)^2 . \tag{4.7}$$

Equation 4.7 may be simplified by observing that

$$\left(R_{MD} \hat{N}_i \right) \cdot \left(\vec{D}_{ij} - \vec{T}_{MD} \right) = \hat{N}_i \cdot \left(R_{DM} \vec{D}_{ij} + \vec{T}_{DM} \right),$$

where

$$R_{DM} = R^{-1}_{MD} \qquad \vec{T}_{DM} = -R_{DM} \vec{T}_{MD}.$$

Equation 4.7 may now be rewritten as:

$$E = \sum_{i=1}^{n} \frac{\ell_i}{2} \sum_{j=1}^{2} \left(\hat{N}_i \cdot \left(R_{DM} \vec{D}_{ij} + \vec{T}_{DM} \right) - s_{MD} \rho_i \right)^2 . \tag{4.8}$$

A weighting coefficient has been added to the perpendicular error terms in equation 4.8. The squared perpendicular distances for each pair i is multiplied by $\frac{\ell_i}{2}$, where ℓ_i is the length of the data line segment. Length is divided by two in order to take the average between the two endpoints. Weighting by length is important in order to prevent small fragments from exerting a disproportionate influence over the fit of the model to the data.

There is a helpful geometric interpretation for equation 4.8. The data is subjected to a rigid transformation, R_{DM} and $\vec{T}_{DM}$, while the model is scaled by s_{MD}. Since the sum of squared distances error measure is invariant under rotation and translation alone, and because the crucial best-fit scale is being computed subject to scaling the model, an equivalent best-fit pose may be obtained by subjecting the model to the transformation R_{MD}, T_{MD} and s_{MD} where

$$R_{MD} = R^{-1}_{DM} \qquad \vec{T}_{MD} = -R_{MD}\vec{T}_{DM}. \tag{4.9}$$

4.3.2 Finding the Best-fit Similarity Transform

Solving directly for the rotation R^*_{DM}, translation $\vec{T}^*_{DM}$ and scale s^*_{MD} which together minimize equation 4.8 is complicated by the presence of the rotation matrix R_{DM}. The task becomes simpler if R_{DM} is replaced with a rotation vector $\vec{\Phi}_{DM}$ analogous to $\vec{\Phi}_{MD}$ in equation 4.3. In consort with this change, it is necessary to replace the vector $\vec{D}_{ij}$ with the matrix D_{ij}. With these changes, equation 4.8 may be rewritten as:

$$E = \sum_{i=1}^{n} \frac{\ell_i}{2} \sum_{j=1}^{2} \left(\hat{N}_i \cdot \left(D_{ij}\vec{\Phi}_{DM} + \vec{T}_{DM} \right) - s_{MD}\rho_i \right)^2 \qquad \text{where} \qquad D_{ij} = \begin{vmatrix} x_{ij} & -y_{ij} \\ y_{ij} & x_{ij} \end{vmatrix}. \tag{4.10}$$

Multiplying out the terms in equation 4.10 and collapsing the sums yields the following:

$$E \;=\; \vec{\Phi}^{\mathrm{T}} A \vec{\Phi} + 2\vec{T}^{\mathrm{T}} B \vec{\Phi} + \vec{T}^{\mathrm{T}} C \vec{T} - 2\vec{U}^{\mathrm{T}} \vec{\Phi} s - 2\vec{V}^{\mathrm{T}} \vec{T} s + k s^2, \tag{4.11}$$

$$\begin{array}{rclrcl} A &=& \sum_{i=1}^{n} \sum_{j=1}^{2} (\ell_i/2)\, D_{ij}^{\mathrm{T}} \hat{N}_i \hat{N}_i^{\mathrm{T}} D_{ij} & \vec{U} &=& \sum_{i=1}^{n} \sum_{j=1}^{2} (\ell_i/2)\, \rho_i D_{ij}^{\mathrm{T}} \hat{N}_i \\ B &=& \sum_{i=1}^{n} \sum_{j=1}^{2} (\ell_i/2)\, \hat{N}_i \hat{N}_i^{\mathrm{T}} D_{ij} & \vec{V} &=& \sum_{i=1}^{n} \ell_i \rho_i \hat{N}_i \\ C &=& \sum_{i=1}^{n} \ell_i \hat{N}_i \hat{N}_i^{\mathrm{T}} & k &=& \sum_{i=1}^{n} \ell_i \rho_i^2 \end{array} . \tag{4.12}$$

For simplicity the subscripts have been dropped from the transformation terms: $s = s_{MD}$, $\vec{\Phi} = \vec{\Phi}_{DM}$ and $\vec{T} = \vec{T}_{DM}$.

To compute the best-fit 2D pose, one must solve for s^*, $\vec{\Phi}^*$ and $\vec{T}^*$ which minimize equation 4.11 subject to the constraint that

$$\vec{\Phi}^{\mathrm{T}}\vec{\Phi} = 1. \tag{4.13}$$

To solve for s^*, $\vec{\Phi}^*$ and $\vec{T}^*$, first set to zero the derivative of equation 4.11 with respect to s and solve for s^* as a function of $\vec{\Phi}$ and $\vec{T}$.

$$s^* \;=\; \left(\vec{U}^{\mathrm{T}}\vec{\Phi} + \vec{V}^{\mathrm{T}}\vec{T} \right) / k \tag{4.14}$$

Substituting s^* for s in equation 4.11 yields a new equation:

$$E = \vec{\Phi}^{\mathrm{T}} D \vec{\Phi} + 2\vec{T}^{\mathrm{T}} E \vec{\Phi} + \vec{T}^{\mathrm{T}} F \vec{T}, \tag{4.15}$$

where

$$D = A - \vec{U}\vec{U}^{\mathrm{T}}/k \qquad E = B - \vec{V}\vec{U}^{\mathrm{T}}/k \qquad F = C - \vec{V}\vec{V}^{\mathrm{T}}/k. \tag{4.16}$$

Next set the partial derivative of equation 4.15 with respect to $\vec{T}$ equal to zero and solve for $\vec{T}^*$ as a function of $\vec{\Phi}$:

$$\vec{T}^* = -F^{-1} E \Phi. \tag{4.17}$$

Finally, substitute $\vec{T}^*$ into equation 4.15 to get

$$E = \vec{\Phi}^{\mathrm{T}} G \vec{\Phi} \qquad \text{where} \qquad G = D - E^{\mathrm{T}} F^{-1} E. \tag{4.18}$$

By Rayleigh's Principle ([91], pg. 429) the vector $\vec{\Phi}^*$ which minimizes equation 4.18 is the unit eigenvector associated with the lesser eigenvalue of the matrix G.

The geometric intuition underlying this result is quite simple. Note that equation 4.18 is quadratic and hence $E \geq 0$ for any choice of vector $\vec{\Phi}$. Therefore, G is positive definite and the level curves of the error function are ellipses. Moreover, since there are no first order terms involving $\vec{\Phi}$ in equation 4.18, the ellipses are centered at the origin. Were the choice of $\vec{\Phi}$ unconstrained, then $\vec{\Phi} = 0$ would minimize equation 4.18. However, remember from equation 4.3 that $\vec{\Phi}$ is defined to be a vector formed by $\sin\phi$ and $\cos\phi$. Hence, only $\vec{\Phi}$ vectors which lie on the unit circle correspond to valid rotations, and the goal is that vector $\vec{\Phi}^*$ on the unit circle at which equation 4.18 obtains a minima.

Why the solution turns out to be the unit eigenvector associated with the lesser eigenvalue of the matrix G is illustrated in Figure 4.3. The level curves of the perpendicular error, as a function of the components of the $\vec{\Phi}$ vector, are ellipses centered about the origin. The constraint that $\vec{\Phi}$ be a unit vector restricts the choice of components to the the unit circle. The points on the unit circle with the lowest error are the two points at which the largest ellipsoidal level curve to fit entirely within the circle meets and is tangent to the unit circle. The eigenvectors of G are aligned with the minor and major axes of the ellipsoidal error curves, and it turns out that the unit eigenvector associated with the lesser eigenvalue falls on the unit circle at the desired point of lowest error. Therefore, this unit eigenvector is the rotation vector $\hat{\Phi}^*$ which minimizes the perpendicular error.

With $\hat{\Phi}^*$ known, the optimal translation $\vec{T}^*$ and scale s^* may be computed from equations 4.17 and 4.14 respectively. As Figure 4.3 suggests, there are in fact two vectors $\hat{\Phi}^*$, one for each of the two points where the unit circle and the major axis intersect. If the wrong $\hat{\Phi}^*$ is chosen then s^* will be negative. In this case, the sign on $\hat{\Phi}^*$, $\vec{T}^*$ and s^* can be flipped to obtain the desired transformation.

4.3.3 Symmetric 2x2 Eigensystems: The Lesser Vector

Finding the unit eigenvector associated with the lesser eigenvalue is conceptually simple. However, since this operation will be carried out millions of times by the matching algorithms developed later in this thesis, a practical guide to efficiently solving this problem will be

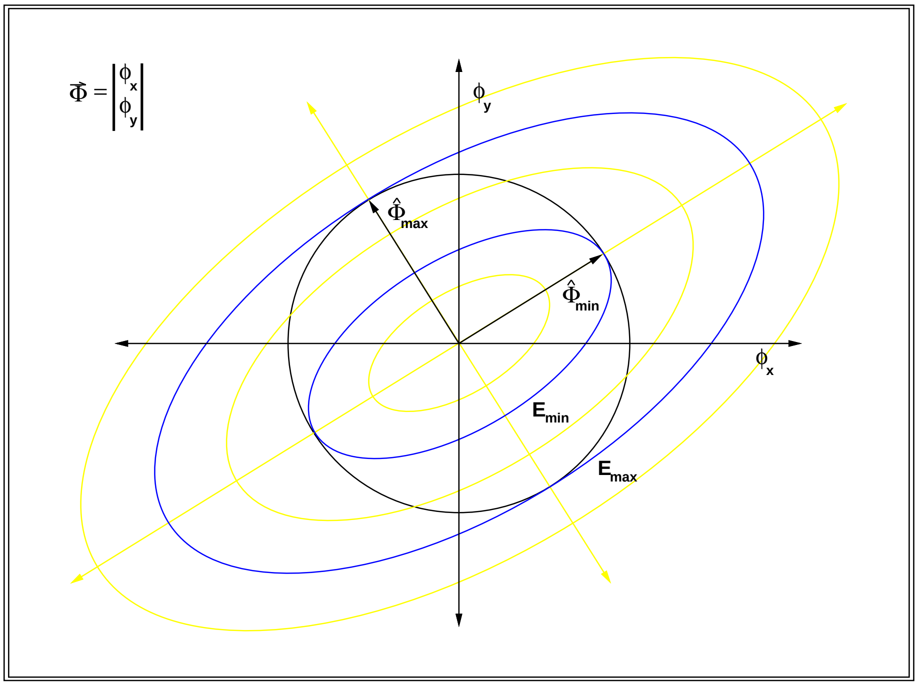


Figure 4.3 Illustrating elliptical error level curves. The optimal rotation vector $\hat{\Phi}^*$ is the point on the circle where the largest ellipsoidal error curve to fall within the circle contacts the circle.

helpful for readers interested in how the best-fit weak-perspective pose is most efficiently determined.

In a recent article, Newton [90] focuses on efficient ways of solving 2x2 eigenvector problems. The following is based upon the method presented in the article, but adapts the approach to the more restricted case of a 2x2 symmetric matrix. Remember that because the matrix is symmetric, the eigenvalues are both real though not necessarily finite.

Let the symmetric 2x2 matrix be

$$A = \begin{vmatrix} a & b \\ b & c \end{vmatrix}. \tag{4.19}$$

Newton observes that the paired eigenvalues and eigenvectors associated with matrix A may be expressed as:

$$\left[\; a + bm, \; \frac{1}{\sqrt{m^2+1}} \begin{vmatrix} 1 \\ m \end{vmatrix} \; \right] \tag{4.20}$$

for each of the roots, m, of the quadratic equation

$$bm^2 + (a - c)\,m - b = 0. \tag{4.21}$$

When $b \neq 0$ this may be simplified as:

$$m^2 + em - 1 = 0 \qquad \text{where} \qquad e = (a - c)/b. \tag{4.22}$$

Newton gives special consideration to the case of $b = 0$. In particular, one root of Equation 4.22 is defined as ∞. In our special case in which A is symmetric, it further follows that the other root is $m = 0$. Therefore, the values and vectors for this case derived from equation 4.20 are:

$$\left[\ a, \left|\begin{matrix} 1 \\ 0 \end{matrix}\right|\ \right] \qquad \left[\ c, \left|\begin{matrix} 0 \\ 1 \end{matrix}\right|\ \right]. \tag{4.23}$$

Determining which is the lesser eigenvalue is trivial.

Returning to the general case, the two roots of Equation 4.22 are:

$$m_1 = -\frac{e}{2} + \frac{\sqrt{e^2+4}}{2} \qquad m_2 = -\frac{e}{2} - \frac{\sqrt{e^2+4}}{2}, \tag{4.24}$$

and the two eigenvalues and eigenvectors are

$$\left[\ a + bm_1, \frac{1}{\sqrt{m_1^2+1}}\left|\begin{matrix} 1 \\ m_1 \end{matrix}\right|\ \right] \qquad \left[\ a + bm_2, \frac{1}{\sqrt{m_2^2+1}}\left|\begin{matrix} 1 \\ m_2 \end{matrix}\right|\ \right]. \tag{4.25}$$

It is possible to present a single algorithm, accounting for both numeric accuracy and special cases, for reliably finding the lesser eigenvector. The strict equality test shall be replaced with an approximately equal operator $\simeq$. For practical purposes this will be implemented as equal out to the first 10 decimal places.

If $a - c \simeq 0$ and $b \simeq 0$, then the lesser eigenvalue is not well defined. The reason is that the when $b \simeq 0$ the two eigenvectors are are simply a and c, and if they are nearly equal then no well defined lesser eigenvalue exists. In this case the best-fit pose algorithm returns a special flag indicating the pose is undefined.

If $b \simeq 0$, then the lesser unit eigenvector is aligned with the coordinate axes:

$$a < c \Rightarrow \left|\begin{matrix} 1 \\ 0 \end{matrix}\right| \qquad c < a \Rightarrow \left|\begin{matrix} 0 \\ 1 \end{matrix}\right|. \tag{4.26}$$

Otherwise, and this is the general case, the two eigenvalues are:

$$\lambda_1 = a + b\left(-\frac{e}{2} + \frac{\sqrt{e^2+4}}{2}\right) \qquad \lambda_2 = a + b\left(-\frac{e}{2} - \frac{\sqrt{e^2+4}}{2}\right). \tag{4.27}$$

Then, the determination of which is the lesser eigenvalue is strictly dependent upon the sign of b:

$$b < 0 \Rightarrow \lambda_1 < \lambda_2 \qquad b > 0 \Rightarrow \lambda_1 > \lambda_2, \tag{4.28}$$

and consequently the eigenvector associated with the lesser eigenvalue is:

$$b < 0 \Rightarrow \frac{1}{\sqrt{m_1^2+1}}\left|\begin{matrix} 1 \\ m_1 \end{matrix}\right| \qquad b > 0 \Rightarrow \frac{1}{\sqrt{m_2^2+1}}\left|\begin{matrix} 1 \\ m_2 \end{matrix}\right|. \tag{4.29}$$

4.4 Integrated Point-to-line Distance

The sum of squared point-to-line distances as described in equation 4.10 has a weakness; it is not invariant to breaks in the data line segments. For the ith pair of corresponding segments, the error is the weighted sum of the perpendicular distance v_{ij} measured from endpoints 1 and 2 of the data segment:

$$E = \sum_{i=1}^{n} \frac{\ell_i}{2} \left(v_{i1}^2 + v_{i2}^2 \right) \quad \text{where} \quad v_{ij} \;=\; \left(\hat{N}_i \cdot \left(D_{ij} \vec{\Phi}_{DM} + \vec{T}_{DM} \right) - s_{MD} \rho_i \right). \tag{4.30}$$

This error changes if a single data segment is fragmented. The problem arises because equation 4.30 concentrates the error at the two endpoints. Integrating the squared perpendicular distance over the length of the data line segment cures this problem. Hence, the ISPD first presented in Chapter 3 is a preferable measure. Equation 3.4 in Chapter 3 expressed the perpendicular distance between points on the data segment and the model line as a parametric function of the distance v_{ij} measured from the endpoints of the data segment. The ISPD for a single pair of segments was given in equation 3.5. Here, this same measure is shown summed over n corresponding pairs of segments:

$$E = \sum_{i=1}^{n} \frac{\ell_i}{3} \left(v_{i1}^2 + v_{i1} v_{i2} + v_{i1}^2 \right) \qquad v_{ij} \;=\; \left(\hat{N}_i \cdot \left(D_{ij} \vec{\Phi}_{DM} + \vec{T}_{DM} \right) - s_{MD} \rho_i \right). \tag{4.31}$$

Equations 4.30 and 4.31 differ in that equation 4.31 contains a cross term and is renormalized accordingly. When the terms in equation 4.31 are expanded and collected as was done for equation 4.11, the form of the ISPD error is now identical to that given in equation 4.11:

$$E_{\text{ISPD}} \;=\; \vec{\Phi}^{\mathrm{T}} A \vec{\Phi} + 2 \vec{T}^{\mathrm{T}} B \vec{\Phi} + \vec{T}^{\mathrm{T}} C \vec{T} - 2 \vec{U}^{\mathrm{T}} \vec{\Phi} s - 2 \vec{V}^{\mathrm{T}} \vec{T} s + k s^2. \tag{4.32}$$

All constants except for A remain the same as defined above, but due to the cross term A becomes

$$A \;=\; \sum_{i=1}^{n} (\ell_i / 2) \left(D_{i1}^{\mathrm{T}} \hat{N}_i \hat{N}_i^{\mathrm{T}} D_{i1} + D_{i1}^{\mathrm{T}} \hat{N}_i \hat{N}_i^{\mathrm{T}} D_{i2} + D_{i2}^{\mathrm{T}} \hat{N}_i \hat{N}_i^{\mathrm{T}} D_{i2} \right). \tag{4.33}$$

Since equations 4.11 and 4.32 are identical up to the definition of A, the similarity transform which minimizes ISPD may be found in exactly the same manner as was described for minimizing equation 4.11.

4.5 Underdetermined Cases and Regularization

The ISDP measure does not uniquely define the model pose for a number of important configurations, such as when two model lines each match a single data line. An example is provided in Figure 4.4. In this case, the absolute size of the model does not alter the perpendicular error, and hence the pose is underdetermined. Figure 4.4a shows two model line segments drawn in bold. Figure 4.4b shows two data line segments drawn in black. Figures 4.4c and 4.4d show two alternative fits of this model to this data for which the ISPD is identical. The only difference is that the model is larger in Figure 4.4b than in Figure 4.4a.

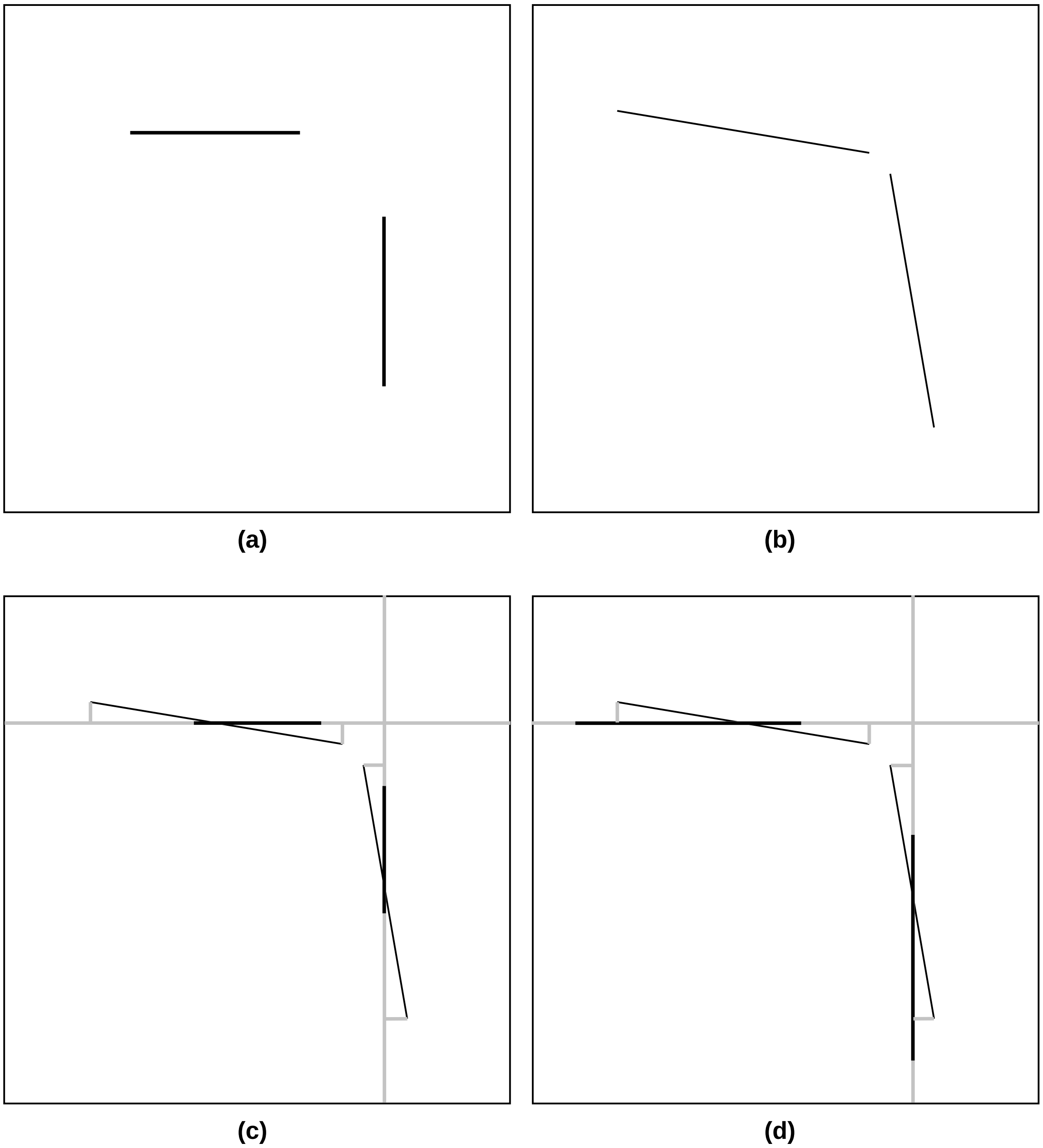


Figure 4.4 ISPD does not always uniquely determine pose. a) a model consisting of two segments, b) data segments to which this model may match, c) one possible pose of the model relative to the data, d) a second pose in which the larger model has equivalent ISPD error.

The reason that underdetermined cases are a problem will become more apparent once the local search algorithms are introduced in Chapter 5. To give a brief motivation here, the problem stems from the fact that when model pose is underdetermined the match error is undefined. Consequently, local search is cut off from considering such correspondences. Although the correspondences which are typically underdetermined are not generally the ones ultimately being sought, it is important to have the option of moving through such matches during search. Speaking very generally, the more that the search space of possible correspondence mappings is broken up by undefined states, the more likely it becomes that a potential path through the search space will be cut off.

A regularizing term is added to equation 4.32 to increase the number of correspondences with well defined best-fit poses. The midpoint-to-midpoint Euclidean distance between corresponding model and data line segments is an excellent regularizing term:

$$E_{\mathrm{PTP}} \;=\; \tau \sum_{i=1}^{n} \ell_i \left(D_i \vec{\Phi} + \vec{T} - s\vec{M}_i \right)^2, \tag{4.34}$$

where $\vec{M}_i$ is the midpoint of the ith model line segment and D_i is the matrix associated with the midpoint of the ith data line segment using the construction presented in equation 4.10. The weight τ controls the relative importance of the regularization term.

Adding midpoint-to-midpoint error to ISPD yields a new measure:

$$E \;=\; E_{\mathrm{ISPD}} + E_{\mathrm{PTP}}. \tag{4.35}$$

Using this measure, best-fit poses are uniquely determined for many otherwise underconstrained matches including the one illustrated in Figure 4.4. Whereas before, all matches involving only two model line segments were underdetermined, now the best-fit pose using the regularized fit measure is well defined for virtually all such matches.

To avoid biasing otherwise well determined configurations, τ may be made very small, for example 10^{-4}. In practice this makes the overall effect of the term E_{PTP} negligible except in those particular cases for which the transformation minimizing E_{ISPD} alone is underdetermined. Setting τ much higher than 10^{-2} begins to introduce noticeable biases into the resulting best-fit poses. Setting τ lower than around 10^{-6} runs the risk of dropping the regularization contribution below the floating point precision of the pose algorithm.

Equation 4.34 may be written in our standard form as follows:

$$E_{\mathrm{PTP}} \;=\; \vec{\Phi}^{\mathrm{T}} A \vec{\Phi} + 2\vec{T}^{\mathrm{T}} B \vec{\Phi} + \vec{T}^{\mathrm{T}} C \vec{T} - 2\vec{U}^{\mathrm{T}} \vec{\Phi} s - 2\vec{V}^{\mathrm{T}} \vec{T} s + k s^2, \tag{4.36}$$

$$\begin{array}{lll} A = \sum_{i=1}^{n} \tau \ell_i D_i^{\mathrm{T}} D_i & B = \sum_{i=1}^{n} \tau \ell_i D_i & C = \sum_{i=1}^{n} \tau \ell_i I \\ \vec{U} = \sum_{i=1}^{n} \tau \ell_i D_i^{\mathrm{T}} \vec{M}_i & \vec{V} = \sum_{i=1}^{n} \tau \ell_i D_i^{\mathrm{T}} \vec{M}_i & k = \sum_{i=1}^{n} \tau \ell_i \vec{M}_i^{\mathrm{T}} \vec{M}_i \end{array}. \tag{4.37}$$

In this way the regularized form of the error, equation 4.35, is expressed in the same form as equations 4.30 and 4.31. The regularized ISPD is obtained by adding the constants from equation 4.37 to those defined in equations 4.33 and 4.12.

There are certainly other ways of dealing with matches for which the optimal pose is not determined by ISPD. One alternative is to test for and handle underdetermined cases as special. There is nothing fundamentally wrong with such an approach. However, it involves extra mechanisms for both detecting and handling individual special cases. The attraction of the regularization approach lies in its simplicity and generality.

4.6 The Special Case of 2D-rigid Transformations

There is a somewhat unexpected result concerning the fitting of 2D-rigid models. This superficially simpler case is in fact more difficult than the variable scale problem addressed in the previous section. The rigid fitting problem may be obtained by setting $s = 1$ in equation 4.32:

$$E_{\text{ISPD}} = \vec{\Phi}^{\mathrm{T}} A \vec{\Phi} + 2\vec{T}^{\mathrm{T}} B \vec{\Phi} + \vec{T}^{\mathrm{T}} C \vec{T} - 2\vec{U}^{\mathrm{T}} \vec{\Phi} - 2\vec{V}^{\mathrm{T}} \vec{T} + k. \tag{4.38}$$

The rotation $\hat{\Phi}^*$ and translation $\vec{T}^*$ which minimize equation 4.38 depend upon the roots of a quartic equation. General closed form solutions for quartics are known. However, they require considerably more computation than is required to solve a quadratic equation. There do not appear to be any obvious factorizations which reduce the order of the quartic equation, and hence it appears that solving the quartic equation is the best way of finding $\hat{\Phi}^*$ and $\vec{T}^*$:

$$\vec{T}^* = -C^{-1}\left(B\vec{\Phi} - \vec{V}\right). \tag{4.39}$$

Substitute $\vec{T}^*$ back into equation 4.38 to obtain:

$$E = \vec{\Phi}^{\mathrm{T}} F \vec{\Phi} + 2\vec{W}^{\mathrm{T}} \vec{\Phi} + l, \tag{4.40}$$

where

$$F = A - B^{\mathrm{T}} D B \tag{4.41}$$

$$\vec{W} = B^{\mathrm{T}} D \vec{V} - \vec{U} \tag{4.42}$$

$$l = k - \vec{V}^{\mathrm{T}} D \vec{V} \tag{4.43}$$

$$D = \left(C^{-1}\right)^{\mathrm{T}}. \tag{4.44}$$

$$\tag{4.45}$$

The optimal rotation $\hat{\Phi}^*$ is that vector $\vec{\Phi}$ which minimizes equation 4.40 subject to the constraint that

$$\vec{\Phi}^{\mathrm{T}}\vec{\Phi} = 1. \tag{4.46}$$

Unlike equation 4.18 in which the elliptical error function was centered at the origin, the quadratic form in equation 4.40 is offset from the origin by an amount determined by $\vec{W}$, and this means the elliptical error curves are no longer centered at the origin. For this reason, the rigid problem does not reduce to a simple 2x2 eigenvector problem.

This constrained optimization problem may be solved by forming the Lagrangian

$$H = \vec{\Phi}^{\mathrm{T}} F \vec{\Phi} + 2\vec{W}^{\mathrm{T}} \vec{\Phi} + l + \lambda\left(\vec{\Phi}^{\mathrm{T}}\vec{\Phi} - 1\right), \tag{4.47}$$

and setting the partial of this equation with respect to $\vec{\Phi}$ equal to zero. This results in two constraints:

$$ax + cy + \lambda x + d = 0 \tag{4.48}$$

$$cx + by + \lambda y + e = 0 \tag{4.49}$$

where

$$F = \begin{vmatrix} a & c \\ c & b \end{vmatrix} \qquad \vec{W} = \begin{vmatrix} d \\ e \end{vmatrix} \qquad \vec{\Phi} = \begin{vmatrix} x \\ y \end{vmatrix}. \tag{4.50}$$

A single constraint equation in x and y alone may be obtained by multiplying equation 4.48 by y, multiplying equation 4.49 by x, and then subtracting one from the other:

$$cx^2 - cy^2 + bxy - axy + ex - dy = 0. \tag{4.51}$$

To solve equation 4.51 subject to the constraint $x^2 + y^2 = 1$, the first order y terms are brought over to the right side and both sides are squared. The resulting equation has only second and fourth order terms in y and the substitution $y^2 = 1 - x^2$ yields a quartic equation in x alone.

$$k_1 x^4 + k_2 x^3 + k_3 x^2 + k_4 x + k_5 = 0. \tag{4.52}$$

$$\begin{aligned} k_1 &= a^2 + b^2 + 4c^2 - 2ab & \qquad k_4 &= bd - da - ce \\ k_2 &= da - bd + 2ce & k_5 &= c^2 - d^2 \\ k_3 &= d^2 + e^2 - k_1 & & \end{aligned} \tag{4.53}$$

Letting the desired root of equation 4.52 be $x^* = \cos\phi^*$, the best-fit rotation

$$\hat{\Phi}^* = \begin{vmatrix} \cos\phi^* \\ \sin\phi^* \end{vmatrix} \tag{4.54}$$

minimizes equation 4.40. The associated optimal translation $\vec{T}^*$ follows directly from equation 4.39. Since closed form methods exist for solving general quartic equations, the solution to the optimal rigid transform problem is closed form.

It is interesting to consider whether there might be a way of reducing the complexity to that of solving a third or even a second order polynomial. One avenue of investigation involves equating this problem to another perhaps more commonly studied one. Toward this end, consider again the quadratic solution of the variable scale case, and Figure 4.3 in particular. It was possible to find the optimal vector $\hat{\Phi}^*$ as the solution to an eigenvector problem because the origin of the unit circle is coincident with the origin, or bottom of, the elliptical ISPD error surface. One way to think about what takes place when scale is fixed, is that this condition no longer holds. The unit circle on which the solution vector $\hat{\Phi}^*$ is constrained to lie may be displaced relative to the elliptical error function.

If this geometric interpretation is pursued, it is possible to transform the rigid fitting problem into an equivalent problem of determining the point on an ellipse closest to an arbitrary point in the plane. It might have been hoped that this more basic geometric problem would posses a general solution simpler than that of solving a quartic equation. However, this is not the case. To the contrary, it is recognized that this problem has no better than a general quartic solution [88].

In conclusion, there are strong practical incentives for bypassing the rigid matching case and moving directly to the variable scale formulation. One of the basic computational costs of computing the best-fit variable scale pose is that of solving the quadratic equation. In turn, the principle cost of this computation involves finding a single square root. In comparison, the closed form for the quartic equation requires solving 4 square roots. There is also a comparable increase in the number of floating point additions and multiplications.

CHAPTER 5

Local Search Geometric Matching

To review the matching problem as it has been developed thus far, Chapter 3 defined the combinatoric space of possible matches and a match error function. The match error function returns an evaluation for essentially any correspondence mapping between model and image segments. As emphasized in Chapter 4, to do this requires determining the best-fit pose of the model relative to the data, and in fact the only correspondences for which the match error is undefined are those for which there is no unique best-fit pose.

This chapter takes up the problem of finding the correspondence mapping, and hence match, with the lowest possible match error. The combinatorics of this optimization problem are daunting. There are 2^n possible correspondences for n potentially paired segments. It may surprise some readers that the simple and general algorithms developed here perform as well as they do. These algorithms probabilistically find globally optimal matches for geometric matching problems of widely varying size and form. As will be described in Chapter 6, there is empirical evidence suggesting that required computation grows as a linear function of n^2 rather than exponentially.

Local search is the basis for the algorithms presented below. The first uses of local search on difficult combinatorial optimization problems is generally considered to be that of Kernighan and Lin [75, 64, 76]. Of course, some of the constituent ideas date back even farther. The essentials of local search have been generalized by Papadimitriou and Steiglitz in their text on combinatorial optimization [92]. Local search has been found to be robust and practical for a variety of problems. These previous successes, coupled with the elegant simplicity of the central ideas, has made the adaptation of local search to matching an attractive and rewarding proposition.

Random-start local search relies principally upon a combination of iterative improvement and random sampling. Iterative improvement refers to a repeated generate-and-test procedure by which the algorithm moves from an initial state to one that is locally optimal. Variations of this basically simple idea have appeared under many names, including hill-climbing, gradient-descent, and greedy search. The essential attribute is that a discrete local neighborhood of states is defined with respect to a current state. Local search moves to successively better states until it arrives at one which is locally optimal.

Random sampling is often used in conjunction with local search in order to overcome local optima. One execution of local search may not arrive at the global optimum. It may in fact become stuck upon a quite undesirable local optimum. However, in a series of trials, the probability of missing the global optimum drops exponentially with the number of trials.

Adapting local search successfully to the problem of geometric matching of visual models to image data has required several novel innovations. One is the match error function and fitting procedures introduced in Chapters 3 and 4. Another is the development of new local neighborhood definitions and algorithms which are particularly well-suited to matching. The first neighborhood defined in this chapter is the Hamming-distance-1 neighborhood. A more novel and powerful neighborhood is introduced as part of the *subset-convergent local search* algorithm described below. The subset-convergent algorithm is demonstrated to be highly effective on a wide range of matching problems.

Another novel aspect of local search matching is the recognition that local optima are in some cases desirable. Consider an image with two distinct instances of the same object. Random imperfections will cause one instance to match slightly better than the other, in which case a match to one is globally optimal, while a match to the other is not. A series of multiple instance matching problems are shown in Chapter 6, where each instance is found as a distinct locally optimal match using the subset-convergent algorithm developed in this chapter.

5.1 Hamming-distance-1 Steepest-descent Local Search

This section introduces a fairly simple local neighborhood definition based on Hamming-distance. This neighborhood will be shown to be adequate for some problems and inadequate for others. More will be said later about adequacy below, but for now suffice it to say a neighborhood is inadequate when locally optimal solutions proliferate to such an extent that in practical terms search never finds the globally optimal match.

5.1.1 The Hamming-distance-1 Neighborhood

A local neighborhood is a set of states obtained by perturbing the current state. A particularly simple neighborhood consists of all correspondence mappings obtained by adding or removing a single pair of model data segments from the current correspondence. Recall the notation developed in Chapter 3 (page 47), and define a neighborhood consisting of all correspondence mappings c' obtained by adding or removing a single pair $s \in S$ from the current correspondence mapping $c \in C$.

When correspondence mappings are represented as bitstrings, this neighborhood C_1 may be equivalently described as the 'Hamming-distance-1' neighborhood, since it contains all bitstrings for which Hamming-distance to the current correspondence equals 1. An example is provided in Figure 5.1. Figure 5.1a illustrates the encoding of correspondence mappings as bitstrings first introduced in Figure 3.4. The specific correspondence mapping shown is that of the globally optimal match. Figure 5.1b illustrates the Hamming-distance-1 neighborhood defined about the optimal match.

The size of the Hamming distance-k neighborhood is n^k where, as before, n is the number of candidate pairs in S. Early phases of this work considered the Hamming-distance-2 neighborhood [11]. However, it quickly became apparent that the n^2 neighborhood size was a problem. As has already been mentioned, and will be described in detail in Chapter 6, the algorithms developed below require on the order of n^2 time to find optimal matches with high confidence. An algorithm which requires on the order of n^2 time to examine the local neighborhood is not interesting.

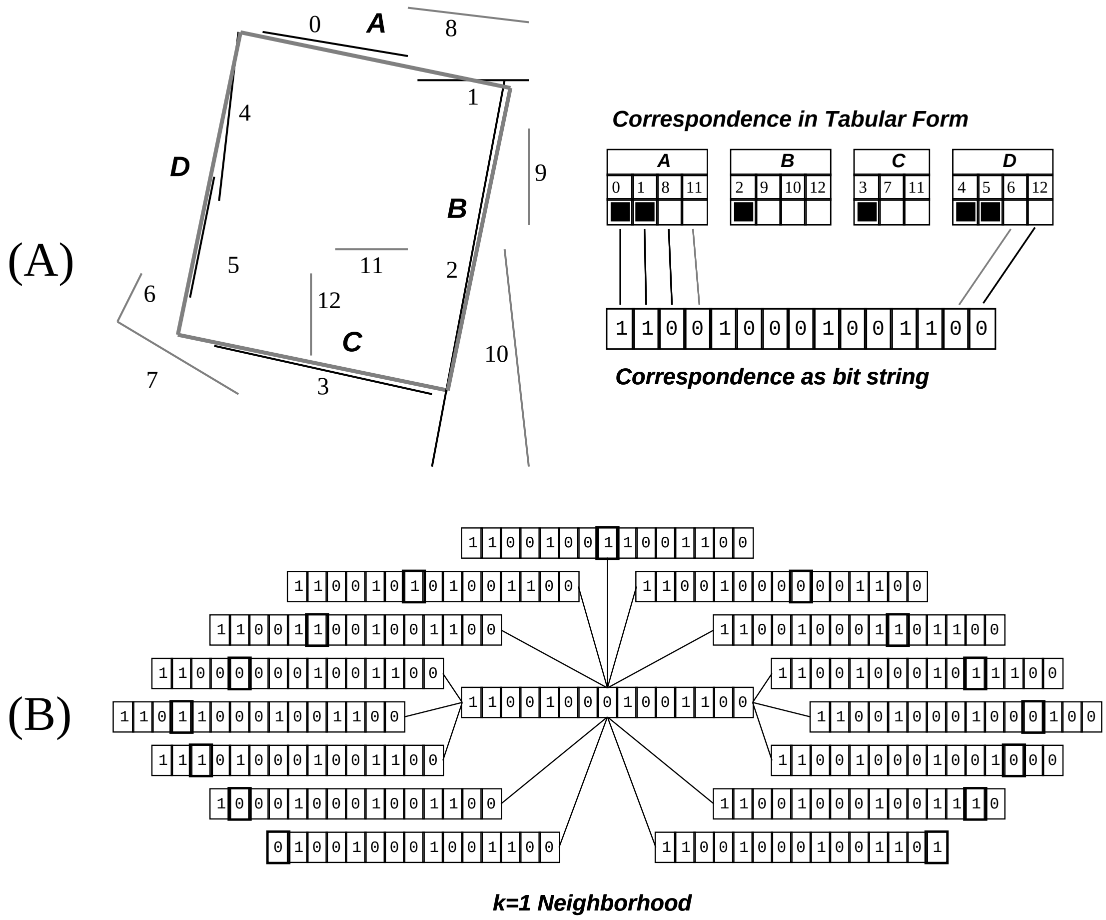


Figure 5.1 Hamming-distance-1 neighborhood. A) The rectangle shown matched to image data and the bitstring encoding. B) The 15 Hamming-distance-1 neighbors of the optimal match.

5.1.2 Steepest-descent Versus First-improvement

Two common ways of examining a neighborhood are first-improvement and steepest-descent [92]. In first-improvement, the first neighbor found to be better is adopted as the new state. In steepest-descent, the entire neighborhood is examined and the neighbor yielding the greatest improvement becomes the new current state. Early phases of this work [11] used a first-improvement strategy, while more recently steepest-descent has been favored.

One reason for this preference is the tendency of steepest-descent to find shorter paths to better matches. Another is that neighborhood evaluation is more uniform using steepest-descent, and when an entire Hamming-distance-1 neighborhood is evaluated, there are mechanisms described in Section 5.2 for making this evaluation more efficient. Finally, steepest-descent is the inherently deterministic while, for the reason given below, first-improvement is not.

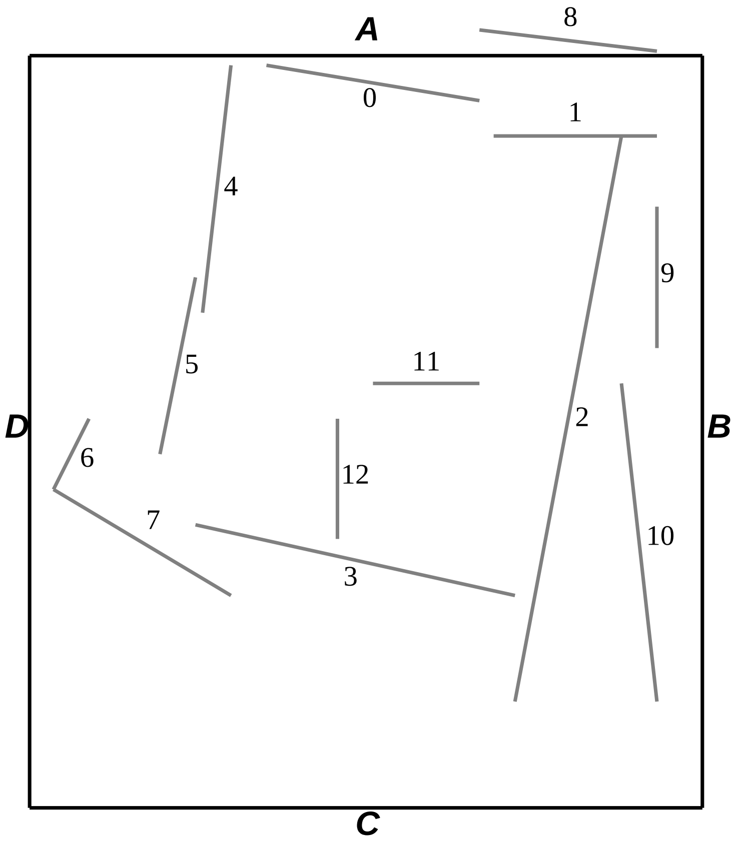
In first-improvement, there is an arbitrary dependence upon the order in which the neighbors are tested. If several neighbors are better than the current match, then depending upon which is found first, the algorithm may proceed down different search paths to different locally optimal matches. Because this ordering is arbitrary, it is common to randomize the selection of neighbors in order to avoid biasing the search. This randomness invalidates the simple conceptualization of local search to be presented in Section 5.3.2, and it is conceptually simpler to introduce randomness solely through the selection of random initial matches.

Figure 5.2 and 5.3 provide an example of the Hamming-distance-1 steepest-descent algorithm. Each row in the table of Figure 5.2 indicates a successively better match, starting with a match picked at random, and converging to the locally optimal match indicated in row 12. The labeled model and data segments are also shown for convenience. Successive matches are shown in Figure 5.3 to emphasize the updating of pose using the techniques described in Chapter 4. Since the initial match was chosen at random, the initial fit between model and data is poor. The pose improves considerably as pairs of poorly matching segments are dropped and replaced by pairs of better matching segments.

Looking into the example in Figures 5.2 and 5.3 in greater detail, recall from Sections 3.4 and 3.5 that the match error function is parameterized by maximum displacement σ and attenuation a. In the examples in this chapter, $\sigma = 5.0$ and $a = 0.75$. Also recall that the highest possible omission error is 1.0, and that therefore any correspondence with a match error above 1.0 can be improved by removing one or more pairs of segments.

The match error of 3.52 for the initial random correspondence in the first row of Figure 5.2 is considerably above 1.0. The first thing the steepest-descent algorithm does is remove pair $(A, 11)$, dropping the match error by more than half to 1.52. Two more pairs are removed, until in row four a correspondence is found with a match error of 0.60. From this match, the greatest improvement is obtained by adding pair $(D, 4)$. The algorithm continues to add and remove pairs until it arrives at a locally optimal match in row 12.

Depending upon the choice of the initial starting match, the globally optimal match may or may not be found. Figure 5.4 shows an example of a locally but not globally optimal match in the Hamming-distance-1 neighborhood. There is no addition or removal of a single pair which improves upon the match shown. Overcoming local optima is at the heart of creating effective local search algorithms, and an example of how subset-convergent local search breaks out of this local optima will be shown below.



Row	Error	A				B				C			D			
		0	1	8	11	2	9	10	12	3	7	11	4	5	6	12
1	3.52	●			●	●		●		●	●					●
2	1.52	●				●		●		●	●					●
3	0.81	●						●		●	●					●
4	0.60	●						●		●	●					
5	0.45	●						●		●	●		●			
6	0.38	●					●	●		●	●		●			
7	0.34	●					●	●		●			●			
8	0.31	●					●	●		●			●	●		
9	0.30	●	●				●	●		●			●	●		
10	0.28	●	●				●			●			●	●		
11	0.20	●	●			●	●			●			●	●		
12	0.10	●	●			●				●			●	●		

Figure 5.2 Hamming-distance-1 search: successive correspondences. Successive rows indicate successively better matches. The final locally optimal match is the same as shown in Figure 5.1. The labeled model and data segments are repeated for convenience. The successive matches are shown in Figure5.3

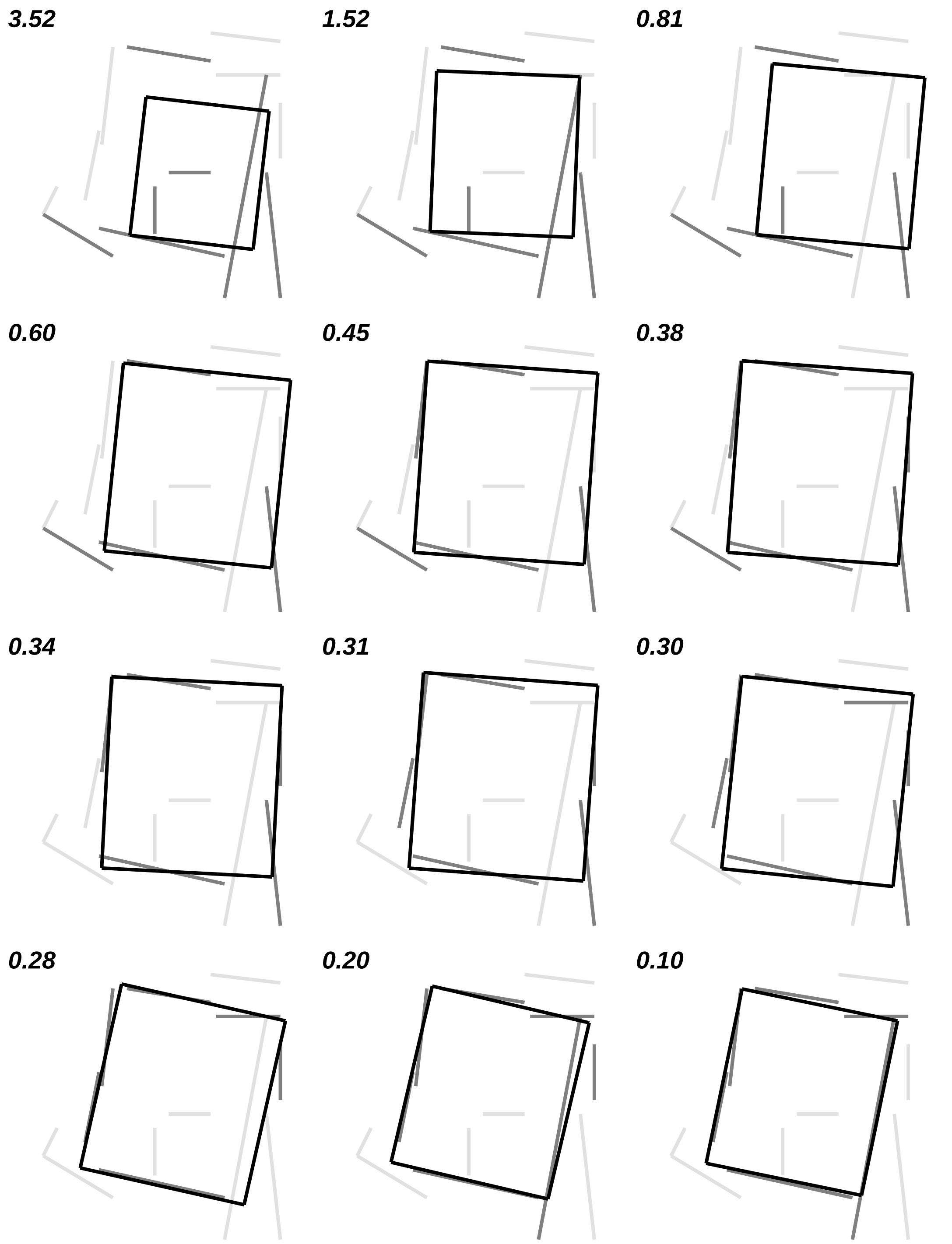
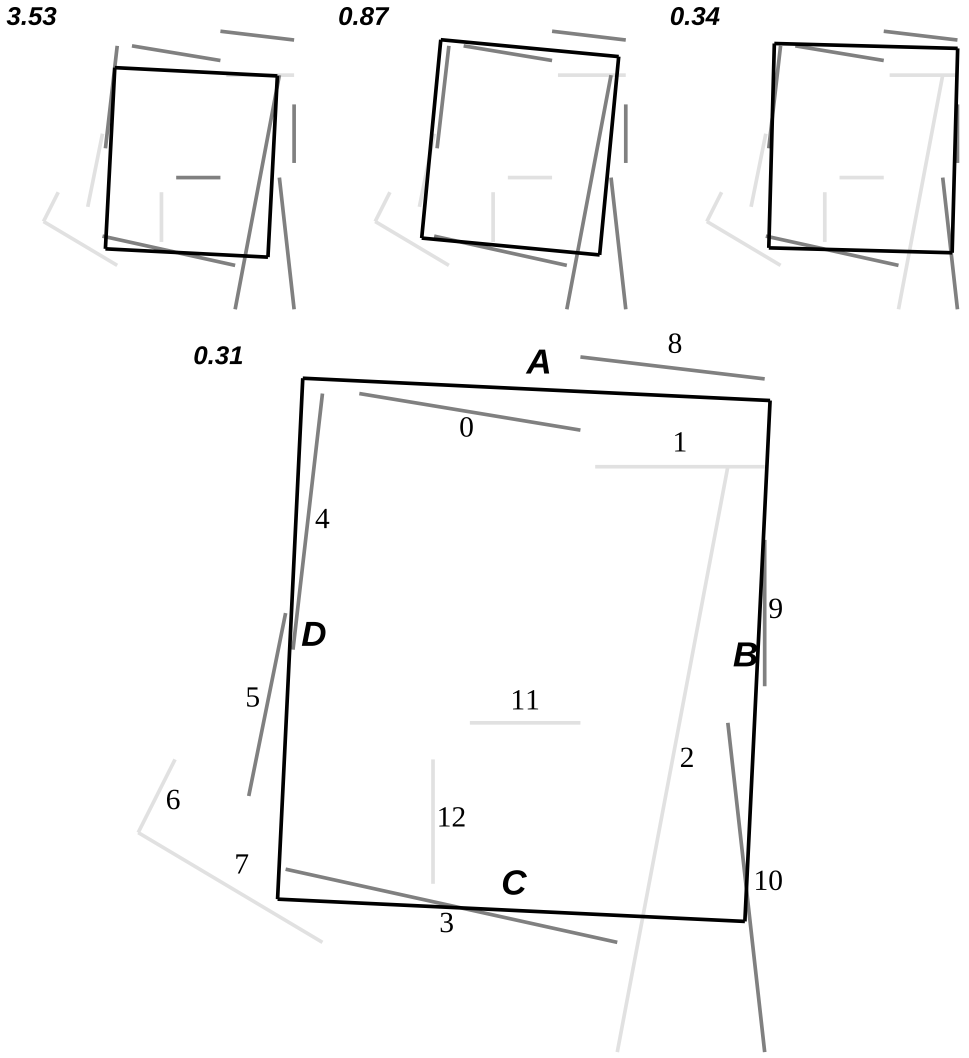


Figure 5.3 Hamming-distance-1 search: successive matches. The model is shown in best-fit relation to the data for each of the correspondences shown in Figure5.2



Row	Error	A: 0	A: 1	A: 8	A: 11	B: 2	B: 9	B: 10	B: 12	C: 3	C: 7	C: 11	D: 4	D: 5	D: 6	D: 12
1	3.53	●		●	●	●	●	●		●			●			
2	0.87	●		●		●	●	●		●			●			
3	0.34	●		●			●	●		●			●			
4	0.31	●		●			●	●		●			●	●		

Figure 5.4 Hamming-distance-1 local optima. Starting from the correspondence in the first row, the algorithm arrives at a locally optimal match in 4 steps. There is no single pair which may be either added or removed from this final match so as to reduce the match error.

5.1.3 *Qualifying the term 'Globally Optimal Match'*

The use of the terms locally and globally optimal warrants some explanation. In many difficult combinatorial optimization problems, it is unrealistic to presume that the true global optima is known. Consider, for example, an instance of the traveling salesperson problem with 100 cities. In many cases it will be difficult, if not impossible, to determine whether the best tour found by a local search algorithm is in fact the lowest cost tour possible. In other words, the best or globally optimal tour is not known. Consequently, evaluation of algorithms is often expressed in terms of how many of the solutions found are within some ϵ of the best found.

In contrast, with geometric matching it is often possible to tell from inspection whether a match is globally optimal. The match error quantifies the basic factors which go into how a person might judge a match. Consequently, when image data is not particularly ambiguous, as is the case for the rectangle, it is possible to be certain which match is globally optimal. Of course this is not always possible. For example, it is more difficult to be sure from inspection precisely which match is globally optimal in a case such as the car tracking example in Figure 1.6 (page 12). Nonetheless, when a geometric match is said here to be globally optimal, it is to be understood that this has been judged by visual inspection.

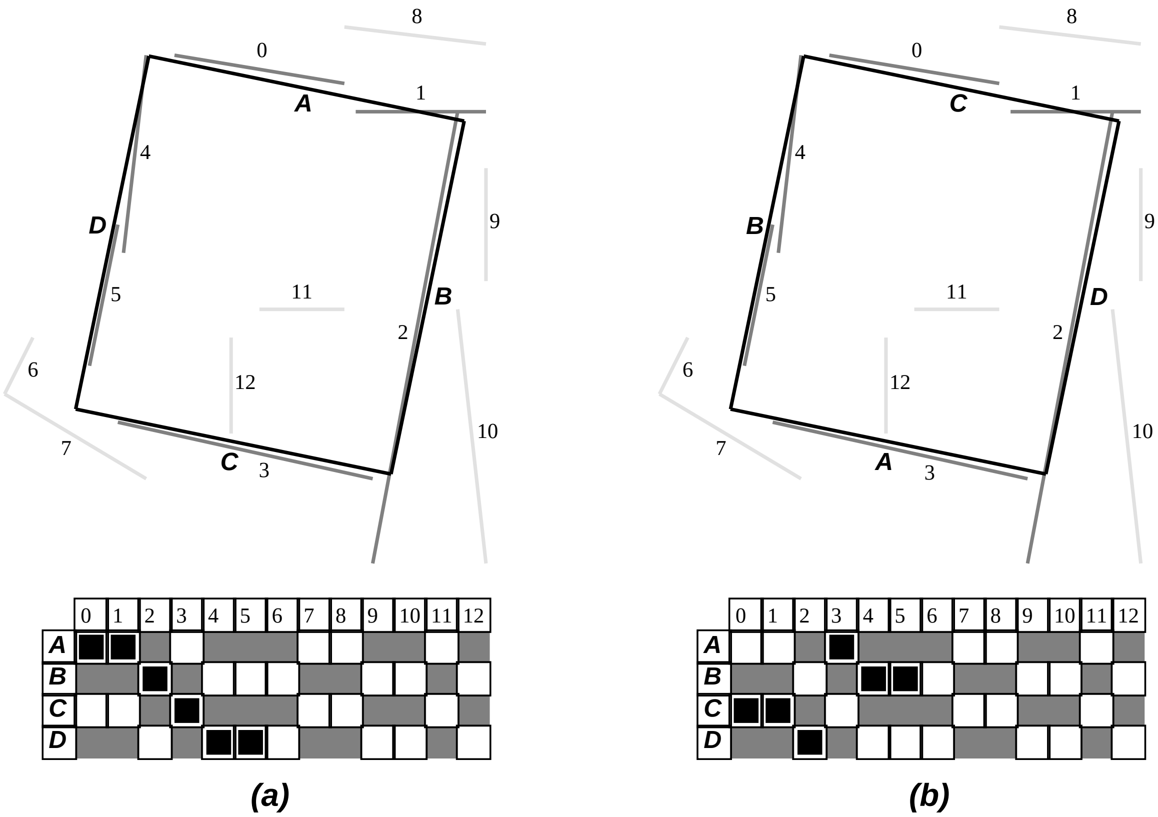


Figure 5.5 Symmetric models mean several matches are equally good. Black squares indicate matched segments, white squares indicate potentially matched pairs. The remaining pairs are not considered potential matches. a) one match, b) model rotated 180 degrees.

A comment should be made about symmetric models such as the rectangle. For such models there are multiple equivalent 'global' optima. The two equivalent matches for the rectangle are illustrated in Figure 5.5. The search space in this case is constrained to pairs S such that roughly horizontal image segments potentially match roughly horizontal model segments and similarly for vertical segments.

In the interests of brevity, the existence of the equivalent solutions will not typically be mentioned when they are of equivalent quality. More will be said in Chapter 6 about the relationship between symmetric and partially symmetric models and search. It will turn out that one of the strengths of random-start local search is that it is quite good at finding symmetric models. Moreover, and this is a very interesting alternative use of probabilistic local search, it turns out local search may be used to identify partial symmetries in a model. This is done by matching a model to itself and identifying distinct local optima which approach the global optima in overall match quality.

5.2 Efficient neighborhood evaluation

Steps can be taken to make the evaluation of the Hamming-distance-1 neighborhood more efficient. An understanding of these details is not essential and therefore this section may be skipped. However, those concerned with efficiency issues or desiring to build such an algorithm should be aware of these steps.

5.2.1 Incrementally Computing Fit Error

The first major computational saving comes from streamlining the way in which the best-fit pose is computed when testing the n neighbors of the current correspondence c. It is possible to compute these best-fit poses 'incrementally' off the current best-fit pose rather than computing them from scratch.

To see this, recall that weak-perspective fitting equation 4.11 (page 66) is multiplied out so as to be expressed in terms of matrices, vectors and a scalar, each of which is in turn summed over the set of pairs $s \in c$. What this means is that given these sums for the current match, the new values associated with adding or removing a particular pair may be computed incrementally by adding or subtracting the contribution associated with that pair.

The less efficient alternative would recompute the matrices, vectors and scalar in equation 4.11 for each neighbor. This in turn would be done by evaluating the sums over the set of pairs in the neighboring correspondence. Using this less efficient method, the complexity of computing E_{fit} for a neighborhood would be order the number of pairs in the match. Using the incremental approach, the complexity is constant: it does not depend upon the number of pairs in the the match.

5.2.2 Considering only Localized Changes in Omission Error

In contrast to the constant time required to compute E_{fit} for neighbors of the current match, the time required to compute the omission error E_{om} depends both upon the number of model segments and the number of pairs in the correspondence. This suggests it might be expedient to try to avoid computing E_{om} in cases where the increase in E_{fit} relative to the current match almost certainly precludes the neighboring match from being better than the current match. One way of doing this is to only consider the change in omission associated with the specific model line segment affected by the change in the correspondence. In other words, only the model segment for which the corresponding data segment is being either added or removed is examined.

The advantage of only considering localized changes to E_{om} is that it precludes the need to transform all model segments to the best-fit pose for all neighbors being tested. This saves computation. The disadvantage is that it means that indirect benefits associated with a change in correspondence may be missed during search, an a better match may be overlooked during neighborhood evaluation. An example of such an indirect effect will be illustrated using the example in Figure 5.6.

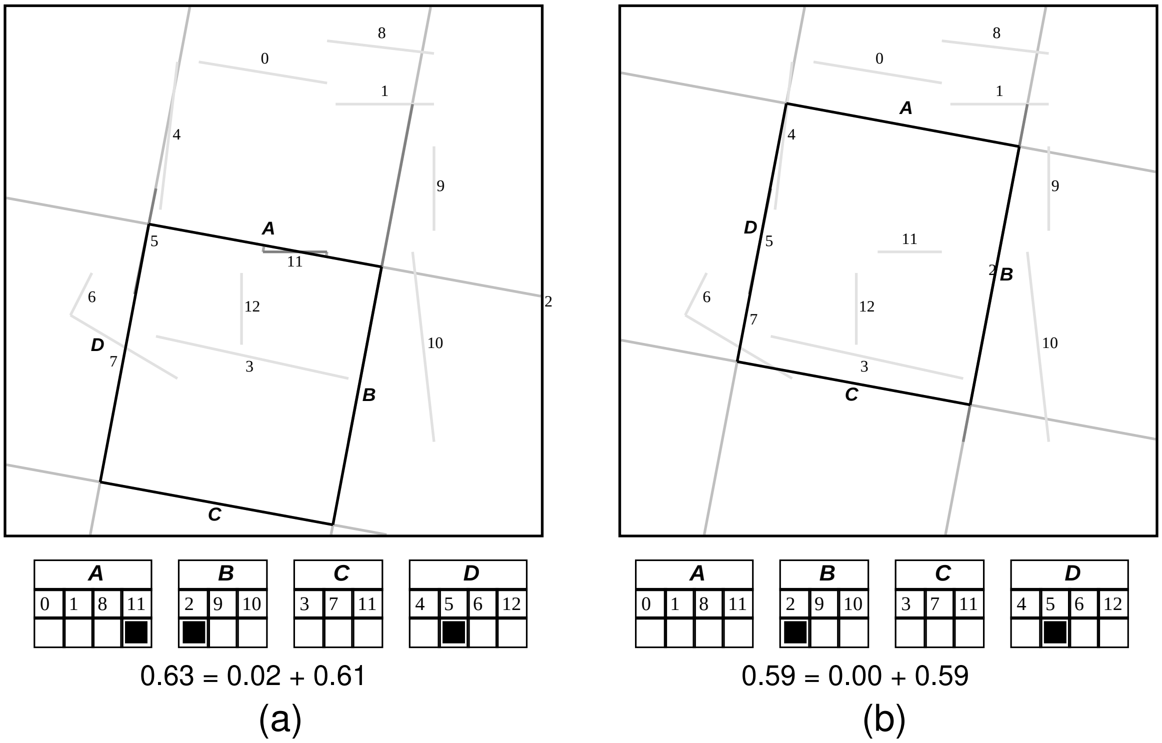


Figure 5.6 Indirect Omission Effects. a) Match with three image segments. b) Match with two image segments for which E_{om} is lower. Increased coverage of segments B and D makes up for the loss of coverage of segment A.

The correspondence for the match in Figure 5.6a is $((A, 11), (B, 2), (D, 5))$. The corresponding data segments are darker than the others. Note that the second two pairings are a part of the globally optimal correspondence shown in Figure 5.2, while the pairing $(A, 11)$ is not. The match error for this correspondence is 0.63 while the match error for the match obtained by removing pair $(A, 11)$ and shown in Figure 5.6b is 0.59. Removing $(A, 11)$ improves the match.

With the pair $(A, 11)$ removed, the model translates up relative to the two other data

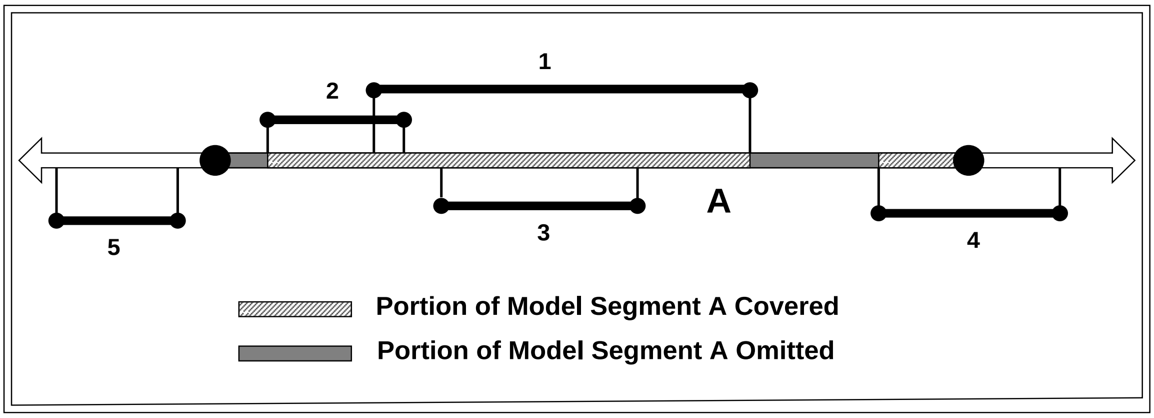


Figure 5.7 More precisely how omission is measured. Endpoints of model segment A are indicated with large dots. Portions of A omitted are shown in dark grey.

segments. This is because the midpoint-to-midpoint regularization measure described in Section 4.5 is being used, and with $(A, 11)$ gone, the only constraint in the direction of segments B and D is to minimize the squared distance between midpoints of paired segments B and 2 and D and 5. As a consequence of translating the model, the remaining data segments 2 and 5 more completely cover B and D. Note for instance that the segment D now lies completely on top of 5, and that no portion of 5 is visible in Figure 5.6b. As a consequence of this translation, the overall omission error drops from 0.61 to 0.59, despite the fact that A is completely uncovered in the latter case.

When the overall omission error is computed in evaluating a change such as that illustrated from the match in Figure 5.6a to the match in Figure 5.6b, the local search algorithm will identify this as a better match. However, if the change in omission error associated only with the model segment involved in the change is considered, in this case segment A, then the new match will not appear to be an improvement. This improved match in Figure 5.6b would be overlooked testing only for localized change in E_{om}.

5.2.3 Sorting Endpoint Projections to Compute Omission Error

It is worth saying a little bit about the mechanics of how E_{om} is computed. The first step is to transform the model line segments to the best-fit pose. It is with respect to these segments that coverage by corresponding data segments is measured. Then for each model segment the endpoints of corresponding data segments are projected onto the infinitely extended model line.

Figure 5.7 illustrates a model segment A in relation to corresponding data segments 1 through 5. The model segment has been fattened to permit alternative fill patterns to indicate the portions which are and are not covered by data. The projections of the data segment endpoints are indicated with line segments perpendicular to A. The endpoints of line segment A are shown as large dots. The portion of the infinite line containing A is mapped to the unit interval, with the origin at the left endpoint and coordinate 1.0 at the right endpoint. For every data segment i, it is possible to write a pair (l_i, h_i) representing the lower and higher coordinate of the endpoint projections.

For the example, this yields a list of pairs:

$$\{(l_1, h_1), (l_2, h_2), (l_3, h_3), (l_4, h_4), (l_5, h_5)\}\,. \tag{5.1}$$

These pairs may be interpreted as representing bounded intervals along the infinite line. The fraction of the model segment omitted may be described as the union of these low-high intervals intersected with the unit interval $(0, 1)$.

A fairly efficient way to compute this fraction is to begin by removing intervals from the initial list which do not intersect the model line. In the example, this removes (l_5, h_5), leaving the shortened list:

$$\{(l_1, h_1), (l_2, h_2), (l_3, h_3), (l_4, h_4)\}\,. \tag{5.2}$$

To determine the portion of the interval $(0, 1)$ not included in this list, it is helpful to sort the list based in ascending order of l_i. Again for the example, this gives a new list:

$$\{((l_2, h_2), l_1, h_1), (l_3, h_3), (l_4, h_4)\}\,. \tag{5.3}$$

It is now a simple matter to go through this list 'marking-off' covered portions of $(0, 1)$. To be a bit more specific, for each successive element in the list, it is possible to mark the portion of the segment beginning with l_i and ending with h_i as covered.

In order to try to make the potentially expensive sorting step efficient, a sorted list based upon the current correspondence is always maintained. When E_{om} is computed for a neighboring match, the assumption is that the pose does not generally change dramatically, and therefore the problem of sorting the endpoints after they have been projected from the new pose is one of resorting a nearly sorted list. Bubble sort is known to be fairly efficient under these conditions and is therefore used.

5.3 Local Optima and Random Sampling

Random sampling is commonly associated with local search as a means of overcoming local optima. More generally, it is a common technique for taking a statistically unreliable process, one which fails more than it succeeds, and through repeated trials producing a new reliable process. Readers already familiar with this idea will find the motivation familiar. However, it is important to understand how random sampling is not only used to solve individual problems, but how it is used to measure the difficulty of a new problem domain and thereby determine how many trials to run on problems drawn from this domain.

5.3.1 Using Independent Random Trials

Initiating t trials of local search from randomly and independently selected initial correspondences can dramatically improve the probability of finding the globally optimal match. Let P_s be the probability of successfully finding the globally optimal match on a single trial starting from a randomly selected initial correspondence. The probability of failing on all t trials is conjunctive:

$$Q_f \;=\; (P_f)^t\,, \qquad \text{where} \qquad P_f \;=\; 1 - P_s. \tag{5.4}$$

Table 5.1 Trials required to probabilistically solve problems as function of P_s. The number of trials required to ensure seeing the global optimum with confidence 0.90, 0.95 and 0.99 for selected values of P_s ranging from 0.75 to 0.01.

P_s	0.75	0.50	0.40	0.30	0.20	0.15	0.10	0.05	0.03	0.01
Trials 90	2	4	5	7	11	15	22	45	76	230
Trials 95	3	5	6	9	14	19	29	59	99	299
Trials 99	4	7	10	13	21	29	44	90	152	459

Given this relationship, it is possible to compute the number of trials t_s required to succeed with probability $Q_s = 1 - Q_f$:

$$t_s = \lceil \log_{P_f} Q_f \rceil. \tag{5.5}$$

To provide some intuition for the relationship expressed in equation 5.5, Table 5.1 gives the number of trials required to succeed with probability 0.90, 0.95 and 0.99 for a range of P_s values. So long as P_s does not drop below 0.05, 100 trials is sufficient to insure success with probability better than 0.99.

Two important conclusions can be drawn from Table 5.1. First, even a fairly weak algorithm, for example one which succeeds only 5 out of 100 times, can be used to solve a problem with very high confidence: better than 99% confidence if 100 independent trials are run. Second, the number of trials required to solve a problem begins to go up dramatically as the probability of success on a single trial begins to drop below about 0.05, and this imposes a practical limit on just how weak an algorithm can get and still be saved by random sampling.

Knowing how many trials are required to confidently solve a particular matching problem requires an estimate for P_S. Random sampling can be used to estimate P_S. To see this, let Ω be the space of possible geometric matching problems. This encompasses the geometric layout of model and data, the definition of the objective function, and the discrete space of possible correspondence mappings. Let Ψ be the space of possible random-start local search algorithms. This includes the neighborhood definition and the randomized procedure for selecting initial matches. Let $\mathcal{O} = \{s, f\}$ be the possible outcomes, where s indicates search has found the globally optimal match, and f it has not. Now P_s may be more precisely defined as

$$P(s \mid \omega, \psi). \tag{5.6}$$

If t independent trials of algorithm ψ on matching problem ω succeeds l_i times, then

$$\hat{P}(s \mid \omega_i, \psi) = \frac{l_i}{t} \tag{5.7}$$

is the maximum likelihood estimate for the true probability of success.

Provided a sufficient number of trials t are run, this estimate will be quite accurate; exactly how accurate can be derived based upon the underlying binomial distribution of the success/failure process. Table 5.2 shows 95% percent confidence ranges for the true probablity of success based upon $t = 100$ trials. The table is read as follows: given an estimate of the probability of success, here denoted simply $\hat{P}_s$, the table says that with probability 0.95 or better, the true probability of success lies between $P_{<s}$ and $P_{>s}$.

Table 5.2 Confidence bounds on probability of success estimates

$\hat{P}_s$	$P_{<s}$	$P_{>s}$	$\hat{P}_s$	$P_{<s}$	$P_{>s}$	$\hat{P}_s$	$P_{<s}$	$P_{>s}$	$\hat{P}_s$	$P_{<s}$	$P_{>s}$	$\hat{P}_s$	$P_{<s}$	$P_{>s}$
0.01	0.00	0.03	0.11	0.05	0.17	0.21	0.14	0.29	0.31	0.22	0.40	0.41	0.32	0.51
0.02	0.00	0.05	0.12	0.06	0.18	0.22	0.14	0.30	0.32	0.23	0.41	0.42	0.33	0.52
0.03	0.00	0.06	0.13	0.07	0.20	0.23	0.15	0.31	0.33	0.24	0.42	0.43	0.34	0.53
0.04	0.01	0.08	0.14	0.08	0.21	0.24	0.16	0.32	0.34	0.25	0.43	0.44	0.35	0.54
0.05	0.01	0.09	0.15	0.09	0.22	0.25	0.17	0.33	0.35	0.26	0.44	0.45	0.36	0.55
0.06	0.02	0.11	0.16	0.09	0.23	0.26	0.18	0.35	0.36	0.27	0.45	0.46	0.37	0.56
0.07	0.03	0.12	0.17	0.10	0.24	0.27	0.19	0.36	0.37	0.28	0.46	0.47	0.38	0.57
0.08	0.03	0.13	0.18	0.11	0.25	0.28	0.20	0.37	0.38	0.29	0.47	0.48	0.39	0.58
0.09	0.04	0.15	0.19	0.12	0.27	0.29	0.21	0.38	0.39	0.30	0.49	0.49	0.40	0.59
0.10	0.05	0.16	0.20	0.13	0.28	0.30	0.22	0.39	0.40	0.31	0.50	0.50	0.40	0.60

To move from an individual problem to domain of problems, first identify a test suite of problems which are representative of the domain. Next, run many trials on each test problem in order to estimate $\hat{P}_s$ for each problem. Finally, a conservative strategy is to use the lowest measured $\hat{P}_s$ to determine the number of trials to run on new problems. If the test suite is large, it may be desirable to throw out a few of the lowest $\hat{P}_s$ estimates as protection against the possibility of getting an uncharacteristically low estimate.

5.3.2 Searching the Forest

The work by Tovey [106] makes several insightful observations about local search. The first is that a ‘forest structure’ is imposed upon the search space by steepest-descent local search. To be more specific, the space may be viewed as a ‘forest’ of trees, with the root of each individual tree a locally optimal match. From nodes which are not locally optimal, there is path leading ‘down’ the tree to the root. These paths represent the successive matches found by a steepest-descent algorithm. There is one tree, the globally optimal tree, whose root is the global optima [1]. Figure 5.8 illustrates these ideas.

The form taken by these trees, their number and size for example, are a combined product of the local search neighborhood definition and the specific problem instance. This interplay between the neighborhood definition and the evaluation function defined over the state space makes formal analysis of local search difficult [106].

The ideal neighborhood would induce one and only one tree for every possible problem instance. In this way, local search initiated from any point in the search space would be guaranteed of reaching the global optima. In the local search literature, this is called an ‘exact’ neighborhood [92]. In general, and certainly for matching, tractable exact neighborhoods do not exist.

[1] For simplicity, ties for ‘best’ are ignored.

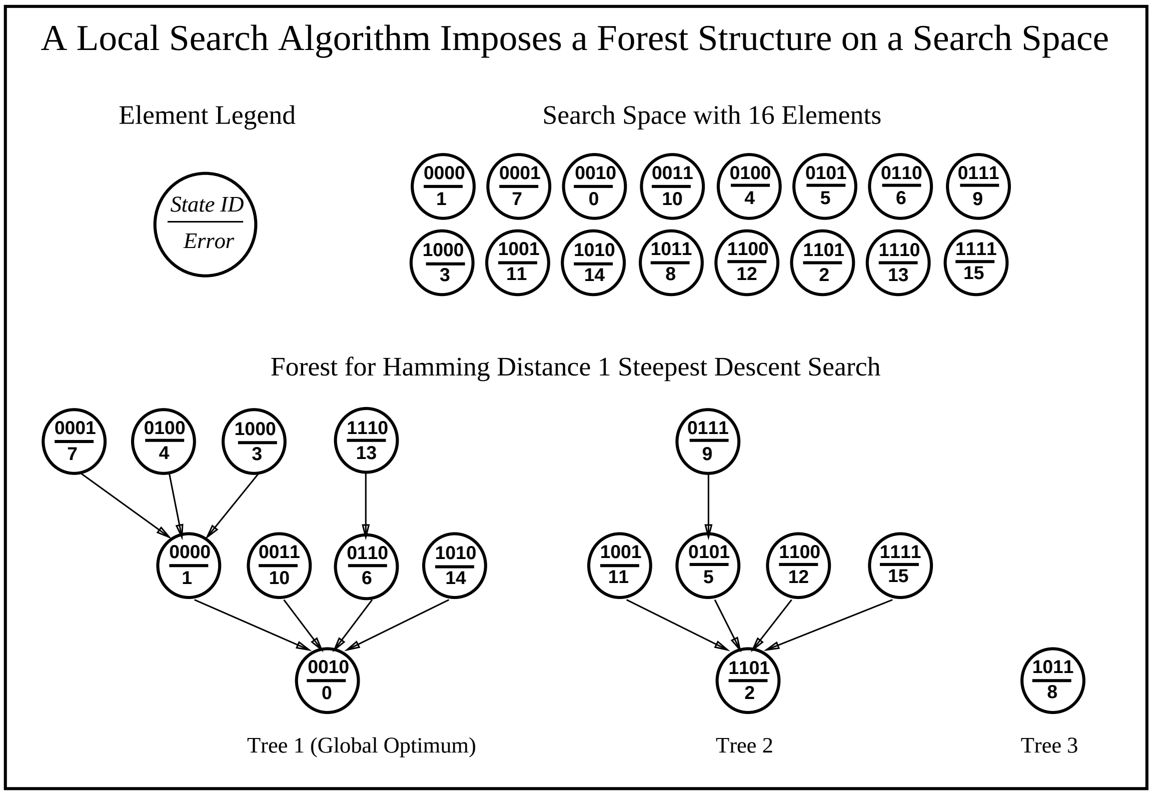


Figure 5.8 A forest imposed on a discrete search space. Steepest-descent local search with the Hamming-distance-1 neighborhood divides the search space into trees. The root of each tree is a locally optimal state. Local search initiated from a state leads down the tree to the root. If random-start local search uniformly selects starting points, then P_s is the ratio of size of the tree leading to the global optimum over the size of the search space.

5.3.3 Randomly Landing on Trees

Random sampling may now be reinterpreted with respect to this forest. Consider the probability of success P_s introduced above. Another interpretation of P_s is that it represents the probability of randomly selecting an initial match which lies on the globally optimal tree. This gives us a more general framework for describing the relationship between P_s and the structure of the search space.

One of Tovey's conclusions is that NP-complete problems will, under any reasonable neighborhood definition, induce an exponentially large number of trees in the forest. In other words, there will be an exponentially large number of local optima. At first this would appear to doom local search. However, Tovey also observes that it is incorrect to assume that all trees are of equal size.

This key issue is not the number of trees, but the size of the globally optimal tree. If initial starting points are drawn uniformly from the search space, then P_s is the ratio of the number of matches on the globally optimal tree over the total number of possible matches in the space. If the globally optimal tree contains a reasonable portion of the states in the search space, then the probability of success will be modestly good. Whether the remainder of the space is made up of one tree or a million trees is irrelevant.

An exciting aspect of using local search as a tool is that random sampling offers a means not only of finding globally optimal matches, but also of exploring empirically the structure of the matching search space. Running a series of trials and recording how often the globally optimal match is found is a means of estimating the structure of the forest.

5.3.4 Non-uniform Sampling, Trying for the Best Tree

The choice of initial random starting matches need not be uniform, and it is common here and elsewhere [92, 106] to bias the choice of starting matches so as to improve the probability that they lie on the globally optimal tree. Bias is introduced into the random selection of initial correspondences by defining a binding probability P_B. A pair $s \in S$ is included in the initial correspondence with probability P_B. The binding probability for a pair depends upon the model segment m in the pair. It is parameterized by r, which establishes the target number of data segments per model segment. To be more specific, let $k(m)$ be the total number of pairs involving m, then the binding probability is defined as:

$$P_B(m) \; = \; \max(0.5, r/k(m)). \tag{5.8}$$

In the implemented system, the parameter r is called the 'start-loading'. The rule of thumb used in this thesis is to set $r = 2$ for problems in which model pose is constrained, and $r = 4$ for problems which lack a pose constraint. Generally, the globally optimal match has between one and two data segments matched to each model segment, and it seems to help to start with matches involving approximately the same number of pairs as ultimately are expected to be in the best match. Those problems lacking a pose constraint are in some sense harder, since every model segment potentially matches every data segment. In these problems, empirical observation has suggested it is better to assign more pairs than are expected in the optimal match.

5.4 Subset-Convergent Local Search

Experience has shown that the Hamming-distance-1 local search algorithm described above is not alway adequate. For some classes of problems, such as those in which the set of candidate pairs S contains all possible pairings, $S = MxD$, the probability of success P_s becomes quite low. To boost performance, the more sophisticated subset-convert local search algorithm has been developed.

The central idea of subset-convergent local search is to test whether subsets of a locally optimal match in the Hamming-distance-1 neighborhood are 'consistent' with the overall match. For a truly good match, a Hamming-distance-1 local search initiated from subsets of the match should converge back to the same match. On the other hand, if the match is globally poor, then subsets of the match are probably incompatible. Drawing upon the ideas of the previous section, subsets of the match may lie on different trees as induced by the Hamming-distance-1 neighborhood, and steepest-descent initiated from these subsets can jump to these other trees and lead to overall better matches. Our experiments have shown this intuition to be correct. Of course the subset-convergent algorithm does not guarantee that the globally optimal match will be found on a single trial of random-start local search, but it does tend dramatically increase the probability of finding the optimal match on a single trial.

5.4.1 Subset Selection

How exactly to best select the subsets has not been a topic of major concern. The one guiding principle has been that the total number of subsets remain small. All the experiments in this and succeeding chapters have used no more than 4 automatically selected subsets. The first heuristic tried was found to be highly effective, and variations have not been explored. It is conjectured that the exact choice of subsets is not tremendously important, and therefore fine tuning the subset selection process has been a low priority.

The heuristics used to automatically select subsets of the model generalize the notion of a corner. Subsets are defined in terms of model segments, and the selection of subsets is performed off-line rather than during matching. The selection process selects pairs of non-parallel model lines with proximal endpoints. The specific selection algorithm begins with a list containing all $\frac{m(m-1)}{2}$ pairs of model segments. It then applies, in sequence, the following operations:

Remove nearly parallel pairs of segments: Remove pairs of segments differing in orientation by less then 5 degrees. The reason is that parallel segments constrain pose, specifically translation, less than non-parallel segments.

Retain the m pairs with nearest endpoints: Sort the remaining pairs in ascending order according to the minimum Euclidean distance between endpoints. Retain the first m pairs in this list, thus selecting the m pairs with the closest endpoints. It seems wiser to pick subsets which are spatially localized sub-features of the model.

Retain 4 disjoint pairs containing longest line segments: Sort the m pairs in descending order according to the sum of the lengths of the two segments. Longer line segments generally are more likely to appear in the data. Select the first 4 disjoint pairs in this list to serve as the subsets for the subset-convergent algorithm. To keep the local neighborhood of the algorithm from becoming excessively large, it is important not to let the total number of subsets grow as a function of m. This last step is modified for models containing fewer than 8 segments so that 4 subsets are still selected, even if they are not disjoint.

Provided there are at least 4 pairs of non-parallel model segments to begin with, this algorithm will always select 4 pairs of model segments to serve as subsets for the subset-convergent local search algorithm.

5.4.2 A Simple Illustration

A sample run of the subset-convergent local search algorithm is illustrated in Figures 5.9 and 5.10. The subset selection algorithm has automatically selected the subsets $\{A, B\}, \{B, C\}, \{C, D\}$ and $\{A, D\}$, which represent the four corners of the rectangle. For

		A				B				C			D				
Row	**Error**	0	1	8	11	2	9	10	12	3	7	11	4	5	6	12	
1	3.53	●		●	●	●	●	●		●			●				***Random Start***
2	0.87	●		●		●	●	●		●			●				
3	0.34	●		●			●	●		●			●				
4	0.31	●		●			●	●		●			●	●			***k=1 Optimal***
5 *	0.66	●		●			●	●									***Test Subset A, B***
6 *	0.51	●		●			●	●		●							
7 *	0.34	●		●			●	●		●			●				
8	0.31	●		●			●	●		●			●	●			***k=1 Optimal***
9 *	0.63						●	●		●							***Test Subset B, C***
10 *	0.48						●	●		●			●				
11 *	0.34	●					●	●		●			●				
12 *	0.31	●					●	●		●			●	●			
13	0.30	●	●				●	●		●			●	●			
14	0.28	●	●				●			●			●	●			
15	0.20	●	●			●	●			●			●	●			
16	0.10	●	●			●				●			●	●			***k=1 Optimal***
17 *	0.54	●	●			●											***Test Subset A, B***
18 *	0.33	●	●			●				●							
19 *	0.17	●	●			●				●			●				
20	0.10	●	●			●				●			●	●			***k=1 Optimal***
21 *	0.51					●				●							***Test Subset B, C***
22 *	0.35					●				●						●	
23 *	0.26				●	●				●						●	***k=1 Optimal***
24 *	0.55	●	●										●	●			***Test Subset D, A***
25 *	0.28	●	●			●							●	●			
26	0.10	●	●			●				●			●	●			***k=1 Optimal***
27 *	0.58									●			●	●			***Test Subset C, D***
28 *	0.29					●				●			●	●			
29 *	0.16	●				●				●			●	●			
30	0.10	●	●			●				●			●	●			***Subset Optimal***

Figure 5.9 Subset-convergent local search: successive correspondences. Successive rows trace out the search path. The starting correspondence and first 4 steps are the same as in Figure 5.4. In row 5, subset testing begins. Rows noted with an asterisk are worse than the best seen so far.

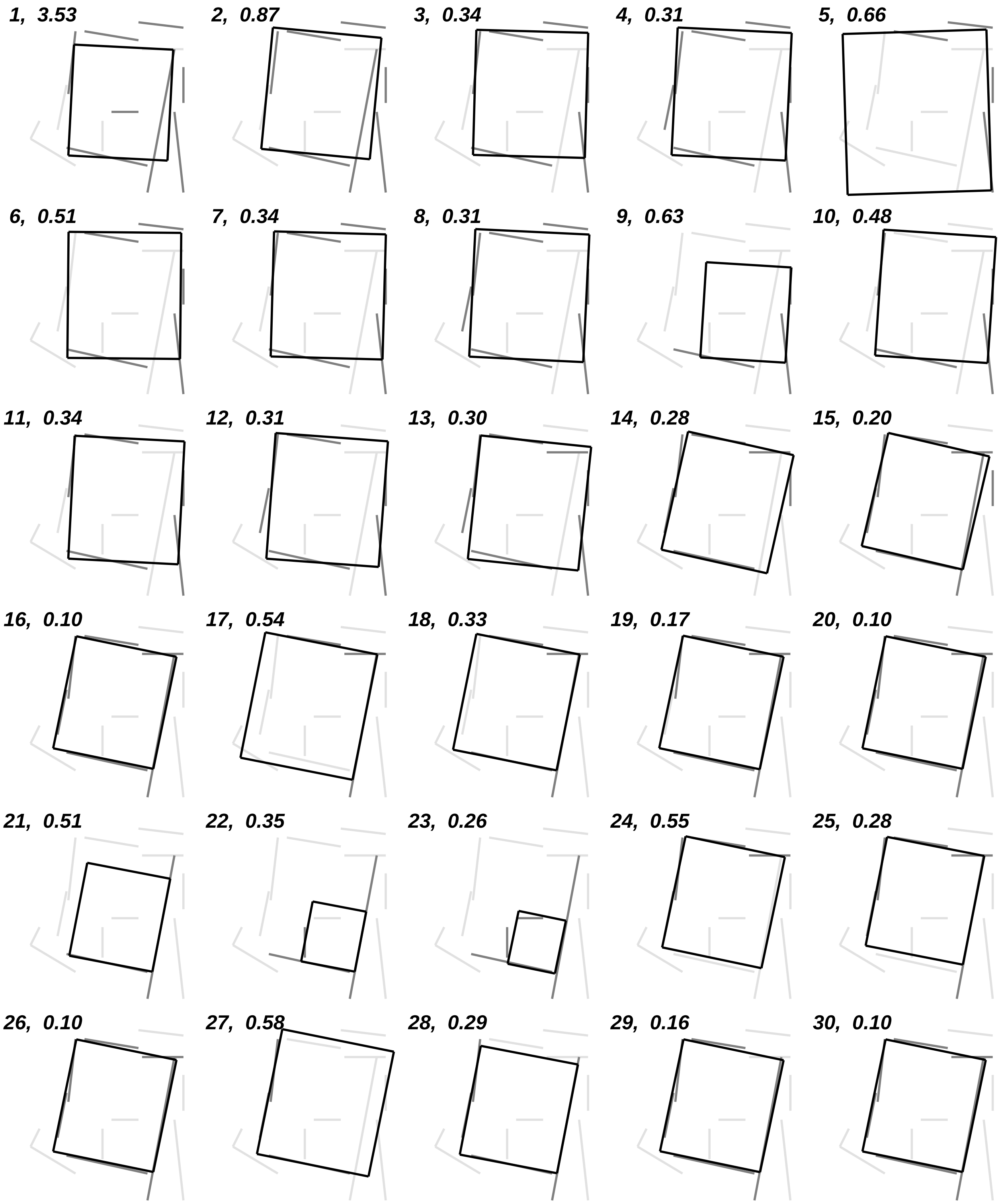


Figure 5.10 Subset-convergent local search: successive matches. These matches go along with the correspondences shown in Figure 5.9.

this illustration, the same starting point used to illustrate local optima in Figure 5.4 is used here. Observe that the first 4 steps are identical to those already shown. However, when the algorithm reaches the Hamming-distance-1 locally optimal match, it records this match and proceeds to search from a subset of the corresponding model-data pairs. For each of the 4 model segment subsets selected off-line, only those pairs of model-data segments including model segments in the selected subset are retained.

In testing from the subset $\{A, B\}$, all data segments not matched to this subset are dropped from the correspondence. The resulting match indicated in row 5 of Figure 5.9 is worse than the previous match. This is not surprising, since dropping segments often increases omission error. The algorithm resumes a Hamming-distance-1 search from this new match, and in this case arrives back at the same match in row 8. Next the subset $\{B, C\}$ is tested. From this match, search proceeds until an overall better locally optimal match is found in row 16.

The specific search path found by the steepest-descent Hamming-distance-1 algorithm in moving from the the match in row 9 to the match in row 16 is interesting. The 4 steps leading to row 13 involve adding pairs of segments to the match. As can be seen in Figure 5.10, in doing this the best-fit pose gets successively closer to the pose associated with the globally optimal match. In fact, evaluated from the best-fit pose for the correspondence in row 13, the pair of segments $(B, 10)$ no longer belongs in the match, and it is removed in the next step. With data segment 10 no longer matched to model segment B, matching data segment 2 to B is now an improvement, and $(B, 2)$ is added in moving from row 14 to row 15. It is then one last step, dropping pair $(B, 9)$, to the globally optima match.

The subset-convergent algorithm treats this Hamming-distance-1 locally optimal match as it would any other, and proceeds to initiate Hamming-distance-1 search from each of the 4 subsets. By row 30 none of this additional searching has revealed a better match, and the match found originally in row 16 is declared to be subset-convergent locally optimal.

Hamming-distance-1 and subset-convergent local search have been compared for the search space illustrated in Figure 5.5. In 1000 trials, subset-convergent local search found the optimal match 88 percent of the time. In contrast, Hamming-distance-1 search found the optimal match only 34 percent of the time. In the 1000 trials, Hamming-distance-1 found over 50 different locally optimal matches. The subset-convergent algorithm, in contrast, finds only one other match. This is the smaller rectangle formed by data segments 2, 3, 11 and 12, whose error is larger than for the best match, but less than most of the locally optimal matches found by the Hamming-distance-1 algorithm.

5.4.3 Examples Using Actual Image data

This section describes several examples of subset-convergent local search applied to matching problems involving data line segments extracted from actual intensity images. Three examples will be discussed: the aerial photography example from Chapter 1, the car tracking example from Chapter 1, and a telephone pole matching example presented here.

The example of finding the building in the aerial photograph presented in Figure 1.4 on page 11 is exactly the type of problem that originally inspired the work in this thesis. Many problems like it have been studied, and this is the sort of problem on which local search excels. On this particular problem, little effort was expended analyzing performance. The entire experiment took under a half hour, including time to setup the model by hand and perform 20 trials of subset-convergent local search. The match shown is the best found in these 20 trials.

In contrast, a detailed analysis of performance on the car tracking example in Chapter 1, Figure 1.6 on page 12, is presented here. The model is a hand picked subset of the straight line segments extracted from the image in Figure 1.5a, and the individual model segments are labeled and shown again in Figure 5.11a. This model is matched to segments extracted from each of the three images shown in Figure 1.5. Figure 5.11 shows in more detail the match to the line segments extracted from the image in Figure 1.5b. Figure 5.11b shows the data line segments labeled with numbers, and the table specifies the correspondence space with squares which indicate those pairs which are candidate matches. Filled in squares indicate a pair of model and data segments correspond to each other in the optimal match. The first match shown in Figure 1.6a involves something not typical in many model matching problems; the model is a subset of the data segments in Figure 1.6a, and therefore a *perfect* match is possible.

The following parameter settings were used in this example. The maximum displacement σ (Section 3.5) was set to 3.0. This means that lines displaced beyond 3 pixels will be dropped from the optimal match. The attenuation a on the omission error (Section 3.4) was set to 0.5. This means it is much more important to half cover two model segments than to fully cover only one. The incrementally computed fit error and local change in omission heuristic (Section 5.2) were used in this example. The weighting of omission error for individual model line segments was based upon relative length (Section 3.4). A scale transformation error term was added to the match (Section 3.6.2) in order to penalize matches for making the model either too small or too large. There is no pairwise error term (Section 3.6.1) included in the match error. Finally, the binding-probability r used to bias the number of initial random correspondences (Section 5.3.4) was set to 2.0. The 4 subsets of model segments used by the subset-convergent local search algorithm were selected off-line for the model using the heuristic selection criterion explained in Section 5.4.1.

To generate performance statistics, 100 trials of subset-convergent local search were run on each of the three matching problems. The resulting best matches are shown in Figure 1.6 and the match from Figure 1.6b is shown in more detail in Figure 5.11. Table 5.3 summarizes the results of these 100 trials. For each matching problem, the local optima found are listed by ascending match error. The number of times the optima was found is indicated. For example, the perfect match with zero match error was found in 78 out of the 100 trials.

The number of pairs of potentially matching pairs of segments, n, is indicated for each problem in Table 5.3, along with the resulting size of the search space: $|C| = 2^n$. There are 27 model segments and 65, 61 and 65 data segments for the three problems respectively. The initial placement of the model is used to filter the set of candidate matching segments $S \subseteq MxD$. Thus, in the first problem only 631 of the 1,755 possible pairs are considered candidates. Even with this filtering, it must be remembered that the search space contains 2^{641} distinct correspondence mappings.

The values shown in the columns of Table 5.3 are explained here:

Fr. The frequency with which the local optima was found.

E_{match} The match error for the local optima.

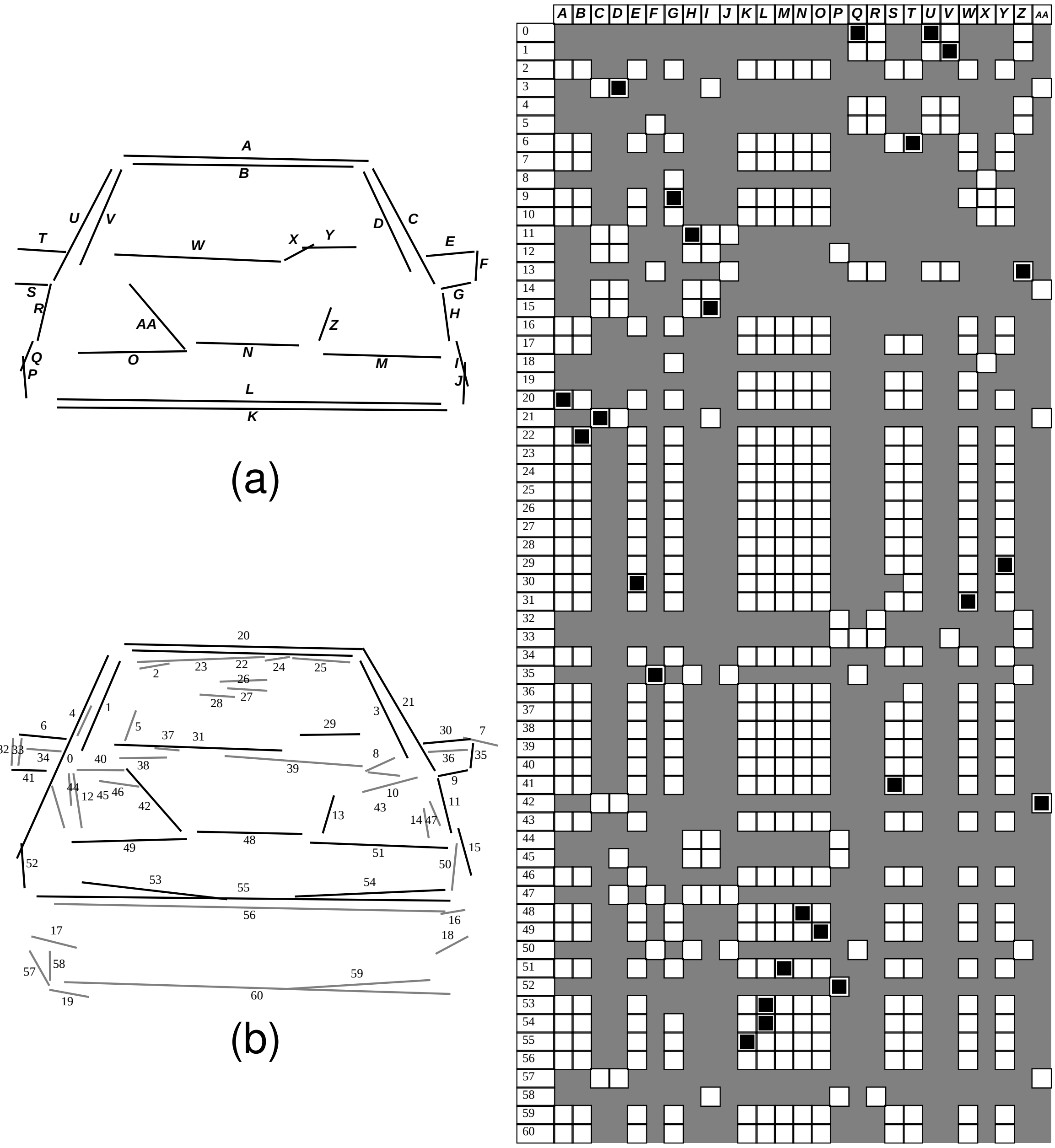


Figure 5.11 Detailed look at one car match from Figure 1.6: a) the car model, b) the data from Figure 1.6b extracted from the image in Figure 1.5b. The correspondence space of potentially matched pairs is indicated by the table. A square indicates a pair which potentially match, and a filled in square indicates a pair belonging to the optimal match.

Table 5.3 Performance summary for car tracking example in Figure 1.6. Each locally optimal solution found in 100 trials is listed in order of ascending match error, and the number of times each solution was found is indicated. Other entries are explained in the accompanying text.

Trials Summary for Figure 1.6a Example, $n = 631$, $\|C\| = 2^{631}$								
Fr.	E_{match}	$\left(\frac{1}{\sigma^2}\right) E_{\text{fit}}$	E_{om}	$E_{\mathcal{F}}$	Moves	Tests	Full Tests	Seconds
78	0.000	0.00	0.00	0.00	229	146,789	44,552	70.7
1	0.451	0.05	0.40	0.00	178	114,098	35,999	56.4
1	0.498	0.11	0.39	0.00	117	74,997	24,535	37.2
1	0.520	0.04	0.48	0.00	117	74,997	22,894	36.7
8	0.522	0.05	0.47	0.00	127	81,407	25,886	40.1
8	0.547	0.06	0.48	0.00	115	73,715	23,784	36.2
3	0.558	0.10	0.46	0.00	132	84,612	27,984	41.9

Trials Summary for Figure 1.6b Example, $n = 559$, $\|C\| = 2^{559}$								
Fr.	E_{match}	$\left(\frac{1}{\sigma^2}\right) E_{\text{fit}}$	E_{om}	$E_{\mathcal{F}}$	Moves	Tests	Full Tests	Seconds
59	0.160	0.09	0.07	0.00	221	123,539	37,586	59.8
11	0.521	0.04	0.48	0.00	123	68,757	20,832	33.6
7	0.588	0.11	0.48	0.00	109	60,931	19,168	30.0
1	0.598	0.10	0.50	0.00	112	62,608	19,698	30.7
2	0.621	0.06	0.56	0.00	109	60,931	18,958	30.2
20	0.724	0.00	0.72	0.00	82	45,838	15,072	23.0

Trials Summary for Figure 1.6c Example, $n = 521$, $\|C\| = 2^{521}$								
Fr.	E_{match}	$\left(\frac{1}{\sigma^2}\right) E_{\text{fit}}$	E_{om}	$E_{\mathcal{F}}$	Moves	Tests	Full Tests	Seconds
56	0.253	0.15	0.10	0.00	249	129,729	36,604	62.1
3	0.539	0.06	0.47	0.00	111	57,831	18,055	28.5
1	0.581	0.08	0.50	0.00	122	63,562	17,331	30.3
38	0.582	0.04	0.54	0.00	106	55,226	17,302	27.2
2	0.587	0.09	0.50	0.00	116	60,436	18,810	30.1

$\left(\frac{1}{\sigma^2}\right) E_{\text{fit}}$	The fit error contribution to E_{match}.
E_{om}	The omission error contribution to E_{match}.
$E_{\mathcal{F}}$	The scale transformation error contribution to E_{match}.
Moves	The average path length of a local search trial. Path length is the number of steps taken by the Hamming-distance-1 steepest-descent algorithm being used by the subset-convergent local search algorithm.
Tests	The average number of correspondence mappings tested during one trial. This value is the average number of moves times the neighborhood size n.
Full Tests	The number of neighbors for which it was necessary to do a full evaluation of omission error. Recall from Section 5.2 that to save time omission is not computed for all neighbors.
Seconds	The average number of seconds taken by a trial on a TI Explorer II Lisp Machine. Recent results suggest the algorithm runs roughly 60 times faster written in C and running on a Decstation 5000.

Subset-convergent local search does well on this example. In all three cases the best match is found in the majority of trials, and the perfect match seems highly attractive, being found 78 out of 100 times. It is likely that the algorithm is finding the globally optimal matches. However, this is not certain, since ambiguous local structure in this data makes it difficult to verify by eye whether the matches being found are globally optimal. There is no guarantee that better matches are found more often. In the second problem, the second to worst of the local optima found was found in 38 out of the 100 trials. Although of poor quality, this optima is clearly attractive in the sense that many trials converge to it. The distributions of optima shown for these problems is typical, with better matches being found with greater frequency.

A look at the constituent parts of the match error shows the fit error is roughly comparable for all local optima. The difference between the optimal and sub-optimal matches is in omission. This indicates that the sub-optimal matches consist of well fitting but smaller sets of data segments. Given how fit and omission compete, as described in Section 3.5, this result is to be expected.

It may seem odd to include the scale transformation error in this table, since it is zero for every locally optimal match shown. The intention is to emphasize, as discussed in Section 3.6, that although this term plays no role in ranking locally optimal solutions, it plays a significant role in directing local search to desirable local optima. Were this term not included in the match error, the algorithm would find many locally optimal matches in which the model is shrunk to perhaps 1/100th of its original size, and the frequency with which the best match is found would drop significantly.

A closer look at the local search path length, ‘moves’, indicates a desirable property of local search matching. For all three problems, the average path length for the best match is considerably longer than for the others. Often the difference is as great as two to one. This observation is not surprising given the discussion in Section 5.3.3 of how random sampling relates to the forest structure of the search space. The more attractive matches tend to be on larger trees, and on average, search paths to the optima down these trees tend to be longer. A consequence of this is that local search leading to less attractive optima tends to expend less effort, and this in turn means that local search matching has the attractive property of not running away and doing uncontrollable amounts of search when started from poor initial matches.

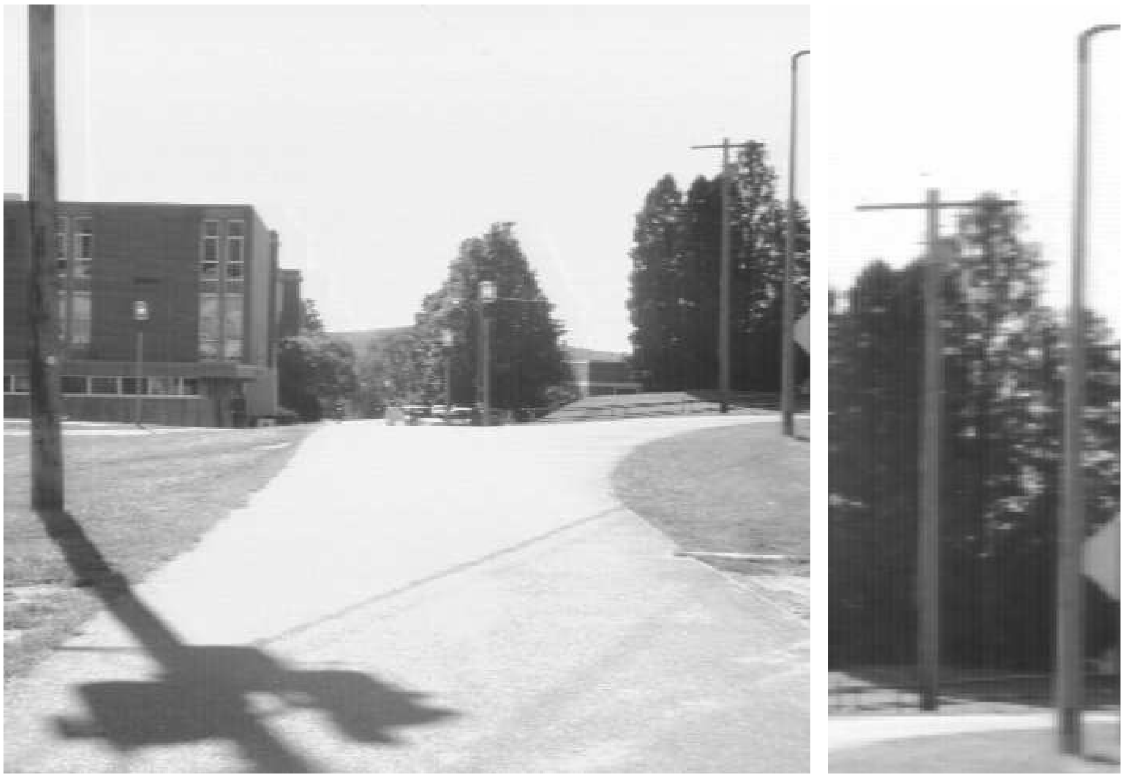
Recall the local omission heuristic described in Section 5.2. Comparing the entries in the 'Tests' and 'Full Tests' columns indicates how much computation is being saved by using this heuristic. The fit error fir neighbors is computed efficiently, and often the growth in fit error is so large as to offset any possible local gain associated with reduced omission error. For these cases, no further examination of the match is necessary. To illustrate, for the optimal match with zero error (top of Table 5.3), the average number of neighbors tested on one trial of local search was $146,789$. For only $44,552$ of these $146,789$ matches, or roughly one third, was it worthwhile to compute the change in omission error after computing the change in fit error.

Two possibly related trends are evident in this example: the match error goes up for the two scenes involving increasing scale change, and the frequency of finding the best match goes down. The match error is 0.00, 0.160 and 0.253 for the three problems respectively. The frequency of finding the best match is 78, 59 and 56. The probable explanation for the rising match error is that the segments extracted from the first image match less well the further the car moves from its original position. Whether local search tends to find better matches more frequently is an interesting conjecture, but one which would be hard to prove.

Granting that three experiments is too few to safely generalize to a problem domain, it is nevertheless instructive to go through the mechanics of generalizing as described in Section 5.3.1. The maximum likelihood estimate for the probability of success is 0.78, 0.59 and 0.56 for the three problems respectively. Picking 0.56 to be conservative, equation 5.5 indicates that 4 trials of local search are required to solve these problems with 95% or better confidence; 6 trials to reach 99% confidence. To indicate what this suggests about the associated amounts of search and the time required, 4 trials would on average require 256, 186 and 187 seconds respectively, and testing of $587,156$, $494,156$ and $586,916$ correspondence mappings respectively. The search spaces for these problems contains well over 10^{150} possible correspondence mappings, and to confidently find optimal matches in these spaces by examining less than $600,000$ states seems a very good result.

The last example presented in this section is that of recognizing a telephone pole in an outdoor image. Figure 5.12 shows a 512x512 intensity image taken looking down a walkway. The blow-up gives a better indication of the image quality. The cross bar at the top of telephone pole is no more than about two pixels across, and aliasing along this structure is pronounced. Figure 5.13 illustrates the telephone pole model, the image data, and the best match. Black squares in the table indicate matching segments. White squares indicate candidate pairs of segments not included in the optimal match. Grey areas indicate pairs of segments not considered candidates based upon the initial projection of the telephone pole into the scene.

The difficulty of this recognition task may not be immediately obvious. After all, the model appears simple, consisting of only three line segments, and the data to which it is to be matched contains an apparently modest number of horizontal and vertical line segments. However, this problem is difficult in part precisely because the model is so simple. There are a variety of ways of partially matching these three segments to the data shown. The two vertical sides of the telephone pole are a good match to the vertical sides of the street lamp. In fact, if the omission error is weighted by the relative length of model line segments as described in equation 3.13 (page 55), the match to the vertical lamp post sides has a lower



(a) (b)

Figure 5.12 Outdoor scene looking down walkway. a) full 512 by 512 image, b) section of image containing telephone pole and street lamp.

match error than the one shown. Adopting a uniform weighting between all three segments places greater emphasize on finding a mate for the top horizontal segment, and makes the match shown here the globally optimal match for this problem.

5.4.4 A Difficult Matching Task

To further illustrate the power of subset-convergent local search, consider the following example. Figure 5.14 shows data containing instances of the two models shown on the left. The models each contain 9 line segments. There are 61 data line segments. Roughly 1/3 of the data lines are broken and skewed pieces of the Deer model, and 1/3 are pieces of the Giraffe model. The remaining lines are random clutter. The search space in this case is the power set of the 549 possible model-data pairs. This problem was generated using a synthetic problem generator described in Chapter 6.

The model segment subsets automatically selected by the subset-convergent algorithm are:

$$\begin{aligned} \text{Deer} &\rightarrow \{B, E\}, \{C, D\}, \{H, I\}, \{A, F\} \\ \text{Giraffe} &\rightarrow \{A, E\}, \{B, D\}, \{C, I\}, \{F, G\} \end{aligned}$$

Each of these models contains more than 8 segments, and hence the automatic selection

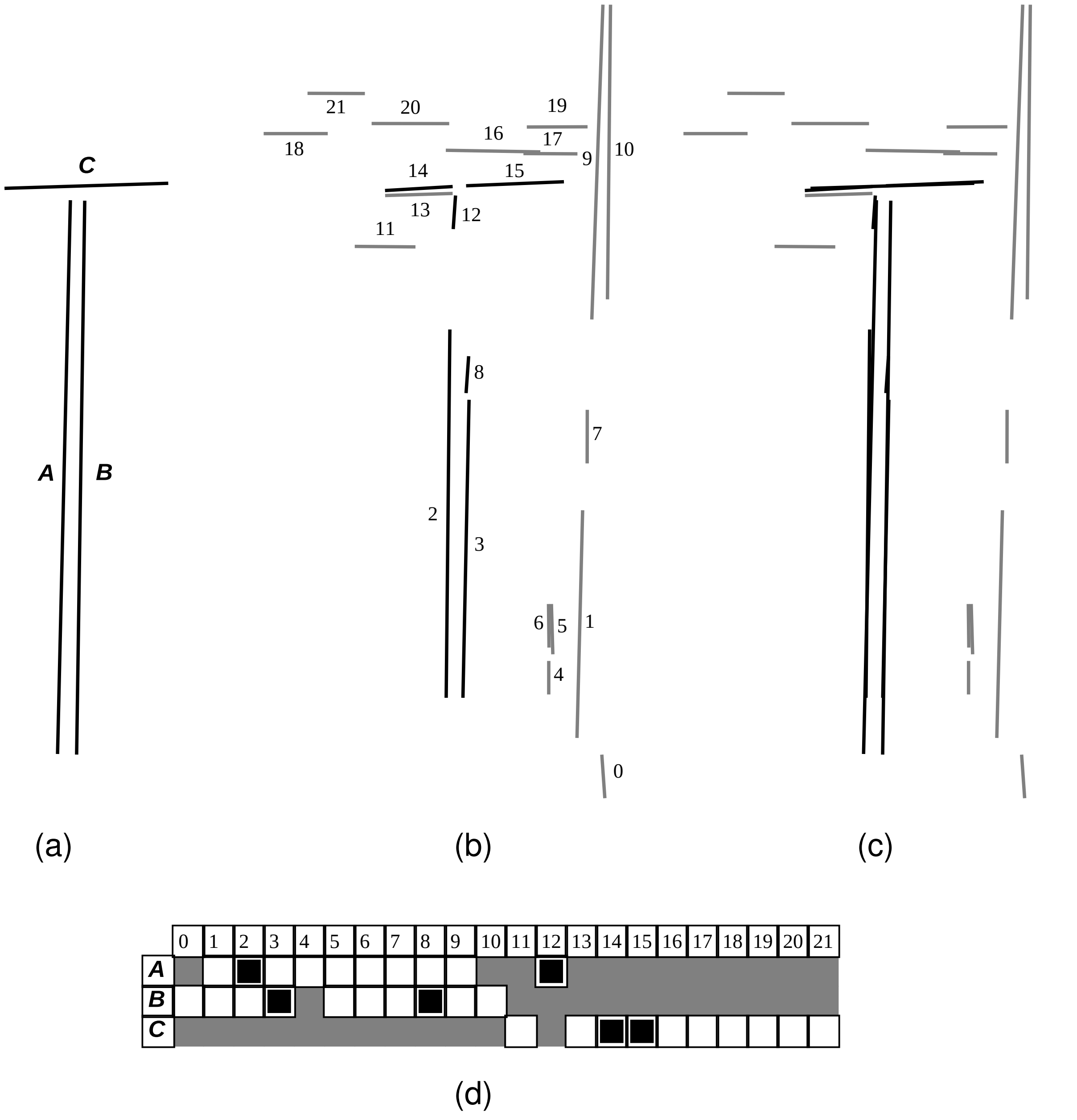


Figure 5.13 Telephone pole match example: a) the model segments labeled with letters, b) data segments labeled and those matched to model indicated in black, c) the model fit to the data for the optimal match found using subset-convergent local search, d) table indicating correspondence space. Squares indicate pairs in the search space, and filled in squares indicate pair belonging to the optimal match.

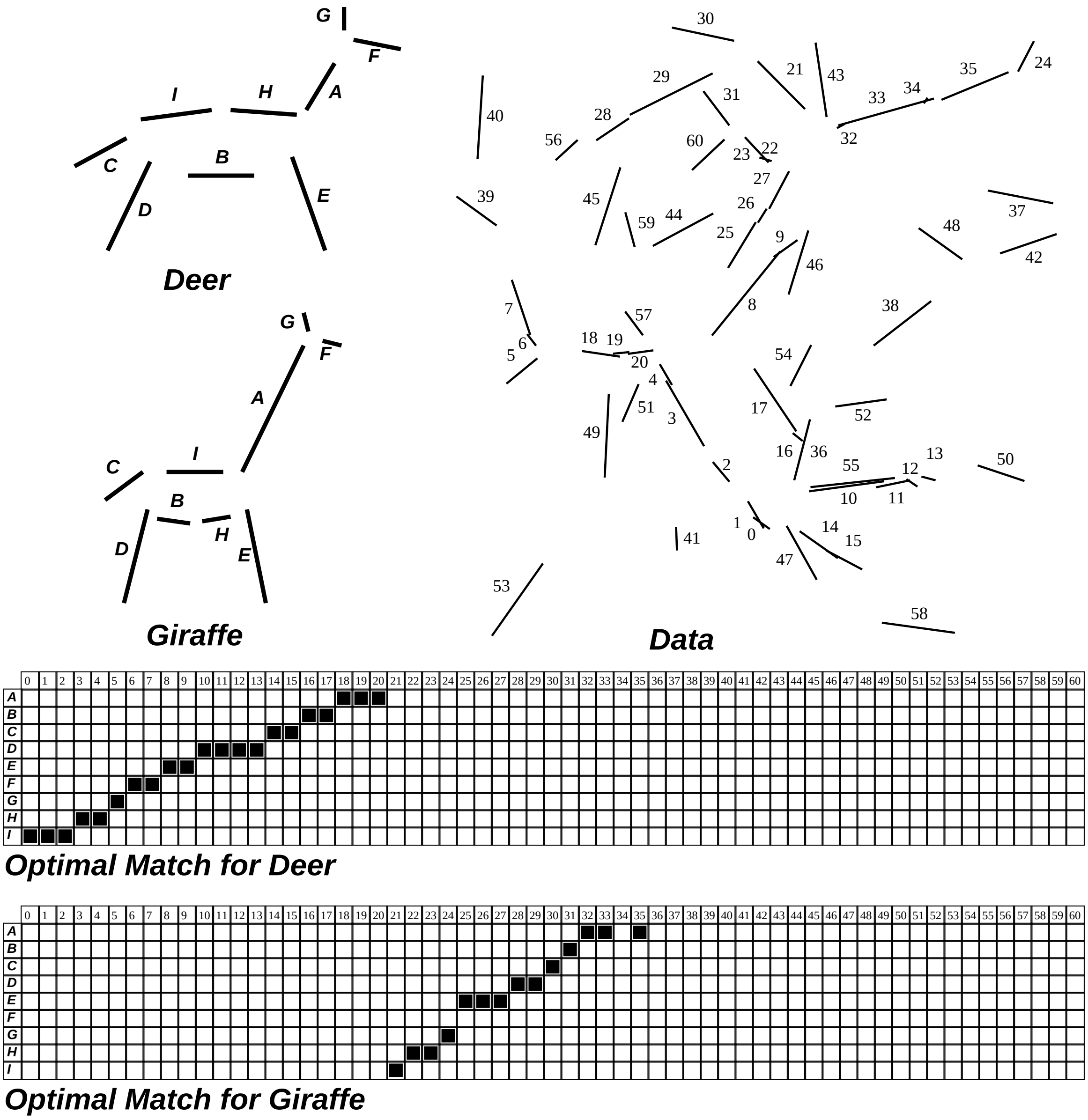


Figure 5.14 Subset-convergent local search finds both the Deer and Giraffe. The Deer is found 15/100 times and the Giraffe 8/100 times. The optimal correspondence mappings are indicated by the tables: filled in squares indicate matched segments. Readers unable to see the matches may look ahead to Figure 5.15

algorithm is able to pick 4 disjoint pairs of segments. This is different from the rectangle, for which the subsets shared model segments.

In 100 independent trials, subset-convergent local search found the best match for the Deer model 15 times, and the best match for the Giraffe model 8 times. This means that roughly 15% and 8% of the initial matches respectively lead the to globally optimal match. Considering the search space contains 2^{549} possible correspondence mappings, this result is both encouraging and perhaps might be a little surprising to some readers.

The initial random correspondences for these experiments were not drawn uniformly from the space of all possible correspondences, but were instead selected using the biased selection process described in Section 5.3.4. In particular, the target number of pairs per model segments r was set to 10. This tends to 'flood' the initial match and encourage local search to begin by removing those pairs which seem most unlikely to match. It isn't obvious whether in general such flooding actually helps. Qualitatively similar results to these will be shown in Chapter 6 using $r = 2$ rather than $r = 10$, but a full empirical study of how best to select r has not been conducted.

5.4.5 The Deer and Giraffe are Hard to See

Many people have found recognizing the Deer and the Giraffe in Figure 5.14 to be difficult. This observation has no direct bearing on the practical utility of local search matching. These algorithms are intended to provide practical solutions to a broad range of geometric matching problems, and were never intended as a model of human vision. However, it is interesting to discover that these probabilistic techniques readily solve problems which are difficult for people.

To be a bit more precise, people seem to have to study the image data for up to several minutes before they find both animal figures. For readers who find this true, Figure 5.15 highlights the two matches. To offer a bit of speculation, this seems to support the claim of David Lowe [80] and others that discernment of key-features plays a crucial role in human perception.

Lowe argues that people rely upon finding discernible, distinctive, features. He likewise argues for computer vision algorithms which do the same. A likely explanation for why people have trouble finding the two animals in Figure 5.14 is that the clutter interferes with people's ability to discern such features. This also suggests that key-feature based computer vision algorithms might suffer a similar fate in some contexts.

This speculation underscores the fact that local search matching *is not* a key-feature approach to matching. It is possible in a first reading to confuse the subsets used in subset-convergent search with key-features. However, the difference between these subsets and key-features is fundamental. Key-features are structures searched for independently in an image. If for some reason this search fails, then recognition fails. In contrast, the role of subsets in subset-convergent local search is to provide a mechanism for breaking out of Hamming-distance-1 local optima. The subset features are not searched for independently, and it does not matter whether or not they are distinctive.

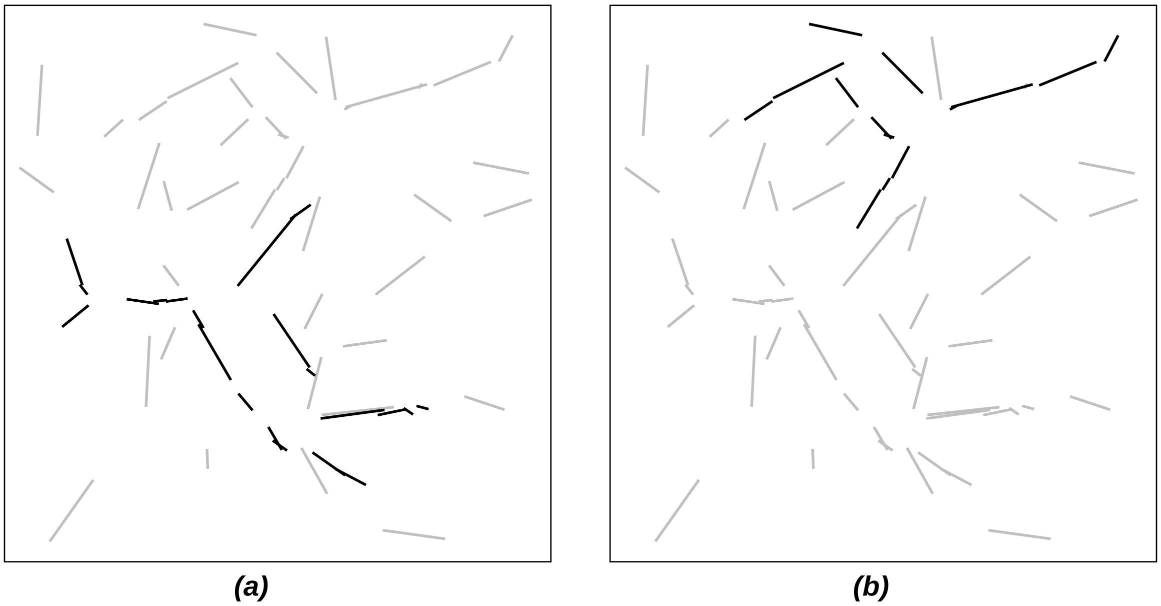


Figure 5.15 Showing the Deer and Giraffe matches. a) data segments matching the Deer, b) data segments matching the Giraffe.

5.5 Conclusion

Now having seen the examples in this chapter, it should be clear why so much attention was given to the problem of fitting models to data in Chapters 3 and 4. Without the ability to fit a model, and therefore accurately assess the global consistency of a hypothesized correspondence mapping, none of what has been presented in this chapter would be possible. It is worth again drawing attention to the subtle interplay between fitting and search: recognizing while locating is a critical aspect of local search matching. In particular, recall the square example from Figures 5.9 and 5.10 how the evolving best-fit of the model was absolutely critical to local search discovering a path from the less desirable match indicated in row 8 to the globally optimal match in row 16.

The subset-convergent algorithm developed in Section 5.4 is a novel and significant contribution of this thesis. It capitalizes upon the principal attribute of geometric matching not shared by most other commonly studied combinatorial optimization problems, the underlying coupling of evaluation through a single pose transformation. In contrast, in the Traveling Salesperson problem (TSP), when two cities are swapped in a tour, the change to the overall length of the tour depends only upon segments of the tour directly associated with these two cities. There is a locality of effect which is lacking in geometric matching. In geometric matching, when a pair of segments is added or removed from a match, the best-fit pose changes, and the match error associated with *all* other pairs of segments change.

Global coupling through the best-fit pose computation is a factor completely missing from problems such as TSP [76] and graph partitioning [64]. This coupling through the best-fit pose is one of the factors which makes geometric matching difficult. However, by testing for pose consistency between subsets of a match, subset-convergent local search takes this complicating factor and turns it into an asset. This suggests there are probably other geometric problems which share this basic property with the line segment matching problem studied in this thesis, and that subset-convergent local search may be an effective and general tool for solving such problems.

Finally, it is important to understand that random sampling not only probabilistically solves geometric matching problems, it offers a principled way of studying problem difficulty. The crucial question facing anyone wishing to use probabilistic local search is: 'how many trials must be run?' Sections 5.3.2 and 5.3.3 have shown that the number of trials needed depends upon how many states in the search space lie on the globally optimal tree and the likelihood of landing on this tree given the random sampling procedure. Sections 5.3.1 has shown this can be rigorously estimated for individual problems and generalized to problem domains given a representative suite of test problems.

This empirical approach to characterizing algorithm performance is extremely attractive when one considers that analytically predicting performance under other than idealized assumptions has been a major problem for the computer vision community. Grimson [45, 48, 49] has analyzed the performance of several general matching techniques, and these analyses are insightful and useful. However, in places Grimson has been forced to make questionably simplistic assumptions; for example assuming that clutter in an image is randomly distributed and is therefore not structured.

CHAPTER 6

How Easy is Local Search Matching

6.1 Introduction

This chapter explores how well the subset-convergent local search algorithm, developed in the previous chapter, performs on a test suite of geometric matching problems. Several aspects of performance are considered. One is sensitivity to changes in the geometric form of the model and image data. The algorithm does well in this regard, reliably finding simple models, moderately complex models, and partially symmetric models. Another aspect of performance is sensitivity to 'tuning'. To put it differently, consider whether it is necessary to make changes to the algorithm in order to solve problems of differing form or size. Here too, the algorithm does well, with the same set of parameters able to solve all problems in the test suite. However, changing the match error does change how local search performs, and this connection is explored.

In general, how performance varies with problem size is a major consideration. In particular, how much longer does it take to reliably solve larger matching problem? Recall that the size of the search space grows as 2^n, where n is the number of potentially matching line segments. It may therefore surprise some readers that over the range of problems studied for this chapter, no evidence of an 'exponential wall' has been found. Quite to the contrary, for the test suite of problems, required computation rises steadily and average growth is bounded by n^2.

It is conjectured that the average case computational demand of local search matching developed in this thesis is bounded by n^2. Obviously, this conjecture is based upon an optimistic interpretation of the data. It is always possible that the computational demand of the algorithm becomes exponential just beyond the range tested. As partial justification for this optimistic conjecture, performance has been measured for n ranging over two orders of magnitude. This chapter presents results for n between 12 and $1,296$. More strikingly, this means the search spaces range in size from 2^{12} to $2^{1,296}$. If there were an exponential wall out there, it can be argued that these experiments should have found it.

Ultimately, formal analysis will be required to prove a complexity bound. However, such analysis is very difficult. There is a developing study of complexity issues associated with local search. For example, Johnson and Papadimitriou [62] consider questions such as whether a state may be determined to be a local optima in polynomial time. [1] However, quoting from a recent paper by Krentel [67] on the topic of local search and complexity:

[1] Johnson and Papadimitriou are thanked for inspiring the title to this chapter with their own title: 'How Easy is Local Search?'

> Although these methods seem to converge quickly in practice, very little is known about them theoretically, or more generally, about the complexity of finding locally optimal solutions.

Analytically deriving complexity bounds for specific local search algorithms applied to specific problems, such as TSP or geometric matching, is complicated by several factors. Analytically capturing the problem-specific nature of the objective function, the local neighborhood, and the interplay between them is particularly difficult. As discussed in Section 5.3.2 of chapter 5, the number of local optima in the space doesn't matter. What matters is the probability that a randomly selected initial state leads to an optimal state, and that traversing this search path requires a predictable and tractable amount of computation. As Tovey [106] suggests, it is difficult to derive this probability through formal analysis. Given these obstacles to formal analysis, this chapter adopts the common approach [60, 61] of empirically studying algorithm performance.

An n^2 bound on computation, although profoundly better than an exponential bound, still has a practical down side: even a parabola begins to look like a wall as n gets large. This rate of growth places a practical ceiling upon the size of problem to be solved. These limits, with different constant factors, are shared by the other algorithms summarized in Section 2.4.9 of Chapter 2, so subset-convergent local search is not alone in this regard.

6.2 A Suite of Test Problems

A test suite of matching problems based upon 6 geometric object models and 48 image data sets provides a controlled test of the subset-convergent algorithm. The 6 geometric models are shown in Figure 6.1, and will be used in this chapter to empirically explore the performance of the subset-convergent algorithm. These models have been chosen to test subset-convergent local search on a variety of geometric forms, some of which are known to be problematic for other matching algorithms. The Pole, for example, is interesting because of its extreme simplicity. This model lacks the distinctive structure required by key-feature and geometric hashing approaches. As will be seen, instances of this model are readily matched using subset-convergent local search.

The Dandelion is interesting because it is has an approximate 16 fold rotational symmetry. Only the stem gives it a distinctive orientation. To put it another way, it is possible to partially match the Dandelion with itself in each of 16 distinct orientations. From the standpoint of fit and omission, 15 of these matches are nearly as good as the one true match. Nearly symmetric models of this form are a particularly demanding test of any matching algorithm, yet subset-convergent local search does successfully find the correct globally optimal match in problems involving this model. However, when the problems of symmetry is compounded by the presence of multiple instances of the model in the image data, then subset-convergent local search is only reliable if the match error function includes the pairwise error term defined in Section 3.6.1.

The polygonal curve approximation in the Leaf is interesting because it leads to matching problems requiring many-to-many mappings between model and image segments. As illustrated with a simpler example in Figure 3.5 in Chapter 3, break points on the piecewise linear curve approximation of the model are different from the break points in the image data. As discussed in Chapter 2, solving for many-to-many correspondences is an important strength of the local search approach not shared by other algorithms.

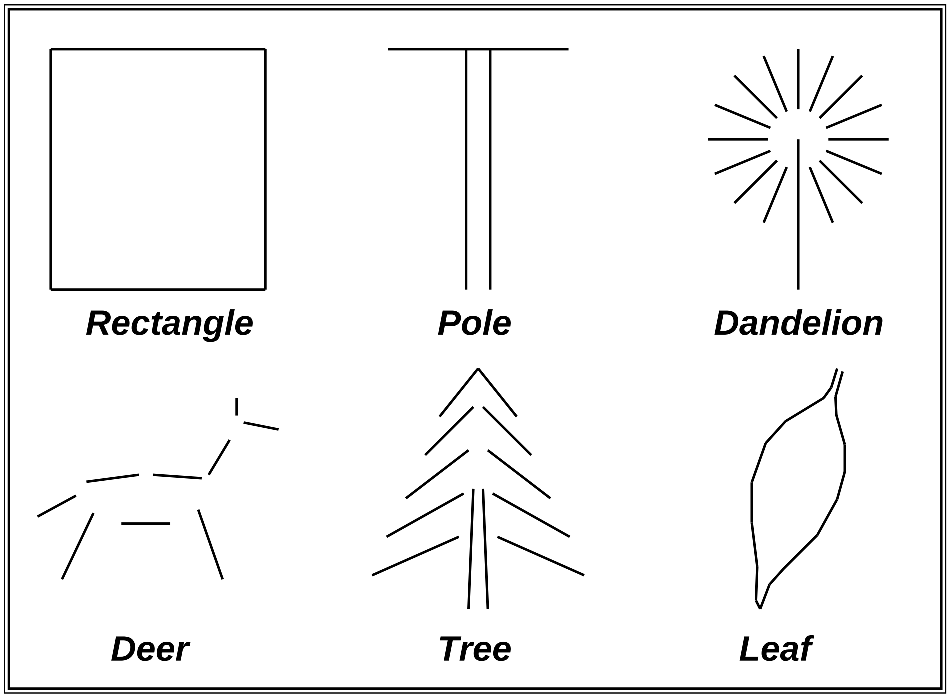


Figure 6.1 Six geometric object models. Some of these are problematic for approaches other than local search. The simplicity of the pole means it lacks distinctive structure and hence will not be recognizable using key-feature or geometric hashing. The 16 fold rotational symmetry of the Dandelion will confound both key-feature and constraint-based tree search approaches. The polygonal curve approximation in the Leaf requires many-to-many mappings between model and image features.

Instances of these models are corrupted in order to create synthetic test data. A synthetic problem generator was developed to provide a controlled means of corrupting model instances and adding clutter segments. The problem generator begins by randomly rotating, translating and scaling an instance of a model. It then randomly fragments and skews model segments. Finally, it adds zero or more random clutter segments. The extent of corruption is controlled by parameters of the problem generator, such as probability of fragmentation per unit length of model line segment, and standard deviation of endpoint skewing. The matching problem is then to successfully find the model instance in the corrupted data. Figures 6.2 and 6.3 show this synthetic image data. There are 4 sets of image data for each model containing 0, 10, 20 and 30 randomly distributed background clutter segments.

Randomly distributed clutter is often considered the norm for algorithm evaluation [80, 48], but it may be misleading. Clutter is, more often than not, the result of unmodeled structure in a scene. In the extreme, clutter mimics the object model itself, and leads to problems in which there are multiple instances of the same model present in the image data. If the additional instances are incomplete, then they are suboptimal. Alternatively, if two or more instances are of comparable quality, then they represent comparable local optima.

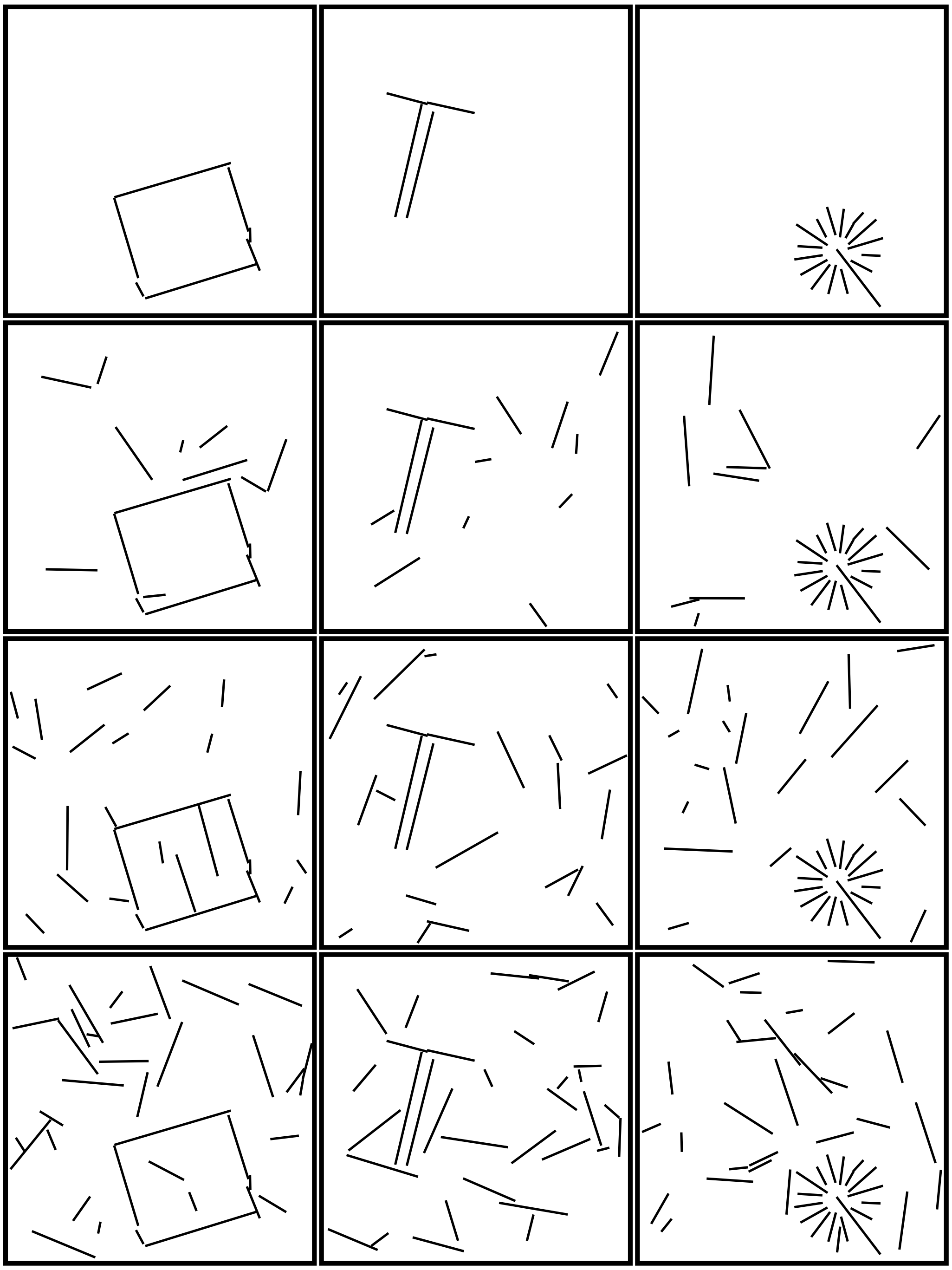


Figure 6.2 Rectangle, Pole and Dandelion with random clutter.

Figure 6.3 Deer, Tree and Leaf with random clutter.

Multiple instance problems are known to be difficult for tree search algorithms [48], and can be expected to cause problems for key-feature and geometric hashing approaches.

The synthetic problem generator was also used to produce 24 problems with multiple corrupted instances of the same model. The resulting image data is shown in Figures 6.4 and 6.5. This represents the opposite extreme from random clutter, with additional segments mimicing the structure of the model. To generate this data, the synthetic problem generator produces between 1 and 4 corrupted copies of the model. The first instance, which repeats in each of the 4 sets, is generated with less severe corruption parameters, and hence is expected to be the globally optimal match. As will be seen, subset-convergent local search readily finds the globally optimal matches in these 24 problems.

6.3 Experiment and Algorithm Setup

To observe the performance of the subset-convergent local search algorithm, parameters such as the weighting coefficients in the match error are fixed, and then 100 trials are run on each of the 48 data sets presented in Figures 6.2 through 6.5. The matching problem is to find the globally optimal instance of the appropriate model in the corrupted image data. Unlike some of the problems presented earlier, no initial assumptions about the pose of the model are used to filter the set of possibly matching pairs of segments. In other words, the set of candidate pairs is the cross product of the sets of model and data segments: $S = M \times D$.

For each run of 100 trials, the resulting matches are recorded and ranked. The key statistic is how often search finds the globally optimal match, and this statistic is reported in Section 6.5.1. However, much can be learned by studying the nature of the sub-optimal matches, and Section 6.4 considers in greater detail the results for 2 of the sets of image data. Also of interest is the total time required to confidently solve each problem, and these time estimates are reported in Section 6.6.

There are two dimensions along which to measure performance, problem variation and algorithm tuning. The aim of this chapter is to study variation over a range of different problems, not to make a thorough study of possible algorithm parameter settings. However, changing the objective function does modify performance, and results for the complete suite of 48 problems are presented for two different match error settings. These will be called 'case 1' and 'case 2'.

Case 1 and 2 differ in two respects. In case 1, the maximum displacement parameter σ is set to 2.0, while in case 2, σ is set to 5.0. Remember from Section 3.5 that σ controls how far a data line segment may be displaced from a model segment and still be considered a good match. The other difference is that in case 1 the pairwise error term described in Section 3.6.1 is included in the match error, while in case 2 it is not. To summarize:

Case 1 Maximum-displacement $\sigma = 2$, pairwise error term.

Case 2 Maximum-displacement $\sigma = 5$, no pairwise error term.

Ideally these changes would be studied independently as well as together, giving rise to four rather than two cases. However, running 100 trials of subset-convergent local search on the 48 data sets for the two cases took over 750 hours, or about a month, on a TI Explorer II Lisp Machine. Time did not permit the testing of additional cases. Since a new C version of

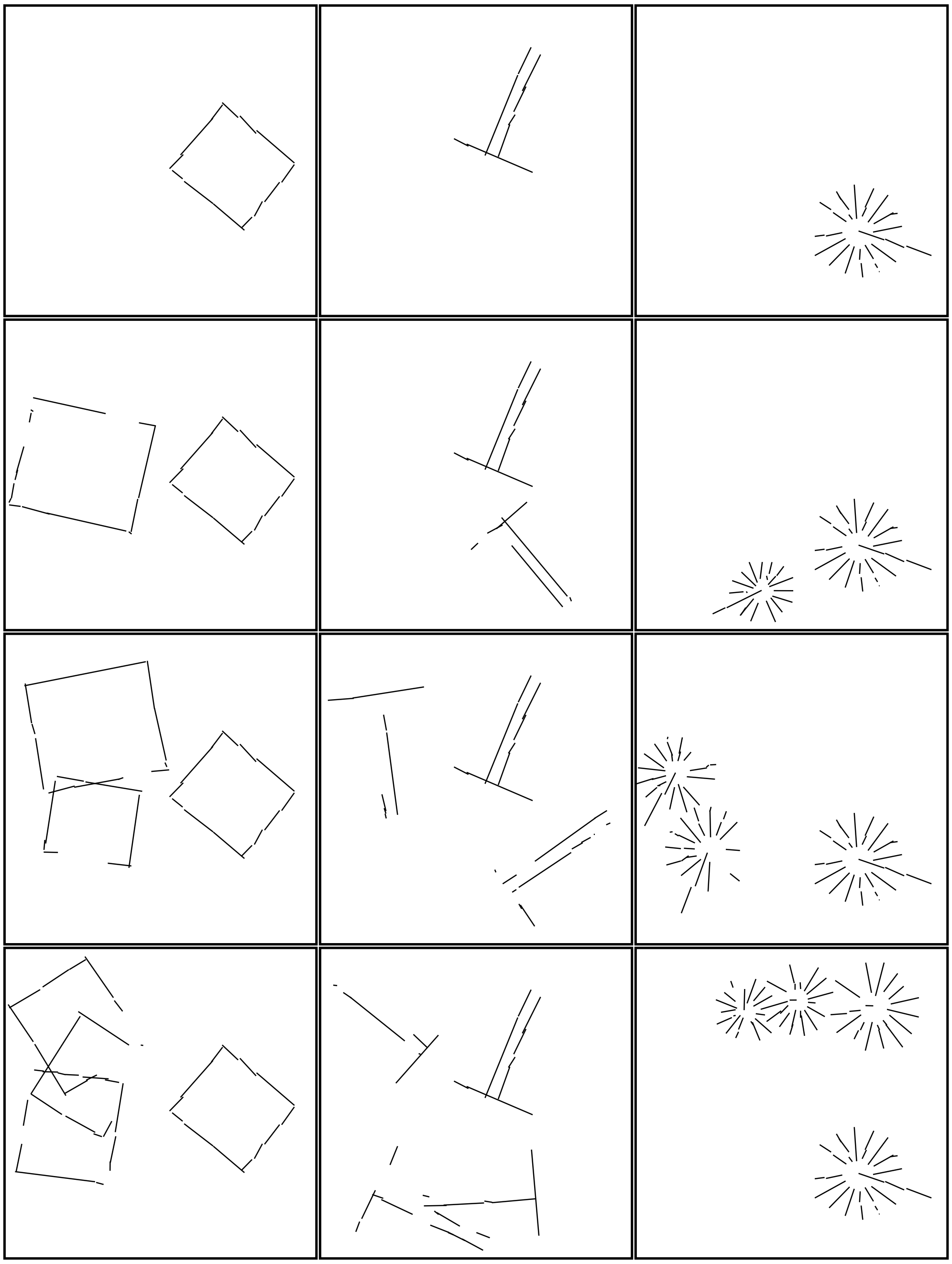


Figure 6.4 Multiple instances of the Rectangle, Pole and Dandelion.

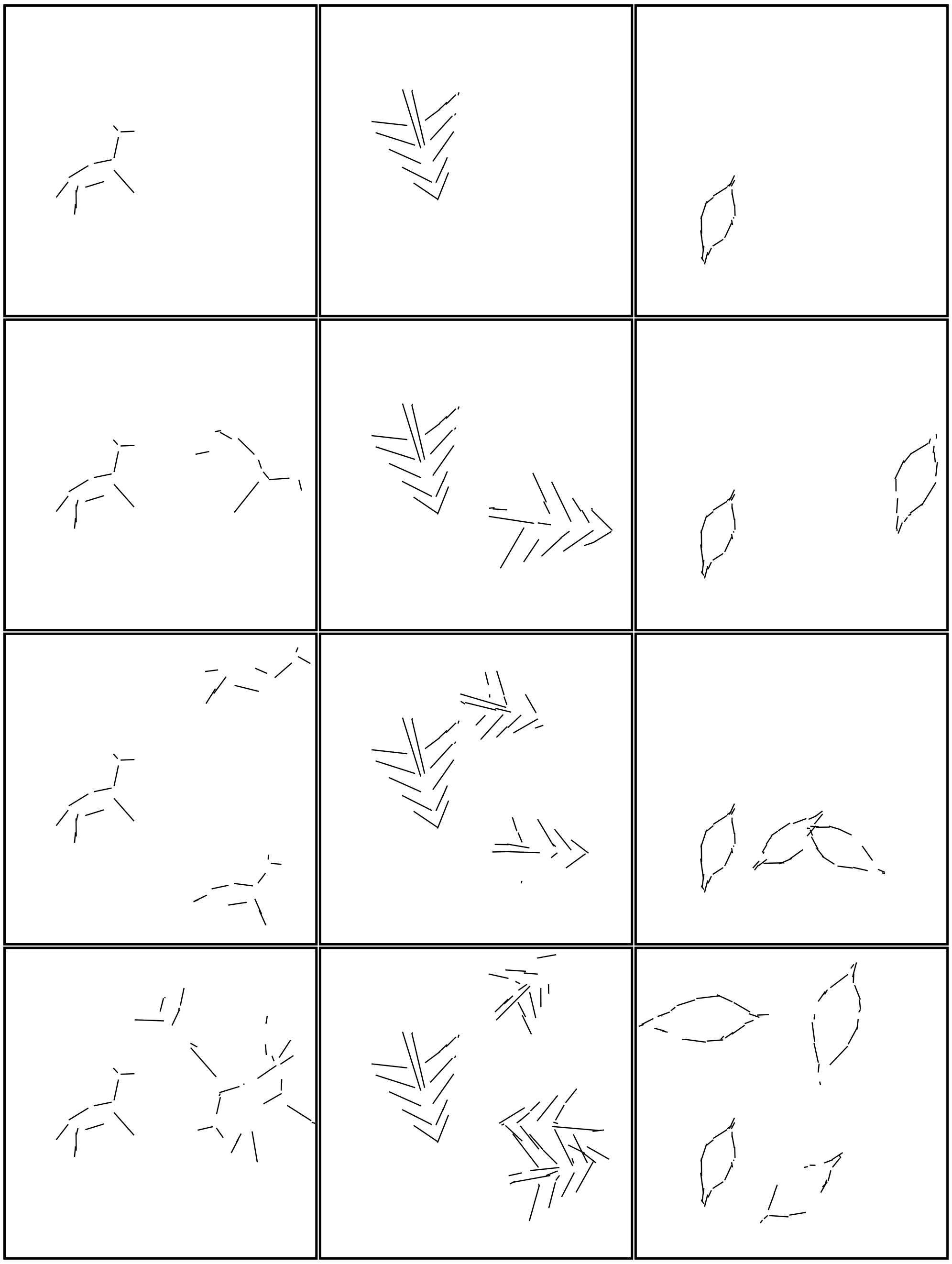


Figure 6.5 Multiple instances of the Deer, Tree and Leaf.

the matching algorithms appears to run roughly 60 times faster, additional studies will be made in the near future.

The following parameters are set identically for both case 1 and case 2. Choices are based upon past experience with the algorithm and are adequate but not necessarily optimal. The omission attenuation parameter a, described (Section 3.4) is set to 0.75. This setting favors matches with more model segments partially matched than with fewer model segments completely matched. The regularization weight τ (Section 4.5) for resolving otherwise underdetermined best-fit model pose is set to 10^{-4}. This setting reliably resolves otherwise pose-ambiguous matches, while not introducing any noticeable midpoint-to-midpoint bias in the fitting. The scale transformation error r (Section 3.6.2) is used to penalize matches which make the model either to large or to small. The parameter is set to 2, and thus a penalty is exacted if the model shrinks below half its original size or grows to more than twice its original size. The binding-probability used to bias the selection of initial random matches (Section 5.3.4) is set to bind, on average, 4 data segments to each model segment. This appears to be a reasonable setting when all possible pairs of segments are candidate matches. Finally, to make local search more efficient, the omission error is only computed for an entire model when the change in fit error relative to the local change in omission suggests it is worthwhile to do so. This is the local omission heuristic described in Section 5.2.

In case 1, the pairwise error term (Section 3.6.1) is configured with a lower bound of $\theta_l = 8$ degrees and an upper bound of $\theta_u = 16$ degrees. When matched segments differ in relative orientation by less than 8 degrees, there is no penalty, while as above 8 degrees, a penalty is added, and this penalty reaches 1.0 when relative orientation reaches 16 degrees. Because decreased omission cannot make up for a penalty of 1.0, segments differing in orientation by more than $\theta_u = 16$ degrees will never be included in a locally optimal match.

6.4 A Look at Local Optima for 2 Select Problems

This section examines in detail the results for 2 of the 48 matching problems for each of the two match error cases. The two problems are for the Tree and Dandelion models without any random clutter, and are shown in the upper left of Figure 6.3. Specific matches will be shown along with histogram information showing the relative frequency and quality of the locally optimal matches found by subset-convergent local search.

Figure 6.6 shows the globally optimal match for the Tree matching problem. As with all the problems studied in this chapter, all pairs of segments are candidates for matching. The absolute match error differs depending upon the setting of σ, with $E_{\text{match}} = 0.042$ for $\sigma = 2.0$ and $E_{\text{match}} = 0.035$ for $\sigma = 5.0$. The larger σ value not only increases the acceptable displacement between corresponding segments, it also discounts the fit error and yields an overall lower match error.

The 2nd and 3rd ranked locally optimal matches for match error case 1 are interesting for what they suggest about the partial symmetry of the Tree. These two matches are shown in Figure 6.7. The branches of the Tree form a repeated chevron pattern, and as these two locally optimal matches show, the model can be shifted up or down one in this pattern and still be a moderately good match to the data.

The histograms in Figure 6.8 summarize E_{match} and the frequency of all the locally

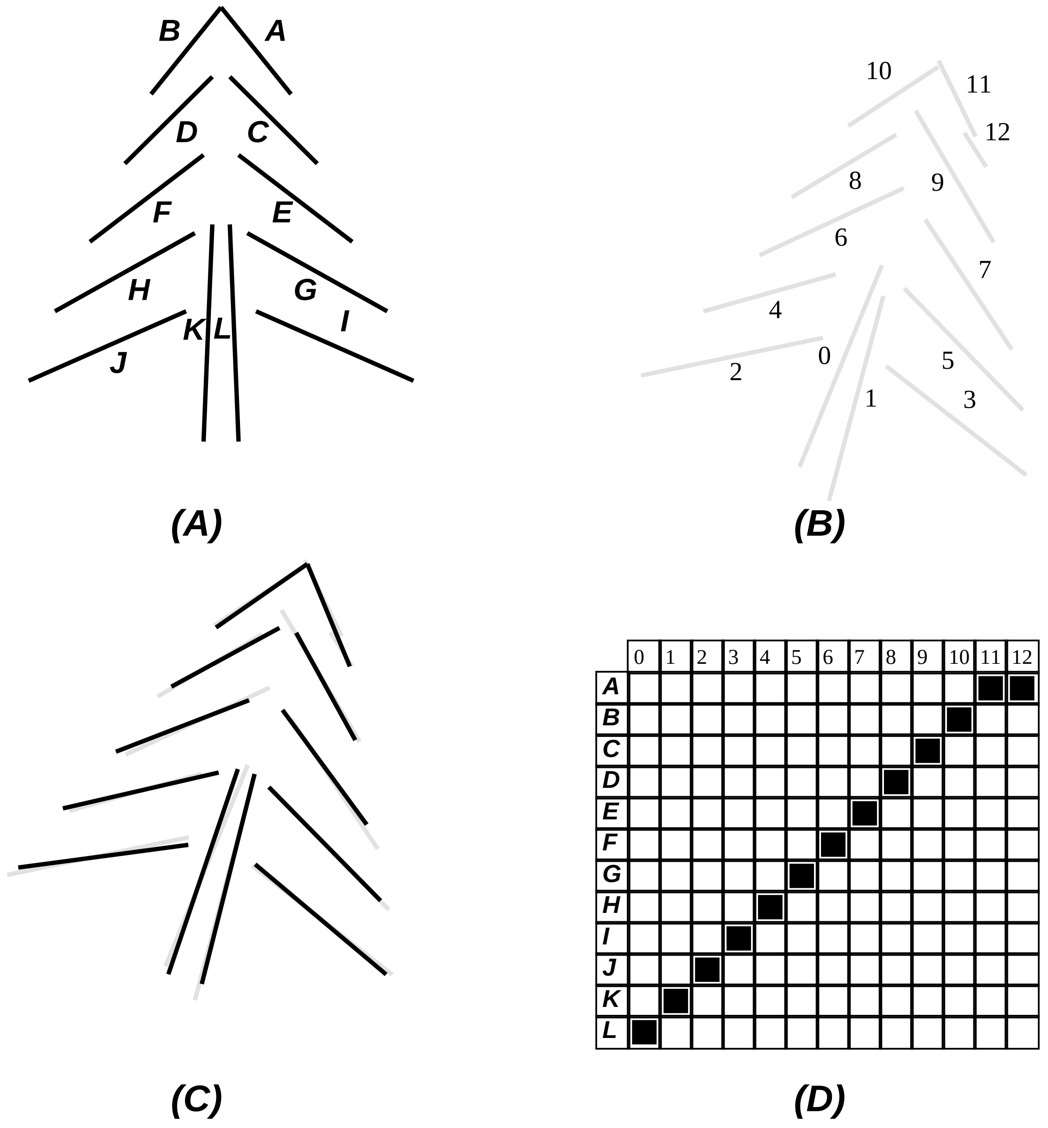


Figure 6.6 Globally optimal match for the Tree model. This match is best for both match error case 1 and 2. a) Model segments labeled, b) data segments labeled, c) the model fit to the data, d) the correspondence mapping. Filled in squares indicate matching segments. $E_{\text{match}} = 0.042$ when $\sigma = 2.0$ and $E_{\text{match}} = 0.035$ when $\sigma = 5.0$.

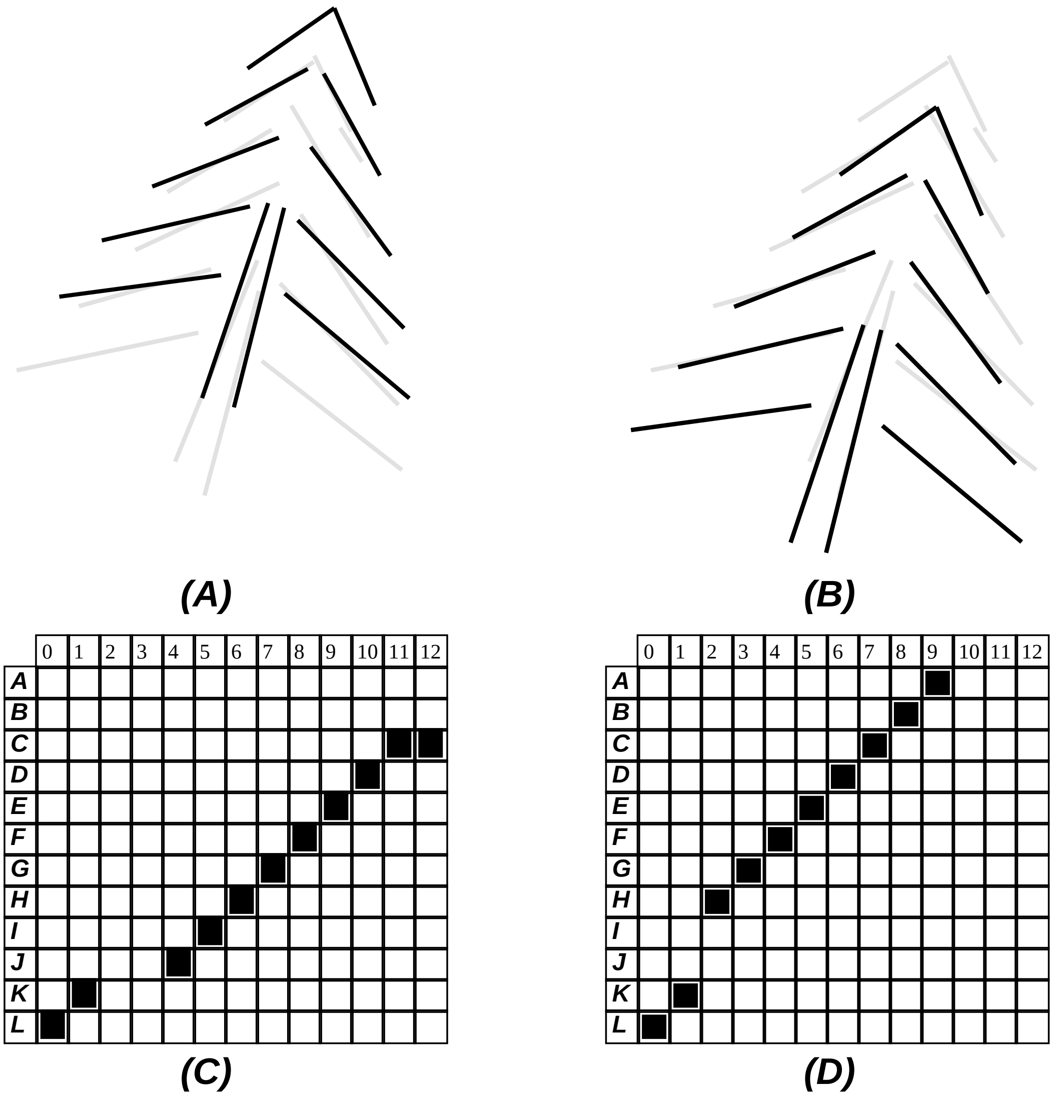


Figure 6.7 The 2nd and 3rd ranked Tree matches for case 1 with $\sigma = 2$. a) The model fit to the data for the 2nd best match, b) the model fit to the data for the 3rd best match, c) the correspondence mapping for the 2nd best match, d) the correspondence mapping for the 3rd best match. The match errors are $E_{\text{match}} = 0.304$ and $E_{\text{match}} = 0.316$ respectively.

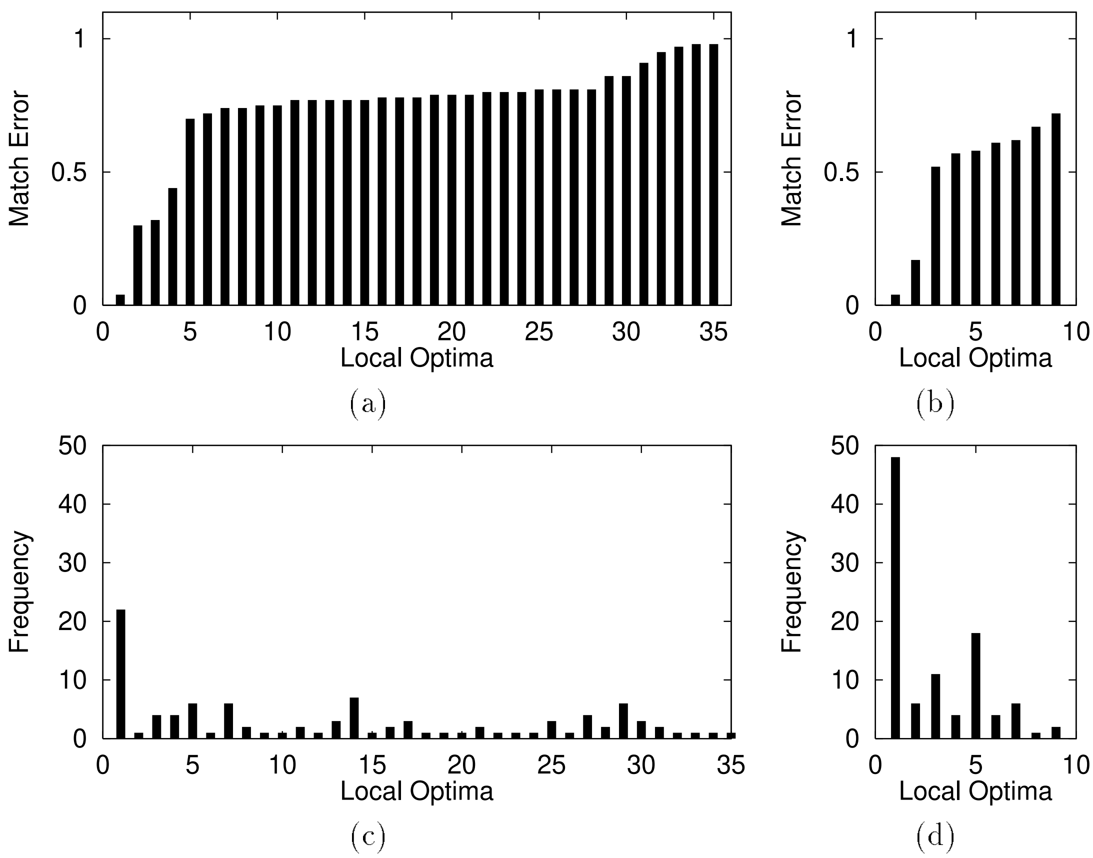


Figure 6.8 Histograms ranking local optima for the Tree model. Local optima are ranked along the x axes by ascending match error: a) Case 1 E_{match} for local optima, b) Case 2 E_{match} for local optima, c) Case 1 frequency of occurrence, d) Case 2 frequency of occurrence.

optimal matches found in 100 trials on the Tree model. Figures 6.8a and 6.8b show E_{match} for the local optima found for match error cases 1 and 2 respectively. Figures 6.8c and 6.8d show the associated frequency of occurrence for each of the local optima. With the increase in σ moving from case 1 to 2, the number of locally optimal matches falls from 34 to 9, and the frequency with which the globally optimal match is found rises from 22 to 48.

The change in the match error alters the frequency with which the globally optimal match is found. In addition, it changes the remaining locally optimal matches. The 2nd ranked local optima differ between match error case 1 and 2. The increased σ for case 2 causes pairs $(A, 11)$ and $(B, 10)$ to be added to the match shown in Figure 6.7a. Another change is that the match shown in Figures 6.7b and 6.7d disappears in case 2, in that it is not found by any of the 100 trials. There are two possible explanations for the dissappearance of the match in Figures 6.7b and 6.7d. First, it could be that changing σ dramatically reduces the number of paths in the search space leading to this match. Alternatively, it is possible that the change in σ has opened up a path between this match and the globally optimal match, and thereby made the match no longer locally optimal.

In general, increasing σ makes local search less anxious to remove pairs of segments. In the limiting case of $\sigma = 0$, local search will simply remove pairs of segments until there are

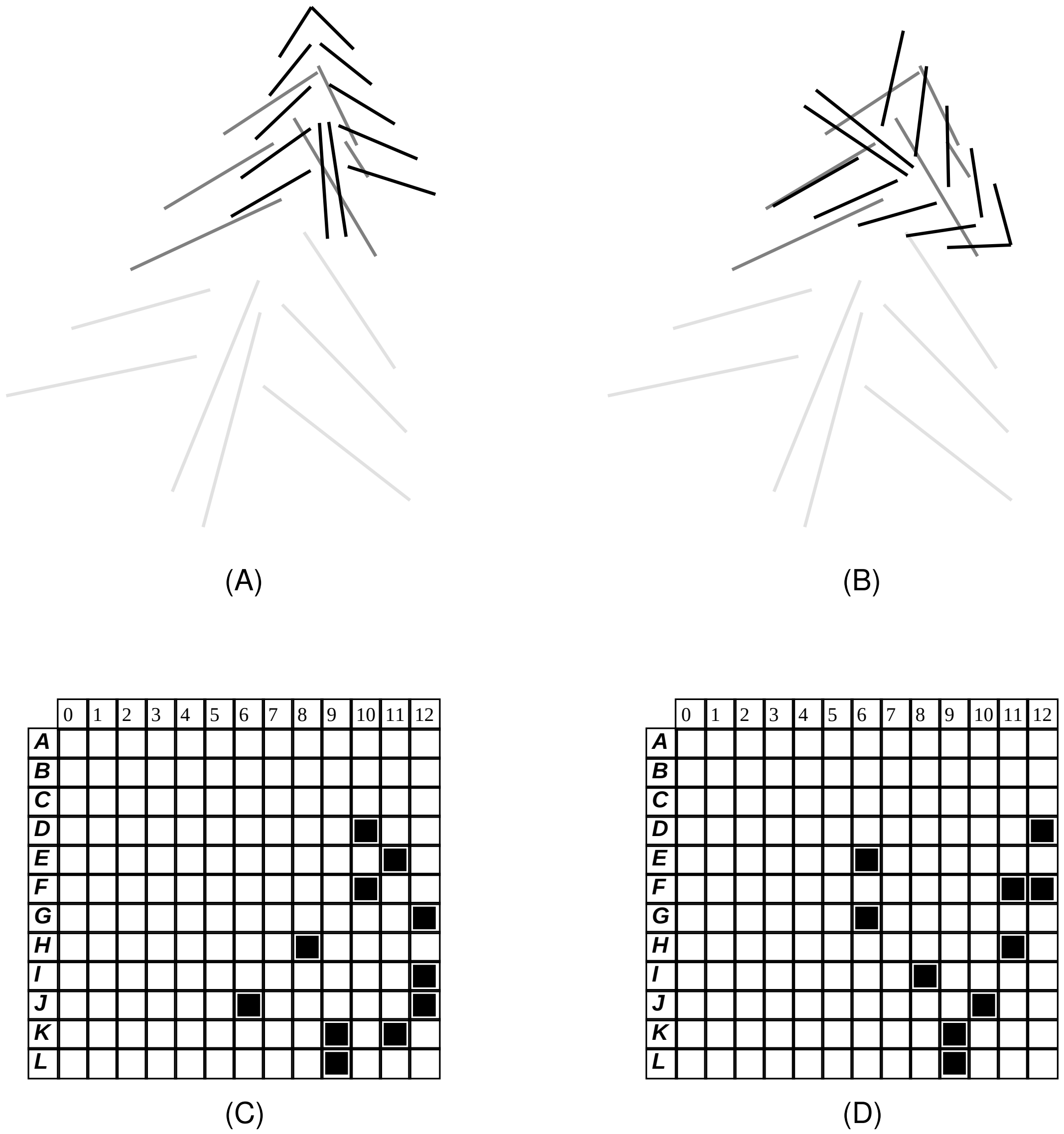


Figure 6.9 For match error case 2, the 3rd and 4th ranked tree matches. a) 3rd ranked match with $E_{\text{match}} = 0.523$, b) 4th ranked match with $E_{\text{match}} = 0.575$, c) 3rd ranked correspondence mapping, d) 4th ranked correspondence mapping.

none left to remove. [2] The higher σ, the sooner local search may begin to start adding pairs, and this may alter significantly the path taken by local search.

Most locally optimal matches are not as interesting as those shown. The vast majority represent poorly matching collections of segments which seldom account, in the omission sense, for more than 50% of the model. To illustrate, Figure 6.9 shows the 3rd and 4th ranked matches for match error case 2. In each case, the trunk of the Tree, segments K and L, have become stuck upon data segment 9. The remaining model segments attach themselves to nearby data where possible. The most noteworthy attribute of these matches is that subset-convergent local search is unable to find a path leading from them to one which is better.

Figure 6.7 has already shown one situation in which local search interacts with the partial symmetries of an object model to produce interesting local optima. The Dandelion model provides an even more compelling example. Figure 6.10 shows sixteen distinct matches between the Dandelion model and the data shown in the upper left corner of Figure 6.5. If one scans Figure 6.10 from left to right, and from top to bottom, the Dandelion rotates clockwise. The final match, in the bottom right, is the globally optimal match.

Histograms in Figure 6.11 summarize the match error and frequency of occurrence for the local optima encountered in the Dandelion matching problem. Results for both the match error cases are shown, and the layout of Figure 6.11 is the same as Figure 6.8. A quick comparison of the histograms in Figure 6.8 and Figure 6.11 reveals significant differences. For the Tree, the globally optimal match is significantly better than the other local optima, while for the Dandelion there is a whole set of fairly good matches. In Figure 6.11a, the difference between the first 18 local optima and the remaining 7 is particularly striking. These first 18 local optima are a consequence of the rotational symmetry illustrated in Figure 6.10. There are more than 16 reasonably good local optima because the corrupted data segments happen to generate 2 additional locally optimal matches which are combinations of the other 16.

The match error distributions for the Dandelion are quite atypical, while the distributions for the Tree are typical. The data shown in Figures 6.2 through 6.5 generally supports only a small number of good matches, and all others are considerably worse. Partially symmetric models and multiple model instances are the two factors which generate multiple local optimal of reasonable quality, and these circumstances conspire most dramatically in the matching problem involving 4 instances of the Dandelion (shown in the bottom right of Figure 6.4). Each instance generates at least 16 local optima of reasonable quality, yielding at least 64 reasonable local optima for this problem. There are also local optima with no meaningful geometric interpretation. Evidence of such local optima is seen in the 7 worst local optima plotted in Figure 6.11a.

There is an encouraging trend evident in the frequency of occurrence histograms in Figures 6.8 and 6.11: the best match is found more frequently than any other. There are no guarantees that this will be the case, but the following section shows it is the rule rather than the exception.

---

[2] To be precise, it will remove pairs until no additional pair may be removed without producing an undefined best-fit pose.

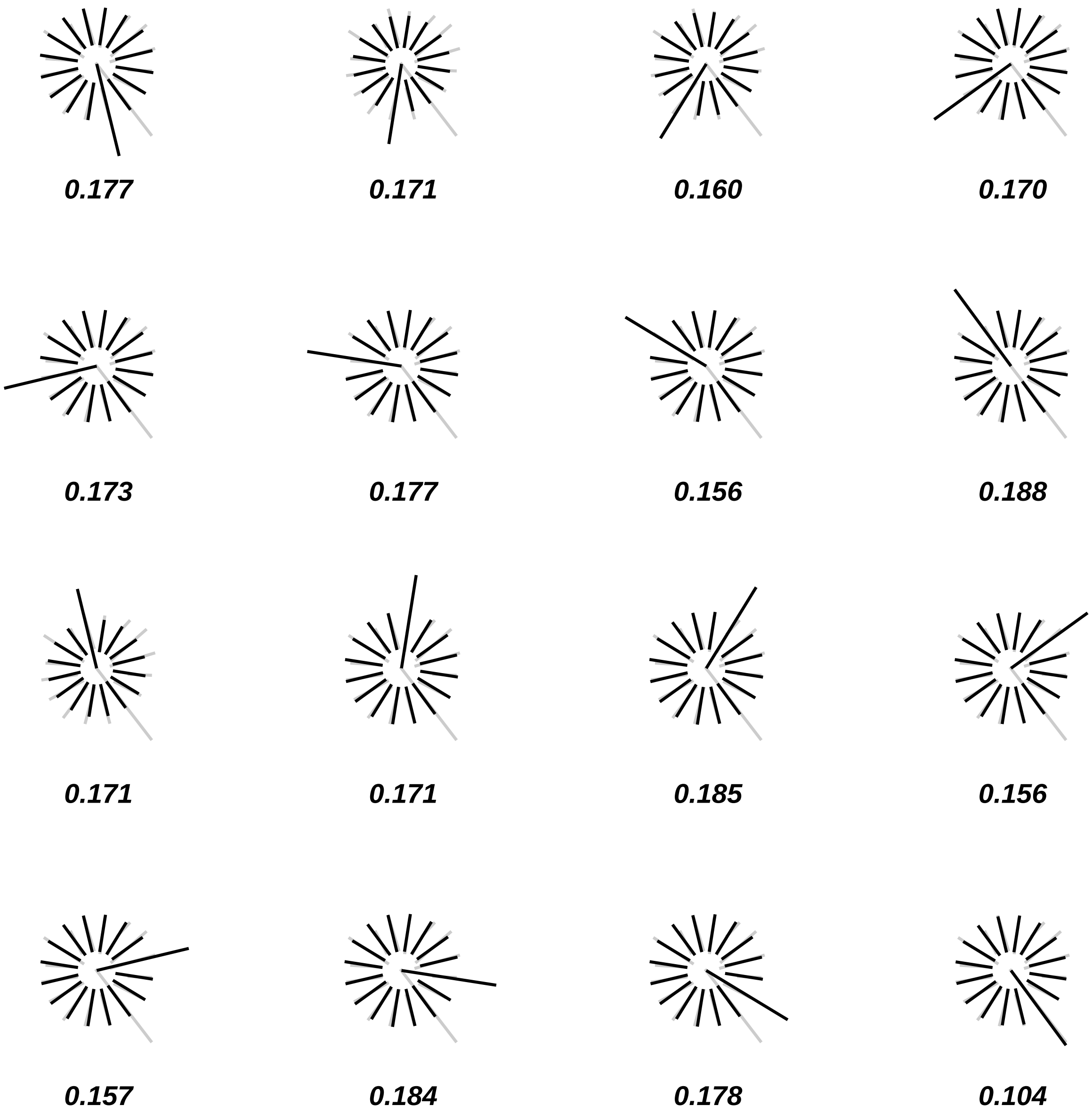


Figure 6.10 Sixteen locally optimal matches for the Dandelion model. The model in black is shown in best-fit registration with the data in grey. The match error for each is shown, the best match is on the bottom right.

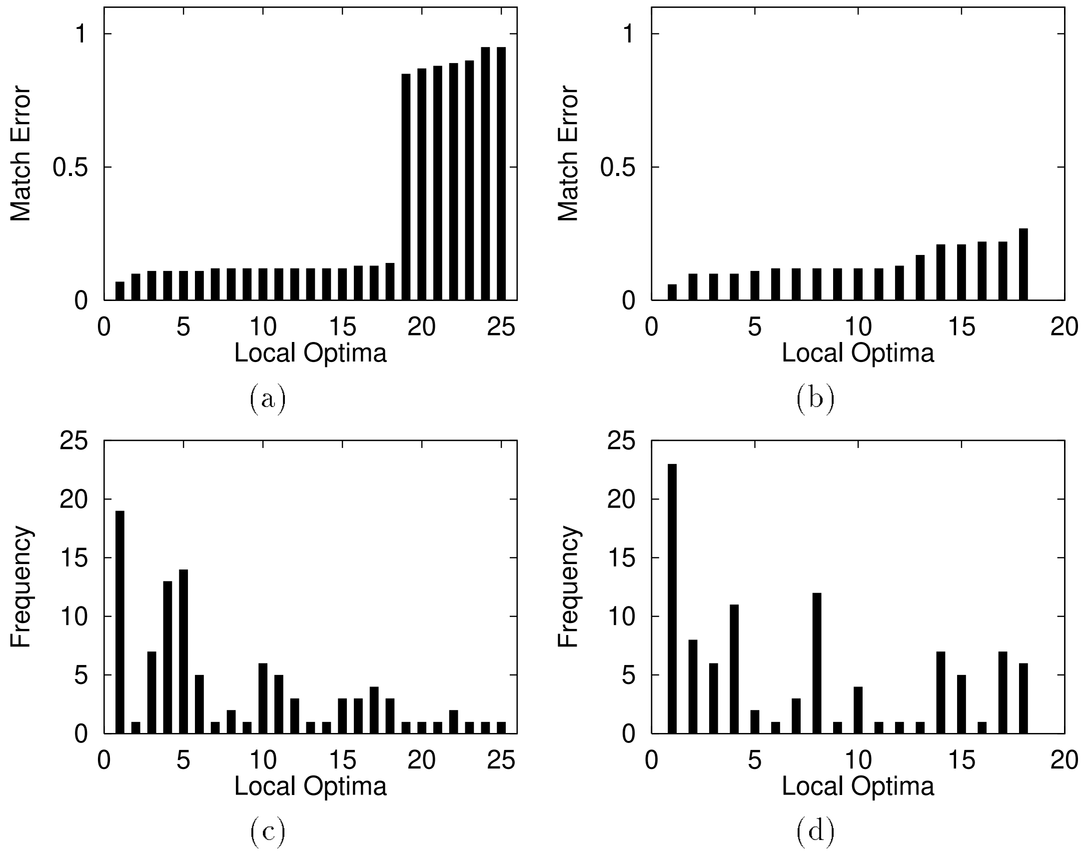


Figure 6.11 Histograms ranking local optima for the Dandelion model. Local optima are ranked along the x axes by ascending match error: a) Case 1 E_{match} for local optima, b) Case 2 E_{match} for local optima, c) Case 1 frequency of occurrence, d) Case 2 frequency of occurrence.

6.5 Performance Summary Over the Entire Test Suite

The previous section looked in detail at the performance of subset-convergent local search on 2 of the 48 matching problems associated with the data sets presented in Section 6.2. In this section, the performance of the subset-convergent algorithm will be summarized for all 48 matching problems and for the 2 match error cases. For the 96 distinct experiments, in which 100 trials of subset-convergent local search were run, this section reports the estimated probability of success, $\hat{P}_s$, and the associated required number of trials, $\hat{t}_s$, needed to confidently solve each problem.

Before moving on to the probability of success estimates, the previous section concluded that subset-convergent local search appears to find the global optima more frequently than any other local optima. This is true for 64 of the 96 experiments run. Table 6.1 shows the actual breakdown by match error case and matching problem.

Table 6.1 Best match as the the most frequently found optima. Problems for which the best match is found more frequently than any other local optima are indicated with a $\diamond$. Individual columns for the random clutter problems are labeled with the number of clutter lines added to the data. The individual columns for the multiple model instance problems are labeled with the number of model instances present in the data.

	Random Clutter								Multiple Instances							
	Case 1				Case 2				Case 1				Case 2			
Model	0	10	20	30	0	10	20	30	1	2	3	4	1	2	3	4
Rectangle	◇	◇	◇	◇	◇		◇		◇	◇	◇		◇	◇	◇	
Pole	◇	◇			◇	◇	◇		◇				◇	◇	◇	◇
Deer	◇	◇							◇	◇	◇	◇				
Tree	◇	◇	◇	◇	◇		◇		◇	◇			◇		◇	◇
Leaf	◇	◇	◇	◇	◇	◇	◇		◇	◇	◇	◇	◇	◇	◇	◇
Dandelion	◇	◇	◇	◇	◇				◇	◇	◇	◇				◇

6.5.1 *Probability of Success $\hat{P}_s$ and Required Trials $\hat{t}_s$*

Whether the globally optimal match is the most frequently encountered match is interesting. However, it is less interesting than the probability that local search initiated from a randomly selected match will find the globally optimal match. It is this latter probability of success which dictates how many trials must be run to confidently solve a problem, and by implication, the difficulty of the matching problem.

Recall from Section 5.3.1 that for a particular matching problem ω and local search algorithm ψ, the maximum likelihood estimate of the true probability of success, $\hat{P}_s$, is the ratio of the number of times the globally optimal match is found over the total number of trials run. The $\hat{P}_s$ estimates for each of the 48 matching problems associated with the data in Figures 6.2 through 6.5 and the two match error cases are based upon 100 trials of the subset-convergent local search algorithm. These estimates are show in Table 6.2a. The layout is the same as Table 6.1. No estimates are provided for the 3 and 4 instances Dandelion problems using match error case 2. As will be discussed below, for these two problems 100 trials was inadequate to reliably estimate $\hat{P}_s$. Subsequent additional trials suggest $\hat{P}_s$ is around 0.01 for the 3 instance problem.

The required number of trials may be computed from $\hat{P}_s$ using the relation defined in equation 5.5 on page 87. Here, equation 5.5 is repeated with the confidence threshold Q_s set to 0.99:

$$\hat{t}_s \;=\; \lceil \log_{\hat{P}_f} 0.01 \rceil \qquad \text{where} \qquad \hat{P}_f \;=\; 1 - \hat{P}_s. \tag{6.1}$$

The values of $\hat{t}_s$ corresponding to the estimates for $\hat{P}_s$ in Table 6.2a are shown in Table 6.2b.

6.5.2 *Observation: Changing Match Error Changes $\hat{P}_s$*

The failure to reliably find the globally optimal match for the 3 and 4 instance Dandelion problem and match error case 2 is the most marked difference to arise out of the change in match error. However, a quick scan of Table 6.2a shows $\hat{P}_s$ frequently differs for the

Table 6.2 Estimates $\hat{P}_s$ and $\hat{t}_s$ for the 96 experiments. a) Maximum likelihood estimates for the probability of success $\hat{P}_s$, b) Associated number of trials $\hat{t}_s$ required to solve matching problem with 99% confidence. The '*' indicates problems for which 100 trials were insufficient to reliably estimate $\hat{P}_s$. Case 1 uses $\sigma = 2$ and pairwise error. Case 2 uses $\sigma = 5$ and no pairwise error.

	Random Clutter								Multiple Instances							
	Case 1				Case 2				Case 1				Case 2			
Model	0	10	20	30	0	10	20	30	1	2	3	4	1	2	3	4
Rectangle	.25	.20	.27	.11	.56	.50	.30	.20	.28	.15	.16	.12	.59	.16	.26	.13
Pole	.43	.06	.07	.04	.40	.14	.12	.03	.35	.19	.22	.12	.29	.22	.11	.04
Deer	.21	.17	.10	.08	.94	.95	.07	.12	.64	.24	.09	.05	.90	.95	.91	.02
Tree	.22	.17	.08	.11	.48	.10	.15	.15	.31	.16	.13	.19	.94	.75	.62	.26
Leaf	.47	.15	.27	.14	.57	.22	.20	.03	.47	.04	.05	.07	.44	.07	.07	.03
Dandelion	.19	.19	.13	.17	.23	.33	.38	.51	.07	.02	.02	.08	.06	.06	*	*

(a)

	Random Clutter								Multiple Instances							
	Case 1				Case 2				Case 1				Case 2			
Model	0	10	20	30	0	10	20	30	1	2	3	4	1	2	3	4
Rectangle	17	21	15	40	6	7	13	21	15	29	27	37	6	27	16	34
Pole	9	75	64	113	10	30	35	182	11	22	19	37	14	19	40	113
Deer	20	25	44	56	2	2	64	37	5	17	49	90	2	2	2	228
Tree	19	25	56	40	8	44	29	29	13	27	34	22	2	4	5	16
Leaf	8	29	15	31	6	19	21	152	8	113	90	64	8	64	64	152
Dandelion	22	22	34	25	18	12	10	7	64	228	228	56	75	75	*	*

(b)

same image data and model. The differences are greater than can be accounted for by the confidence intervals shown in Table 5.1 on page 87, and the natural conclusion is that changing the match error changes how local search explores the search space and consequently $\hat{P}_s$.

Match error case 2 performs better than case 1 in the following sense: the estimate $\hat{P}_s$ is greater in 18 out of the 24 random clutter problems, and 15 out of the 24 multiple instance problems. Breaking this down further into the arguably easier and harder problems, for the no clutter and single instance problems, $\hat{P}_s$ for case 2 is higher for 8 out of 12 problems. For the problems with either 30 clutter lines or 4 model instances, case 2 is higher for 6 out of 12 problems. The drop in relative performance for the harder problems is interesting. However, it is risky to infer too much from such a small data set.

There is another side to the comparison between case 1 and 2. The average drop in $\hat{P}_s$ as either random clutter or multiple instances are added to a data set is nearly twice as high for case 2 as for case 1. The average differences are 0.22 and 0.41 for cases 1 and 2 respectively. Performance, as measured by $\hat{P}_s$, is deteriorating more quickly with added clutter and model instances for case 2. This is most evident in the difference between the 3 and 4 instance Leaf problem, where $\hat{P}_s$ drops from 0.91 to 0.02. This trend can be seen as foreshadowing the problems with the 3 and 4 instance Dandelion problem.

It is difficult to know for certain what is causing case 2 to perform so badly compared to case 1 on these two problems. However, a likely guess concerns the way the pairwise error term directs the early course of local search initiated from a randomly selected match. With the pairwise error based upon relative orientation included in the match error, local search tends to begin by removing pairs of matched segments which are not consistent with the initial orientation of the model. This orientation is, remember, a consequence of the initial random assignment of correspondences. By removing pairs of segments with dissimilar orientation at the beginning of the search, the algorithm is more likely to find a match whose orientation is similar to the initial random pose. Consequently, matches of differing orientation have a fairly uniform chance of being found, and this in turn seems to help the algorithm to find the globally optimal match. In contrast, it appears without the pairwise constraint search is sometimes drawn away from matches at the correct orientation.

Estimates for $\hat{P}_s$ for the 3 and 4 instance Dandelion problems for match error case 2 could be obtained given sufficient trials. Reporting failure should be understood as acknowledging that 100 trials is insufficient, and not taken to mean that $\hat{P}_s$ has necessarily fallen significantly below 0.01. Given additional trials, the globally optimal match would probably appear a sufficient number of times to reliably estimate $\hat{P}_s$. For example, an additional 100 trials were run for the 3 instance problem, and the globally optimal match did appear in 2 out of the additional 100 trials, meaning it appeared in 2 out of a total of 200 trials.

6.6 Run-time Growth

The average time required to complete a single trial of subset-convergent local search, r, has been recorded for the 48 matching problems associated with match error case 1. Estimating the time required to confidently solve each matching problem given r and $\hat{t}_s$ (shown in Table 6.2b) is a simple matter. The estimated run-time $\hat{r}_s$ is the average time for a single trial multiplied by the required number of trials:

$$\hat{r}_s = \hat{t}_s r. \tag{6.2}$$

This section will look into how $\hat{r}_s$ varies as a function of problem size n. Here, as elsewhere, problem size n is the number of possibly matching pairs of features. Bear in mind that if there are m model segments, an equal number of data segments, and no pose filtering limiting the possible pairings between segments, then $n = m^2$. Thus, adding equal numbers of model and data segments to a matching problem causes n to grow quadratically.

Regression analysis reveals the exponent for the best polynomial curve fit to $\hat{r}_s$ versus n is 1.23. The relationship appears polynomial over the range measured, and rounding up to the next highest integer suggests n^2 as an upper bound on average growth. This is equivalent to a bound of m^4 for the restricted conditions just mentioned above. Match error case 1 is considered because 100 trials was sufficient to reliably estimate the probability of success for all 48 matching problems. The average run-times r were measured on a TI Explorer II Lisp Machine. It should be noted that recent experiments suggest a new C version of the algorithm running on a Decstation 5000 is roughly 60 times faster than the Lisp version.

The estimated run-times $\hat{r}_s$ are shown in Table 6.3. Problems are identified by the model name and the number of random clutter segments or model instances added to the image data. The problem size n is shown along with the estimate run-time $\hat{r}_s$. Run-times vary between 9 seconds and 57.3 hours, and so for convenience $\hat{r}_s$ is reported both in seconds and hours. The entries are sorted by n.

Some interesting problem-specific differences are evident in Table 6.3. For example, going from the Leaf problem with 30 clutter segments to the Leaf problem with 3 instances, $\hat{r}_s$ grows from 4.3 to 34.2 hours. This growth takes place despite the fact that the problems are of comparable size, with n values of 846 and 882 respectively. In general, for similar values n, the multiple instance problems tend to take longer. However, the difference is usually less pronounced than in the case just cited. For example, $n = 261$ for both the Deer with 20 clutter segments and the Deer with 3 instances, while $\hat{r}_s$ is 0.9 hours and 1.4 hours respectively.

The two problems requiring the most time are the 2 and 3 instance Dandelion problems, each of which have $\hat{P}_s$ of only 0.02. Perhaps somewhat surprisingly, $\hat{P}_s$ goes up to 0.08 for the 4 instance Dandelion problem. For both the 2 and 3 instance problems, $\hat{t}_s = 228$, and the jump from 44.5 hours to 57.3 hours is due entirely to an increase in the average run-time per trial r.

6.6.1 Regression Analysis of Run-time Versus Problem Size n

Linear regression analysis won't answer the most important question associated with the run-time data presented in Table 6.3. The plots below show $\hat{r}_s$ growing non-linearly as a function of n. Characterizing the non-linearity is the key issue. To do this the following parameterized polynomial curve is fit to the observed data:

$$y = ax^p + b. \tag{6.3}$$

For a given exponent p, the best-fit regression parameters a and b are determined analytically. To find the best choice of p, possible values between 0.10 and 3.0 are sampled in 0.01 increments. It is p which is of greatest interest, since it suggests the degree of non-linearity

Table 6.3 Tabulated estimated run-times $\hat{r}_s$.

Random Clutter				
Problem		Size	Time $\hat{r}_s$	
Model	Cl.	n	Sec.	Hrs.
Pole	0	12	9	0.0
Rectangle	0	28	73	0.0
Pole	10	42	270	0.1
Rectangle	10	68	262	0.1
Pole	20	72	378	0.1
Deer	0	81	664	0.2
Pole	30	102	1,017	0.3
Rectangle	20	108	309	0.1
Rectangle	30	148	1,144	0.3
Tree	0	156	1,858	0.5
Deer	10	171	1,677	0.5
Deer	20	261	3,071	0.9
Dandelion	0	272	6,123	1.7
Tree	10	276	3,508	1.0
Leaf	0	306	2,397	0.7
Deer	30	351	5,835	1.6
Tree	20	396	11,066	3.1
Dandelion	10	432	9,119	2.5
Leaf	10	486	10,469	2.9
Tree	30	516	11,024	3.1
Dandelion	20	592	17,952	5.0
Leaf	20	666	7,338	2.0
Dandelion	30	752	16,735	4.6
Leaf	30	846	15,522	4.3

Multiple Instances				
Problem		Size	Time $\hat{r}_s$	
Model	In.	n	Sec.	Hrs.
Pole	1	24	24	0.0
Pole	2	42	84	0.0
Rectangle	1	52	214	0.1
Pole	3	81	162	0.0
Pole	4	96	370	0.1
Deer	1	99	214	0.1
Rectangle	2	108	835	0.2
Rectangle	3	124	724	0.2
Rectangle	4	168	1,450	0.4
Deer	2	180	1,251	0.3
Tree	1	216	2,054	0.6
Deer	3	261	5,194	1.4
Deer	4	342	11,412	3.2
Leaf	1	342	3,073	0.9
Dandelion	1	416	29,485	8.2
Tree	2	432	7,387	2.1
Tree	3	552	13,121	3.6
Leaf	2	648	70,930	19.7
Dandelion	2	736	160,147	44.5
Tree	4	780	10,650	3.0
Leaf	3	882	123,183	34.2
Dandelion	3	1136	206,317	57.3
Leaf	4	1296	89,837	25.0
Dandelion	4	1296	69,272	19.2

in the relationship between x and y. For example, If $p = 2$, this suggests a quadratic relationship.

The independent variable x is always problem size n. Analysis is presented with different choices of y. Specifically, the growth of the average run-time per trial r, the estimated number of trials $\hat{t}_s$, and the estimated run-time $\hat{r}_s$ is studied. As will be seen, comparing the growth rate of r and $\hat{t}_s$ independently to the growth rate of $\hat{r}_s$ is interesting because of the relationship between them expressed in equation 6.2. Regression is first performed on the complete set of 48 problems, making no distinction between random clutter and multiple model instance problems. Random clutter and multiple instance problems will be analyzed separately in Section 6.6.2.

Figure 6.12 shows the average run-times per trial r for all 48 matching problems. The best-fit curve is shown. Problem size varies along the x axis of the plot, and average run-times measured in minutes along the y axis. The equation for the best-fit curve is

$$r = 0.0038\, n^{1.23} - 0.44. \tag{6.4}$$

The exponent, $p = 1.23$, is the best polynomial estimate for how average run-time per trial grows as a function of problem size n.

The standard deviation for the curve with $p = 1.23$ is 1.98 minutes. The standard deviations for p equals 1 and 2 are 2.04 and 2.40 respectively. The differences in standard deviation suggests the curve with $p = 1.23$ is a much better fit than the quadratic case $p = 2$. The difference in fit between the $p = 1.23$ curve and the linear case $p = 1$ is less significant.

The required number of trials $\hat{t}_s$ is plotted versus n in Figure 6.13. Recall this is the number of trials required to find the globally optimal match with probability 0.99 or better, and that these values are tabulated in Table 6.2b. The best-fit curve is also shown and the exact parameters of the curve are

$$\hat{t}_s = 0.0133\, n^{1.22} + 24.87. \tag{6.5}$$

The standard deviation is 41.53 trials. This value is high and reflects the obvious scattering of points in this plot. The standard deviations for p equals 1 and 2 are 41.58 and 41.97 respectively. The relation between problem size n and $\hat{t}_s$ is weak. Although the overall trend is upward, there are many smaller problems which require more trials than larger problems. The best estimate is that $\hat{t}_s$ grows as $n^{1.22}$, although with such a high standard deviation, this is not a particularly accurate estimate.

Estimated run-times $\hat{r}_s$ required to confidently solve each problem are plotted versus n in Figure 6.14. Equation 6.2 is used to compute $\hat{r}_s$. The y axis gives run-time in hours. The best-fit curve is shown, and the exact parameters are

$$\hat{r}_s = 0.00052\, n^{1.53} - 1.08. \tag{6.6}$$

The standard deviation is 8.11 hours. The standard deviations for p equals 1 and 2 are 8.33 and 8.22 respectively.

Since multiplying r and $\hat{t}_s$ yields $\hat{r}_s$, if r and $\hat{t}_s$ were uncorrelated, then multiplying the approximating functions in equations 6.4 and 6.5 ought to yield a good estimate of growth in $\hat{r}_s$ as a function of n. The resulting exponent would be 2.50, the sum of 1.23 and 1.22.

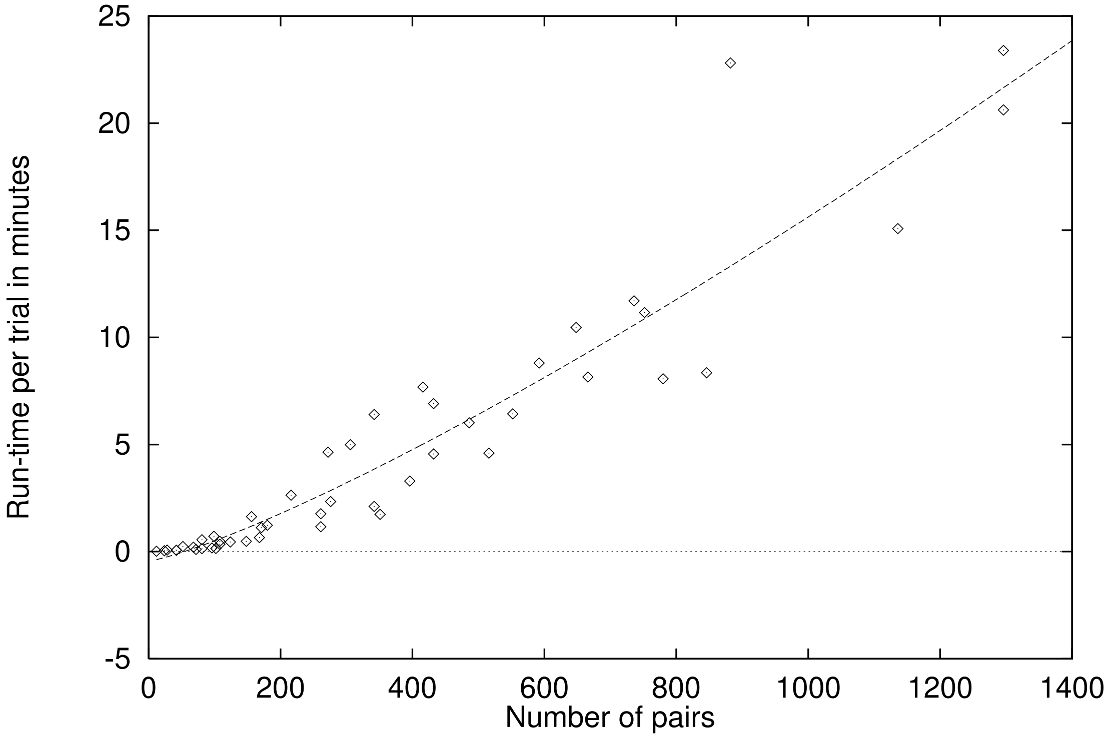


Figure 6.12 Average time to run r for 1 trial plotted versus n. The best-fit curve with exponent $p = 1.23$ is shown.

Table 6.4 Co-occurrence of $\hat{r}_s$ and $\hat{t}_s$ above/below average. Count of problems falling above and below predicted values of r and $\hat{t}_s$ show that most frequently both fall below the predicted value.

Average time per trial r	Required Trials $\hat{t}_s$: Above	Below
Above	3	13
Below	14	18

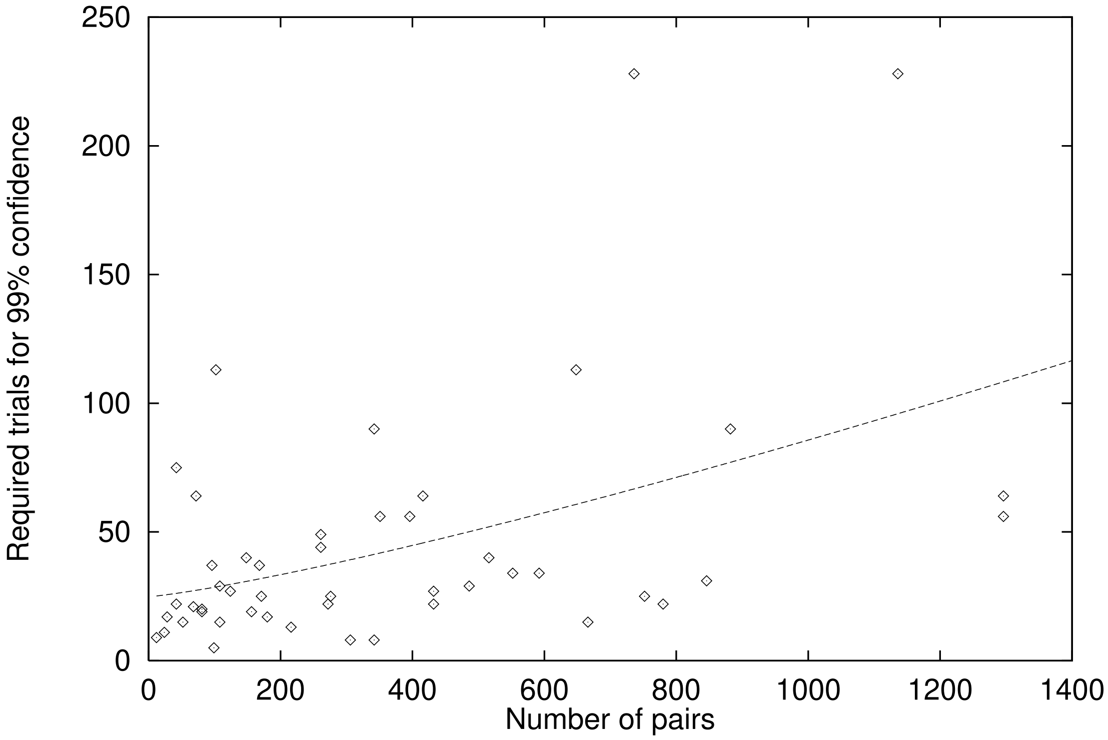


Figure 6.13 Estimated required trials $\hat{t}_s$ plotted versus n. The best-fit curve with exponent $p = 1.22$ is shown. However, the standard deviation for quite high.

This is considerably higher than the 1.53 measured for the actual run-time estimates plotted in Figure 6.14.

This apparent discrepancy is explained by the fact that r and $\hat{t}_s$ are correlated. Table 6.4 shows how the 48 problems break down in terms of falling either above or below the values predicted by equations 6.4 and 6.5. For instance, the upper left corner indicates 3 of the 48 problems had both r and $\hat{t}_s$ falling above the predicted values. In contrast, for 18 out of the 48 problems, both values fell below the predicted value. Finally, for 27 out of 48 problems, when one value fell above the average, the other fell below. The net effect is to produce a measured rate of growth, equation 6.6, lower than predicted by multiplying equations 6.4 and 6.5.

Equation 6.6 is the key result of this section. It represents the best guess of how run-time $\hat{r}_s$ grows as a function of n, subject to the assumed polynomial form of equation 6.3. In particular, it says the exponent $p = 1.53$ is the best guess given the observed data. Rounding up to the next whole integer, and conjecturing that run-time growth is bounded by n^2, seems the most natural extrapolation given the observed data.

6.6.2 Random Clutter and Multiple Instance Problems

By design, the above analysis makes no distinction between the random clutter and multiple instance matching problems, and performance over both sets of problems has been averaged. Analyzing $\hat{r}_s$ separately for each supports the contention that matching problems

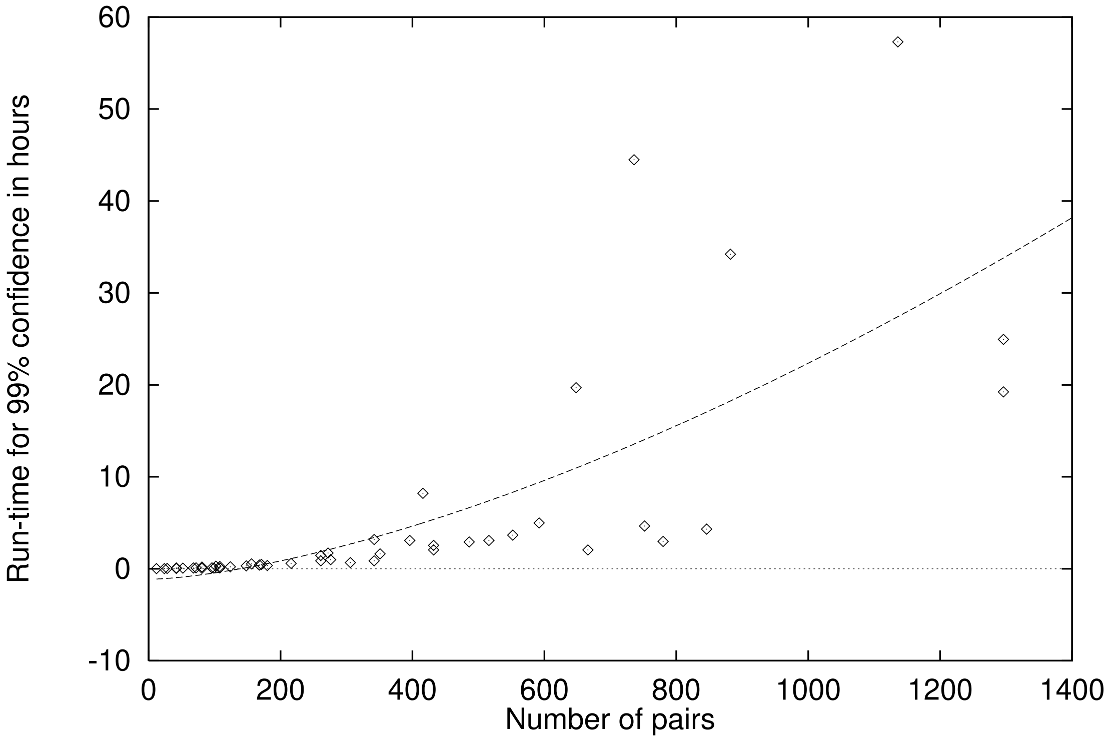


Figure 6.14 Estimated run-times $\hat{r}_s$ plotted versus n. The best-fit curve with exponent $p = 1.53$ is shown.

involving multiple model instances are more difficult to solve. Figure 6.15 shows $\sqrt{\hat{r}_s}$ for the 24 random clutter and 24 multiple model instance matching problems. Unlike before, in these two plots the y axis is the square root of estimated run-time in hours. Square root of run-time is plotted to make the error bars described below easier to see. It also lessons the dominance of points with high n.

The polynomial from equation 6.3 is fit to the data shown in Figure 6.15. The resulting best fit curves are shown. For the random clutter matching problems

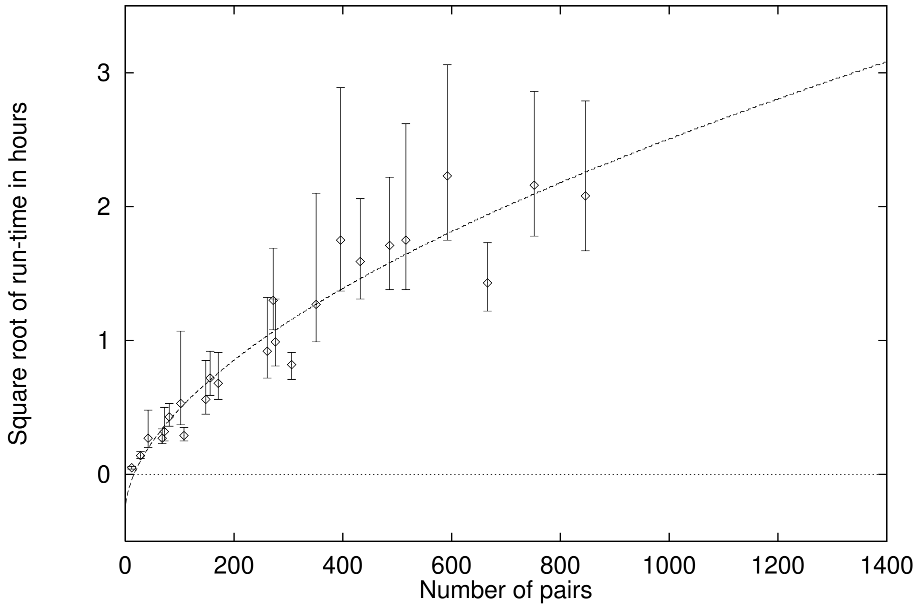
$$\sqrt{\hat{r}_s} \;=\; 0.058\, n^{0.56} - 0.277. \tag{6.7}$$

For the multiple instance matching problems

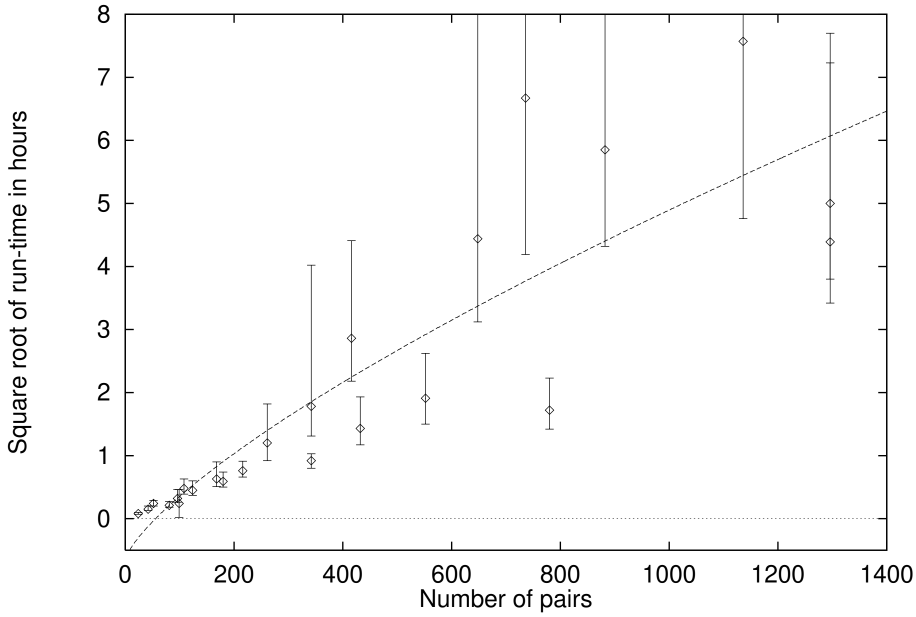
$$\sqrt{\hat{r}_s} \;=\; 0.033\, n^{0.74} - 0.656. \tag{6.8}$$

For the random clutter problem, the standard deviations are 0.20, 0.27 and 0.24 for p equals 0.56, 0.10 and 1.00 respectively. For the multiple model instance problems, the standard deviations are 1.12, 1.30 and 1.15 for p equals 0.74, 0.10 and 1.00 respectively.

Squaring the exponents in equations 6.7 and 6.8 suggests the run-time for the random clutter matching problems grows as roughly $n^{1.2}$, while for multiple instance problems run-time grows as $n^{1.5}$. These measurements show one concrete sense in which the multiple instance problems are more difficult. However, both of these growth rates fall below an n^2 upper bound.



(a) Random Clutter



(b) Multiple Instances

Figure 6.15 Run-time plots for random clutter and multiple instance problems. The square root of $\hat{r}_s$ versus n is plotted to accentuate error bars for smaller n problems.

The error bars suggest how much of the individual variation between problems may be ascribed to the sampling process used to estimate $\hat{P}_s$ and thus $\hat{t}_s$. Recall that Table 5.2 on page 88 provides confidence intervals for $\hat{P}_s$. These are propagted through equations 6.1 and 6.2 and plotted in Figure 6.15.

In Figure 6.15a, the regression curve passes through most of the confidence intervals associated with measured values of $\sqrt{\hat{r}_s}$. It is plausible that the variance in measured performance is due solely to the process of estimating $\hat{t}_s$ from 100 trials of local search. However, this is not the case for Figure 6.15b, where some problems clearly lie above the curve, and others below it. The geometric form of a particular matching problem clearly seems to influence performance.

These results strongly suggest the principal factor governing problem difficulty is the number of candidate pairs of features n. However, factors such as the geometric form of the model also come into play. The internal symmetries of the Dandelion make it a more challenging model. The geometric form of the clutter also matters. Matching a model when the clutter segments are randomly distributed is easier than when additional segments mimic the geometric form of the model, and in the extreme tested here, represent distinct 2nd, 3rd, and 4th instances of the same model.

6.7 Speculation: Why Don't Run-times Grow Faster?

It may have surprised some readers that the growth in average run-time for both random clutter and multiple instance matching problems should appear bounded by n^2. The search spaces contain 2^n matches, and for the larger problems, as few as 12 out of $1,296$ potentially matching pairs of segments participate in the optimal match. That 100 trials of subset-convergent local search typically discovers these small sets of optimally matching pairs of segments is a startlingly good result.

It is difficult to explain precisely why subset-convergent local search does this well. Two outwardly plausible explanations are considered. The first is that expanded problem size means expanded options for local search, and this in turn eases the relative difficulty of larger problems. Recall the Hamming-distance-1 neighborhood defined in Section 5.1. For this neighborhood, if $n = 10$, then local search may select from among 10 options, while if $n = 1,000$, local search may select from among $1,000$ options. More options implies greater mobility, and may explain why paths from randomly selected initial matches to globally optimal matches are found with reasonable probability for both large and small problems. This explanation is explored in Section 6.7.1

The second explanation is that random sampling, by chance, generates initial matches which are 'close' to the optimal match. Because this explanation is intuitively attractive, it will be considered further. It is hypothesized that local search succeeds only when some fraction of the paired segments in the globally optimal match are included in the initial randomly selected match. Performance predicted by this hypothesis is compared to that actually observed, with the result that predicted performance falls orders of magnitude short of that observed on larger problems. Readers already content that this is not a plausible explanation may wish to pass over Section 6.7.2.

6.7.1 Likely Explanation: Larger n Means More Options

The relatively strong performance of the subset-convergent local search algorithm might be explained by considering how increasing the size of the search space changes the options available to local search. For instance, increasing the number of data segments in an image increases the options, or potential paths through the search space. It is entirely possible that a clutter segment may act as a 'bridge', permitting local search to move away from an undesirable match. In this way, adding clutter can create pathways through the search space that would not otherwise exist.

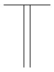
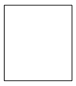
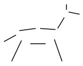
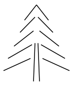
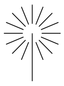
This may explain some otherwise surprising results. Consider the probability of success estimates $\hat{P}_s$ in Table 6.2a (page 123) for the problem of matching the Dandelion with random clutter. For case 1, $\hat{P}_s$ remains roughly constant as clutter is added. With no clutter, $\hat{P}_s = 0.19$, and with 30 additional clutter segments, $\hat{P}_s = 0.17$. For case 2, $\hat{P}_s$ increases monotonically from 0.23 to 0.51. The fraction of the initial randomly selected correspondences leading to the optimal match is remaining roughly constant in case 1 and actually increasing in case 2, and this despite the fact that the underlying search space is growing exponentially. Evidently, the growth in the number of matches leading to the globally optimal match is keeping pace with the exponential growth of the search space.

Adding model segments may also add pathways from initial to optimal matches, and this suggests $\hat{P}_s$ may not necessarily decrease as models get larger. Table 6.5 permits this hypothesis to be tested visually. There is one row for each model, sorted in ascending order of model size m. Models get larger going from top to bottom. Both model size m and number of candidate pairs n are shown. Columns are ordered by descending probability of success $\hat{P}_s$. Problems to right have lower probability of success on a single trial. If larger models generate matching problems with lower probability of success, then the models shown in Tables 6.5a and 6.5b ought to line up along the left to right descending diagonal. With the striking exception of the Leaf model, this is somewhat the case for the problems without random clutter. However, almost the reverse can be seen for the 30 clutter segments problems.

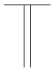
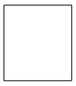
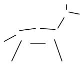
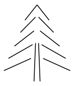
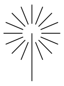
There is a semi-intuitive explanation for why $\hat{P}_s$ is so high for the Leaf model without clutter. The Leaf model is a piecewise linear approximation to the curved outline of an elongated region. If local search is able to get the model pointed in roughly the correct direction relative to the data, then it often 'walks into' the optimal match by adding and removing pairs of segments along the boundary. In contrast to the Leaf problem, the Dandelion problem (already discussed in Section 6.4) is made difficult by the many partial symmetries of the model.

Table 6.5b shows that adding 30 clutter segments alters the relative difficulty of the matching problems associated with the different models. Most striking is the move of the Dandelion from being having the lowest $\hat{P}_s$ estimate without clutter to having the highest with clutter. The absolute value for $\hat{P}_s$ is essentially the same in both cases, even though the size of the search space expands from 2^{272} to 2^{752}. In general, the pattern in Table 6.5b is almost the opposite of what might be expected, with problems tending to get easier as they get larger. The only real explanation for this observation is that adding segments opens up new paths to the globally optimal match at a rate sufficient to keep $\hat{P}_s$ reasonably high, even while the search space is growing exponentially.

Table 6.5 Matching problems ranked by size and $\hat{P}_s$. Models are ranked from smaller to larger going down, and higher to lower $\hat{P}_s$ from left to right.

Matching Problems with No Clutter Segments							
		$\hat{P}_s$					
m	n	0.47	0.43	0.25	0.22	0.21	0.19
3	12						
4	28						
9	81						
12	156						
16	272						
18	306						

(a)

Matching Problems with 30 Clutter Segments							
m	n	0.17	0.14	0.11	0.11	0.08	0.04
3	102						
4	148						
9	351						
12	516						
16	752						
18	846						

(b)

Table 6.6 Percentage of pairs included in optimal matches over n. The number of good pairs g is divided by the total number of pairs n. The average number of pairs l included in initial matches are also shown.

		Random Clutter					Multiple Instances				
Model	l	g	0	10	20	30	g	1	2	3	4
Pole	12	4	33.3	9.5	5.6	3.9	8	33.3	19.0	9.9	8.3
Rectangle	16	6	21.4	8.8	5.6	4.1	12	23.1	11.1	9.7	7.1
Deer	36	9	11.1	5.3	3.4	2.6	9	9.1	5.0	3.4	2.6
Tree	48	13	8.3	4.7	3.3	2.5	15	6.9	3.5	2.7	1.9
Dandelion	64	17	6.2	3.9	2.9	2.3	21 [a]	5.0	2.3	1.8	1.6
Leaf	72	20	6.5	4.1	3.0	2.4	12 [b]	3.5	2.2	1.4	0.9

[a] For the 2 instance problem the 2nd instance with $g = 17$ is optimal
[b] For the 2 instance problem the 2nd instance with $g = 14$ is optimal

6.7.2 *Unlikely Explanation: Random Sampling Finds Good Starts*

The Ransac algorithm [35] and some median filtering algorithms [70] use random sampling to generate a subset of data samples which are correct. In the context of matching, such samples are model-data pairs which match and are therefore elements of the globally optimal correspondence c^*. First this section shows the complete impracticality of this exact strategy for the matching problems presented in this chapter. It is then asked whether subset-convergent local search is finding c^* because random sampling is generating initial correspondences c_i which, although not proper subsets of c^*, do contain some fraction of the pairs in c^*.

The percentage of the possible pairs of features belonging to c^* is a key determinant of problem difficulty for approaches such as Ransac and median filtering. These algorithms uniformly select k pairs of features until they are confident that one or more of the samples contain only 'good' pairs. Here, a good pair is one belonging to c^*. The probability of obtaining k good pairs is $(g/n)^k$, where g out of n pairs are good. The approach is reasonable so long as $g/n > 0.5$ and k is small. However, if g/n drops significantly below 0.5 or k gets larger, the approach fails badly.

Table 6.6 shows the ratios g/n as percentages for the 48 matching problems associated with match error case 1. The l values, which indicate the average number of pairs included in an initial random correspondence, are set to 4 times the number of model segments m. This choice approximates the biased random selection used in the experiments presented earlier. These l values will play an important role in the analysis which follows.

Random sampling to generate a small set of good pairs is infeasible for most of the matching problems. For instance, only 0.09% of the possible pairs are good for the 4 instance Dandelion problem. Using the approach employed by Ransac or median filtering to find 3 correctly matching pairs with probability better than 0.99 would require over 6 million independent samples (in marked contrast, subset-convergent local search need only draw 56 samples in order to confidently find the globally optimal match).

In contrast to these other techniques, local search matching is is able to remove pairs from a correspondence mapping c_i, and therefore c_i need not be a proper subset of c^*. In other words, local search may find c^* starting with a correspondence c_i containing pairs of segments not in c^*. What is of interest is whether it appears that a fraction of pairs must be common to both c_i and c^* in order for subset-convergent local search to find c^* starting from c_i.

Imagine a fairly simple relationship, one where local search initiated from an initial correspondence c_i leads to c^* so long as every 'bad' pair in c_i is offset by a 'good' pair. A good pair is one which is an element of c^*, a bad pair is not. The intuition is rather simplistic; it is that local search can make up for an incorrect pairing so long as there is a correct pairing to counteract it. This may be called the one-good-offsets-one-bad rule, and it is illustrated in Figure 6.16.

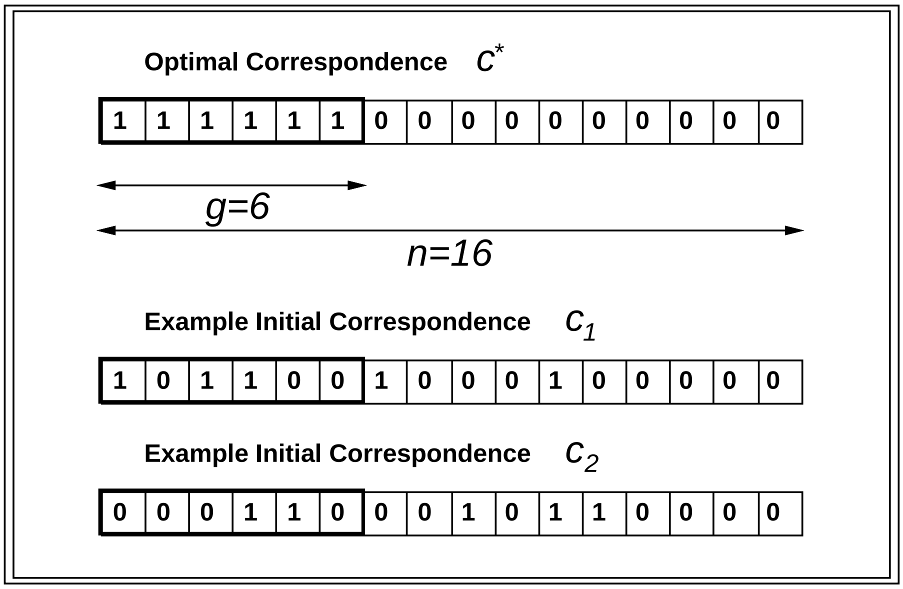


Figure 6.16 Optimal correspondence c^* with g 'good' bits on the left. By the one-good-offsets-one-bad rule, local search from c_1 would lead to c^* while local search from c_2 would not.

The one-good-offsets-one-bad rule is illustrated in Figure 6.16. An optimal correspondence c^* is written as a bitstring with all the 1 bits associated with the matching pairs on the left. There are $g = 6$ such bits in this example, and the ratio $g/n = 0.38$. Two possible initial correspondences, c_1 and c_2, are shown. By the assumption that the one-good-offsets-one-bad rule is valid, local search from c_1 will lead to c^* while local search from c_2 will not.

The one-good-offsets-one-bad is intuitively simple, but it does not explain the observed performance of the subset-convergent local search algorithm. For most of the problems in Table 6.6, the number of bits l initially set is over twice the total number of good bits, and the one-good-offsets-one-bad suggests complete failure. Picking some smaller fraction, such as 1/4 (one-good-offsets-two-bad) instead of 1/2, doesn't help much. Consider the multiple instance Leaf problems. The globally optimal correspondence c^* contains only 12

Table 6.7 Measured $\hat{P}_s$ over predicted P_s using 1/10 correct rule.

	Random Clutter				Multiple Instances			
Model	0	10	20	30	1	2	3	4
Pole	0.4	0.3	1.0	1.1	0.7	1.0	3.9	3.0
Rectangle	0.3	0.9	2.6	1.9	0.8	1.4	2.0	2.6
Deer	0.3	1.8	4.3	9.8	1.5	3.0	3.9	5.6
Tree	0.8	4.3	9.1	40.3	3.1	26.7	64.6	463.4
Dandelion	3.5	43.6	199.0	1,171.0	12.9	120.1	2,009.8	19,250.3
Leaf	12.9	74.1	1,172.5	3,400.3	24.8	143.1	1,647.1	40,567.6

pairs of segments, while $l = 72$. Yet as shown in Table 6.2b, $\hat{P}_s$ never drops below 0.04 for this set of problems. Even picking a smaller fraction, such as 1/10, doesn't lead to predicted probabilities of success P_s remotely resembling those actually measured for the larger matching problems.

This failing may be demonstrated by comparing the probability of success P_s implied by this rule to the maximum likelihood estimates $\hat{P}_s$ measured for each matching problem. To define P_s, recall the bitstring representation for correspondence mappings illustrated in Figure 6.16. The leftmost g bits indicate pairs in c^*. Presuming bitstrings c_i are selected uniformly from the set of all strings with l bits set to 1, then P_s is the ratio of the number of strings leading to c^* over the n choose l possible bitstrings:

$$P_s = \frac{\sum_{i=h}^{l} \binom{g}{i} \binom{n-g}{l-i}}{\binom{n}{l}}. \tag{6.9}$$

The numerator sums all the possible ways that between h and l on bits (1's) can appear in the g leftmost good bits. The g choose i term accounts for the ways of distributing i 1's among the leftmost g bits. The $n - g$ choose $l - i$ term accounts for all the ways of distributing $l - i$ remaining 1's in the remaining $n - g$ bits.

Table 6.7 shows the ratio of $\hat{P}_s$ from Table 6.2a over P_s predicted by equation 6.9 for the 48 matching problems associated with match error case 1. A large value indicates the equation is grossly underestimating $\hat{P}_s$. The predicted probabilities of success are computed using

$$h = \lceil l/10 \rceil. \tag{6.10}$$

The rule implied by this equation says that at least 1/10th of the l bits must be among the g correct bits in order for local search to find c^*. Outwardly, it doesn't seem like much to ask that 1 out of 10 be correct. However, the predicted probabilities of success begin to drop precipitously compared to the actual measured values for the larger matching problems. For example, P_s computed using equation 6.9 is over one thousands times smaller than the actual estimate $\hat{P}_s$ for the Dandelion matching problem with 30 random clutter segments. The predictions get even an order of magnitude worse for the 4 instance Leaf and Dandelion problems. In short, local search is actually performing up to ten thousand times better than predicted by this simplistic model.

The analysis just presented should dispel any notions that subset-convergent local search is performing as observed because the randomly selected initial correspondences are close to, or similar to, the optimal correspondence. Given the precipitous drop in P_s predicted by such an assumption, and the failure to detect such a drop in actual performance, it must be concluded that other factors are at work and causing subset-convergent local search to be effective. As described in Section 6.7.1, a likely explanation lies in the way increasing problem size increases search options, and thereby permits local search to move from initial correspondences c_i to optimal correspondences c^*, even when these correspondences may share few if any pairs of model-data line segments.

6.8 Conclusion

When the experiments reported in this chapter were initially designed, it was by no means obvious that subset-convergent local search would perform as well as observed. This chapter has demonstrated that for all the matching problems set forth in Section 6.2, subset-convergent local search finds the globally optimal match with probability 0.99 or better. Further, doing so requires fewer than 100 independent trials for all but the hardest of these problems; the multiple instance Dandelion problems.

It is significant that the experiments reported in this chapter found no evidence for an exponential relationship between run-time and problem size. Certainly it is possible such evidence would have been found given larger problems. However, in defense of these experiments, remember search spaces ranging in size from 2^{12} to $2^{1,296}$ were tested, and that no distinctions were made between random clutter matching problems, multiple instance matching problems, and matching problems involving symmetric models. As Section 6.6 demonstrated, n^2 appears to bound average run-times observed for all these problems.

The n^2 bound bound on average case growth in run-time conjectured here, if true, would be a lower bound than for any other known general geometric matching algorithm. Tree search shares this bound for random clutter matching problems, but is exponential for multiple instance matching problems [48]. It would be of great theoretical interest to try to validate this bound analytically. However, for all the reasons discussed at the beginning of this chapter, this could prove very difficult.

Theoretical concerns aside, n^2 growth is still m^4 growth in model size when all model segments might match any data segment, and it represents a serious practical limitation for subset-convergent local search, just as it does for all other known general geometric matching algorithms. To test the power of subset-convergent local search algorithm by itself, this chapter made no use of ancillary information. However, this is not the best way to use the subset-convergent local search algorithm. Any source of information which can reduce n by reducing the size of the set of possibly matching pairs will improve performance significantly. An initial pose estimate for an object is an obvious additional source of constraint, and the usefulness of initial pose estimates has already been demonstrated in matching problems presented in Chapters 1 and 5

Up to now, this thesis has considered only essentially 2D or weak-perspective matching problems. This is in keeping with the precedent set by the vast majority of the previous work on geometric matching. However, the next two chapters will extend the local search matching in order to find globally optimal correspondence mappings and associated 3D-to-2D pose estimates for geometric matching problems involving full 3D perspective. To do this, Chapter 7 will summarize the work of Kumar [68, 69] on computing 3D-to-2D pose, and show how his algorithm can be modified to efficiently support full-perspective matching. Using the pose algorithm to fit 3D models to 2D image data, Chapter 8 will extend the local search algorithm presented in Chapter 5 to perform full-perspective matching.

CHAPTER 7

Fitting Under Full-Perspective

7.1 Introduction

Computing the 3D pose of a 3D object model relative to corresponding features in an image is similar in concept to the problem of computing 2D pose already described in Chapter 4. Here 3D pose is defined to be the position and orientation of the model relative to the image plane, and hence the camera, which minimize a sum of squared errors. The error measure for 3D pose is a point-to-plane distance measure described below. The problem of finding the best-fit 3D pose is a non-linear optimization problem, and while the weak-perspective-2D best-fit pose is determined analytically, the 3D full-perspective pose must be determined using an iterative technique.

The basic measure and associated algorithm used to compute 3D pose is the same as that developed and studied by Kumar [70]. Kumar's nonlinear least-squares optimization algorithm used to find the best pose is adapted from Horn [52], and in particular uses quaternions to represent rotation in order to speed convergence. Here, three enhancements to Kumar's algorithm are developed. First, a midpoint-to-midpoint regularizing term is added to Kumar's original measure. In a manner analogous to that already described for weak-perspective fitting in Section 4.5, this added measure will yield a unique best-fit pose in many cases where such a unique pose would not otherwise be defined.

The second enhancement is related to the first, and that is the use of the Levenberg-Marquardt method within the inner loop of the iterative pose algorithm. The Levenberg-Marquardt method [93] is a good way of solving systems of equation which may be nearly singular. In particular, it only incurs an additional computational cost when the equations are near singular. It has been used by David Lowe [82] to solve pose equations similar to those developed here. The introduction of the regularization term makes otherwise singular cases near singular, and hence it is necessary to use the Levenberg-Marquardt method to solve these cases.

The third enhancement is to completely rederive the 3D pose update equations in order to characterize pose in terms of a sum of state vectors: one vector per pair of corresponding line segments. The motivation is to support incremental evaluation of the best-fit pose and associated fit error for the Hamming-Distance-1 neighbors of a match. Recall that this was described for weak-perspective fitting in Section 5.2.1. Removing the summation over the complete set of corresponding segments within the inner loop of the iterative pose algorithm will make full-perspective matching more efficient.

Kumar [69] has recently developed newer fit measures for determining 3D pose which are much closer to the ISPD measure used for weak-perspective fitting. Unfortunately, for the following reasons, they are not used in this thesis. The first reason is that Kumar's original fit measure and associated pose algorithm can be recast in the more efficient state vector form just mentioned. Unfortunately, this is not feasible for the pose algorithm using the newer measures. The second reason is timing; the newer measures did not mature soon enough to be incorporated into this work. Some of the algorithms introduced below make sparing use of the 3D pose algorithm, and in the future these should probably use Kumar's newer measure.

The following section presents the original point-to-plane squared error measure and iterative pose algorithm developed by Kumar [68]. Section 7.3 introduces point-to-ray regularization of Kumar's original algorithm which is analogous to the point-to-point regularization for weak-perspective fitting introduced in Section 4.5. Section 7.4 derives the linear update equation used in the inner loop of the pose algorithm using state vectors associated with individual pairs of 3D world line segments and 2D image line segments.

7.2 Kumar's Algorithm

A line in an image and the focal point of the camera define a plane in 3-space. If 3D models were perfect, and no errors were introduced in during image feature extraction, then model lines in 3-space projecting onto this line in the image would lie in this plane. This simple observation is the basis for the fit measure used in Kumar's original 3D pose algorithm. As illustrated in Figure 7.1, the origin of the camera coordinate system is the camera focal point, and the focal point plus the two endpoints of an image line segment define a plane in 3-space. If the image line segment is the projection of a line in the 3D world, then this 3D world line segment *must* lie on this plane. Due to noise, the segment usually will not lie exactly in the plane, but the distance from the 3D world segment to the plane should be small. Kumar's pose algorithm solves for rigid transformation which minimizes the sum-of-squared distances between points on 3D world lines and these planes.

The point-to-plane distance between endpoints of the 3D world line segment and the plane formed by the image line segment may be written as the dot product of the unit normal to the plane and the 3D endpoints transformed into camera coordinates. The sum-of-squared perpendicular distances may thus be written as

$$E \;=\; \sum_{i=1}^{n}\sum_{j=1}^{2}\left(\hat{N}_i \cdot \left(R\vec{P}_{ij} + \vec{T}\right)\right)^2 . \tag{7.1}$$

The sum is over n pairs of corresponding 3D and 2D line segments. The unit normal to the plane defined by each 2D segment is $\hat{N}_i$. The endpoints of the corresponding 3D segments are $\vec{P}_{i1}$ and $\vec{P}_{i2}$. The pose of the 3D segments relative to the 2D segments, and hence the camera, is expressed as a rotation R and translation $\vec{T}$ applied to the 3D points. The perpendicular point-to-plane distances for endpoint $\vec{P}_{i1}$ is shown in Figure 7.1. Observe that although endpoints of 3D line segments are used here, in general any set of 3D points assumed to project to the image line could be used.

The best-fit 3D pose for a set of corresponding 3D and 2D line segments is defined by the rotation R^* and translation $\vec{T}^*$ which minimize equation 7.1. Solving for R^* and $\vec{T}^*$ is a non-linear optimization problem. The approach proposed by Kumar is to linearize

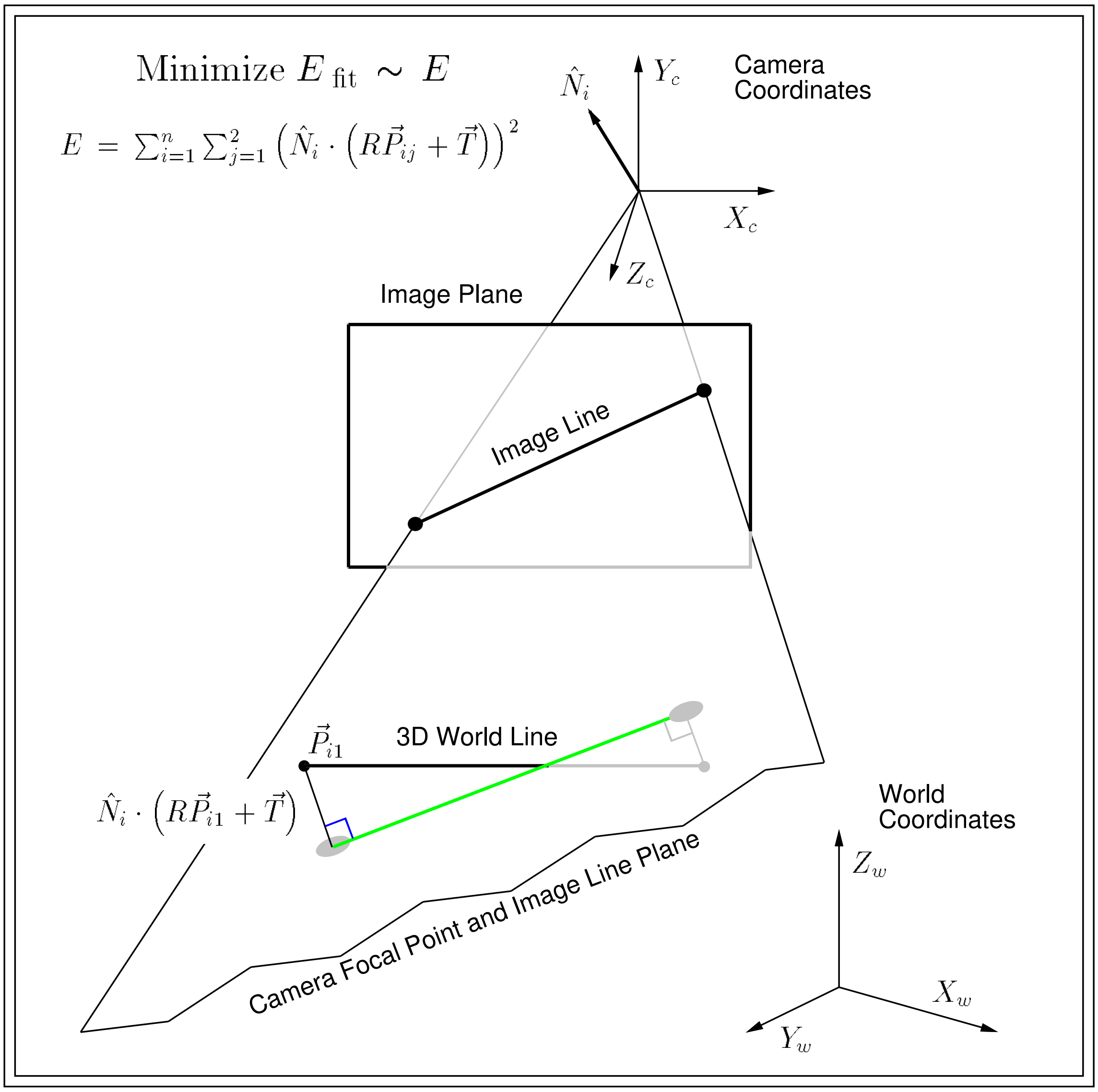


Figure 7.1 Point-to-plane fit measure for 3D-to-2D pose algorithm.

the measure about some estimated pose R^e and $\vec{T}^e$. In particular, Rodriguez's formula approximating a rotation with a cross product is used to derive a linear version equation 7.1:

$$E \;=\; \sum_{i=1}^{n}\sum_{j=1}^{2}\lambda_i\left(\hat{N}_i\cdot\left(R^e\vec{P}_{ij}+\delta\vec{w}\times\left(R^e\vec{P}_{ij}\right)+\vec{T}+\Delta\vec{T}\right)\right)^2. \tag{7.2}$$

The term inside the quadratic is now linear in $\delta\vec{w}$ and $\Delta\vec{T}$.

A weighing term λ_i has also been added to the sum. There are several ways this weight may be used to alter the relative importance of particular sets of corresponding segments. One is to reflect the relative length of the corresponding 2D data line segments in order to give longer segments greater influence over R^* and $\vec{T}^*$. Another use is to offset the tendency for 3D line segments far from the camera to exert a disproportionate influence relative to endpoints near the camera. This latter bias is explained below. In practice, λ_i is typically selected so as to both reflect relative length of the image line segments and to offset the distant 3D points bias.

The heart of the distance bias lies in the fact that the same amount of error or noise applied to a point in the image plane gives rise to a larger perpendicular point-to-plane distance for 3D points lying farther from the image plane. Figure 7.2 illustrates how perpendicular error is accentuated for points farther from the image plane. the same camera coordinates, image plane, and image line segment in Figure 7.2 is the same as in Figure 7.1. Figure 7.2 shows the point-to-plane distance two points: $\vec{P}_{\text{near}}$ and $\vec{P}_{\text{far}}$. Both points project to the same image point, and hence the displacement off the true line in the image plane is the same for both. However, the 3D point-to-plane distance is larger in absolute terms for $\vec{P}_{\text{far}}$ than for $\vec{P}_{\text{near}}$. This means that the same amount of pixel noise applied to the projection of a point far from the image plane will exert a greater influence on the optimal pose estimate R^* and $\vec{T}^*$ than for a point nearer the image plane. An approximate correction for this problem is to set λ_i to be the reciprocal of the distance from the $3D$ endpoint and the camera focal point given the initial pose estimate R^e and $\vec{T}^e$.

The basis for the 3D pose algorithm is to set the partial derivative of equation 7.2 with respect to $\delta\vec{w}$ and $\Delta\vec{T}$ equal to zero. Doing this yields the following set of equalities:

$$\sum_{i=1}^{n}\sum_{j=1}^{2}\lambda_i\left(\hat{N}_i\cdot\Delta\vec{T}+\left(\left(R^e\vec{P}_{ij}\right)\times\hat{N}_i\right)\cdot\delta\vec{w}\right)\hat{N}_i \;=\; -\;\sum_{i=1}^{n}\sum_{j=1}^{2}\lambda_i\left(\hat{N}_i\cdot\left(R^e\vec{P}_{ij}+T^e\right)\right)\hat{N}_i \tag{7.3}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{2}\lambda_i\left(\hat{N}_i\cdot\Delta\vec{T}+\left(\left(R^e\vec{P}_{ij}\right)\times\hat{N}_i\right)\cdot\delta\vec{w}\right)\left(\left(R^e\vec{P}_{ij}\right)\times\hat{N}_i\right) \;=\; -\;\sum_{i=1}^{n}\sum_{j=1}^{2}\lambda_i\left(\hat{N}_i\cdot\left(R^e\vec{P}_{ij}+T^e\right)\right)\left(\left(R^e\vec{P}_{ij}\right)\times\hat{N}_i\right) \tag{7.4}$$

Equations 7.3 and 7.4 together constitute six linear equations in six unknowns: the individual elements of $\delta\vec{w}$ and $\Delta\vec{T}$. The $\delta\vec{w}$ and $\Delta\vec{T}$ values solving this system of equations minimize the linearized error in equation 7.2.

The iterative step of the pose algorithm repeatedly solves for $\delta\vec{w}$ and $\Delta\vec{T}$ and uses these

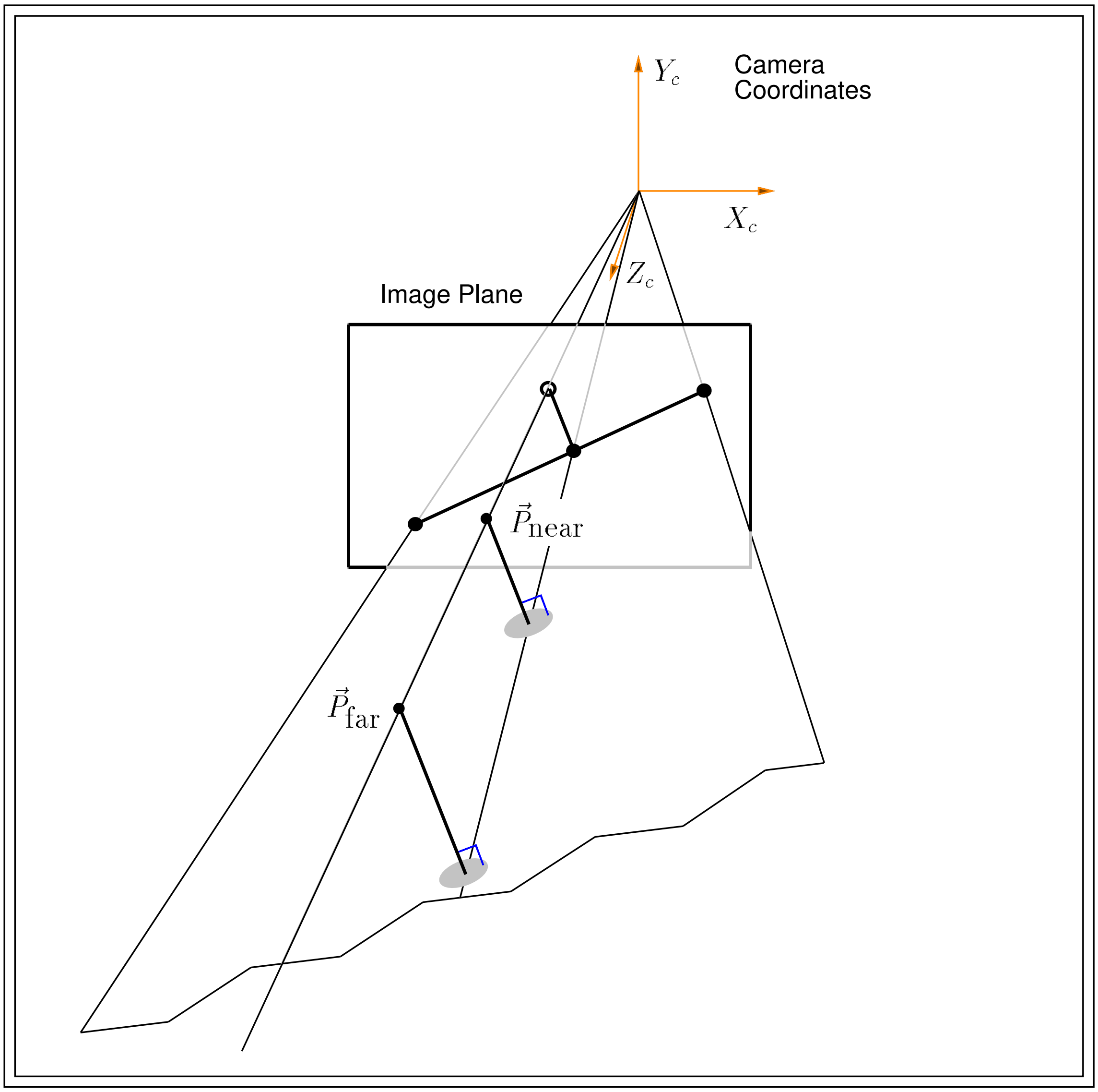


Figure 7.2 Point-to-plane error accentuated for $\vec{P}_{\text{far}}$ versus $\vec{P}_{\text{near}}$.

values to update R^e and $\vec{T}^e$. Iteration terminates with the best-fit pose R^* and $\vec{T}^*$ when both $\delta\vec{w}$ and $\Delta\vec{T}$ are essentially zero or when the relative change in fit drops below a threshold.

The linear equation may be written more compactly as follows:

$$\begin{pmatrix} C & D \\ D^{\mathrm{T}} & F \end{pmatrix} \begin{pmatrix} \Delta\vec{T} \\ \delta\vec{w} \end{pmatrix} = \begin{pmatrix} \vec{G} \\ \vec{H} \end{pmatrix}, \tag{7.5}$$

where

$$C = \sum_{i=1}^{n}\sum_{j=1}^{2} \lambda_i \hat{N}_i \hat{N}_i^{\mathrm{T}} \tag{7.6}$$

$$D = \sum_{i=1}^{n}\sum_{j=1}^{2} \lambda_i \hat{N}_i \left(\left(R^e\vec{P}_{ij}\right) \times \hat{N}_i\right)^{\mathrm{T}} \tag{7.7}$$

$$F = \sum_{i=1}^{n}\sum_{j=1}^{2} \lambda_i \left(\left(R^e\vec{P}_{ij}\right) \times \hat{N}_i\right)\left(\left(R^e\vec{P}_{ij}\right) \times \hat{N}_i\right)^{\mathrm{T}} \tag{7.8}$$

$$\vec{G} = -\sum_{i=1}^{n}\sum_{j=1}^{2} \lambda_i \left(\hat{N}_i \cdot \left(R^e\vec{P}_{ij} + T^e\right)\right)\hat{N}_i \tag{7.9}$$

$$\vec{H} = -\sum_{i=1}^{n}\sum_{j=1}^{2} \lambda_i \left(\hat{N}_i \cdot \left(R^e\vec{P}_{ij} + T^e\right)\right)\left(\left(R^e\vec{P}_{ij}\right) \times \hat{N}_i\right). \tag{7.10}$$

Equation 7.5 is essentially equivalent to equations 19 and 20 in Kumar [70]. From an efficiency standpoint, the problem with using the terms defined in equations 7.7 through 7.10 is the presence of the current best estimates R^e and $\vec{T}^e$ within the sums. These terms must be recomputed on each iteration of the pose algorithm, and doing this means iterating over the n pairs of corresponding segments: iteration over improved pose estimates is compounded by iteration over matching pairs of segments. Section 7.4 will show how this costly internal summing over pairs can be avoided. But first, it is helpful to add a regularizing point-to-ray term to the point-to-plane fitting error just defined.

7.3 Point-to-Ray Regularization

For a number of common geometric configurations, such as three 3D segments meeting at a vertex, there is no unique pose R^* and $\vec{T}^*$ which minimize equation 7.1. From the standpoint of local search, this is a problem. Recall from Chapter 5 that the match error is undefined when a unique best-fit pose cannot be determined. Correspondence mappings without unique best-fit poses are off-limits to the local search process, and experience suggests this can cut off potential paths to better matches.

To produce a unique best-fit pose for many of these geometric configurations, an additional squared error term is added to the existing point-to-plane measure. This additional term measures 3D point-to-ray distance for corresponding 3D model line segments and 2D image line segments. The ray is defined by the focal point of the camera and the midpoint of each of the image line segments. The regularization term is the 3D distance from this ray to the midpoint of the corresponding 3D model line segment.

This point-to-ray distance is easily computed in the framework already developed. Note that the squared distance from a 3D point to a ray may be computed as the sum of two point-to-plane distances. All that is required is that both planes contain the ray and that they be orthogonal. The following equation shows equation 7.2 extended to include the point-to-ray distance measure.

$$\begin{aligned} E \; = \; & \sum_{i=1}^{n}\sum_{j=1}^{2}\lambda_i\left(\hat{N}_i\cdot\left(R^e\vec{P}_{ij}+\delta\vec{w}\times\left(R^e\vec{P}_{ij}\right)+\vec{T}+\Delta\vec{T}\right)\right)^2 \\ + \; & \sum_{i=1}^{n}\lambda_i\tau\left(\hat{N}_i\cdot\left(R^e\vec{P}_{im}+\delta\vec{w}\times\left(R^e\vec{P}_{ij}\right)+\vec{T}+\Delta\vec{T}\right)\right)^2 \\ + \; & \sum_{i=1}^{n}\lambda_i\tau\left(\hat{O}_i\cdot\left(R^e\vec{P}_{im}+\delta\vec{w}\times\left(R^e\vec{P}_{ij}\right)+\vec{T}+\Delta\vec{T}\right)\right)^2 \end{aligned} \tag{7.11}$$

The additional terms measure point-to-plane distance from the midpoint $\vec{P}_{im}$ of the 3D model segment to the plane defined by the unit normal $\hat{N}_i$ and two a second plane passing through the midpoint of the 2D image segment and orthogonal to the first. The unit normal of this second plane is $\hat{O}_i$. The relative importance of the regularizing term is controlled by the additional weight τ.

Equation 7.11 is rewritten more compactly by combining the summation over endpoints and midpoints:

$$E \; = \sum_{i=1}^{n}\sum_{j=1}^{4}\lambda_{ij}\left(\hat{N}_{ij}\cdot\left(R^e\vec{P}_{ij}+\delta\vec{w}\times\left(R^e\vec{P}_{ij}\right)+\vec{T}+\Delta\vec{T}\right)\right)^2, \tag{7.12}$$

and accordingly equating new terms indexed by i and j with the terms appearing in equation 7.11:

$$\begin{array}{llll} \lambda_{i1} = \lambda_i & \lambda_{i2} = \lambda_i & \lambda_{i3} = \tau\lambda_i & \lambda_{i4} = \tau\lambda_i \\ \hat{N}_{i1} = \hat{N}_i & \hat{N}_{i2} = \hat{N}_i & \hat{N}_{i3} = \hat{N}_i & \hat{N}_{i4} = \hat{O}_i \\ \vec{P}_{i1} = \vec{P}_{i1} & \vec{P}_{i2} = \vec{P}_{i2} & \vec{P}_{i3} = \vec{P}_{im} & \vec{P}_{i4} = \vec{P}_{im}. \end{array}$$

Choosing the proper weight τ is a matter of striking a balance between two conflicting constraints. If τ is too large, then the point-to-ray distance begins to significantly influence the best-fit pose. Since, as has been said many times, data segments fragment, this introduces a negative bias into the pose estimate. However, for otherwise underconstrained cases, as τ is made smaller, the linear equation 7.5 becomes more nearly singular. In practice, $\tau = 10^{-2}$ or $\tau = 10^{-3}$ appears to strike a reasonable balance between these two competing considerations.

To solve linear equation 7.5 for $\Delta\vec{T}$ and $\delta\vec{w}$ when using regularization, it has proven important to use the Levenberg-Marquardt method. This is a clever way of smoothly combining first-order and second-order iterative techniques within one algorithm. The Levenberg-Marquardt method will automatically use a quick to converge second-order technique when possible. However, if problems arise due to near singularities in the error measure, it will switch to a slower and more reliable first-order technique. A more complete description of the Levenberg-Marquardt method can be found in [93].

7.4 The State Vector Approach

Using equation 7.5 to repeatedly solve for $\Delta\vec{T}$ and $\delta\vec{w}$ requires recomputing the sums in equations 7.7 through 7.10 for each iteration. This is necessary because R^e and T^e are updated each time through the loop based upon $\delta\vec{w}$ and $\Delta\vec{T}$. Replacements for equations 7.7 through 7.10 are derived in this section for which there is no need to sum over the n corresponding features within the inner loop of the pose algorithm.

By multiplying out the square and rearranging terms, it possible to rewrite equation 7.2 as follows:

$$\begin{aligned} E &= \vec{R}_v A \vec{R}_v^{\mathrm{T}} + \delta\vec{w}^{\mathrm{T}} R_m A R_m^{\mathrm{T}} \delta\vec{w} + 2\vec{R}_v A R_m^{\mathrm{T}} \delta\vec{w} + 2\vec{R}_v B \left(\vec{T} + \Delta\vec{T}\right) + \\ & \quad 2\delta\vec{w}^{\mathrm{T}} R_m B \left(\vec{T} + \Delta\vec{T}\right) + \left(\vec{T}^{\mathrm{T}} + \Delta\vec{T}^{\mathrm{T}}\right) C \left(\vec{T} + \Delta\vec{T}\right). \end{aligned} \tag{7.13}$$

Written in this way, teh summation over the pairs of corresponding segments has been hidden within the terms A, B and C. These terms will be described in detail below. The 9 element vector $\vec{R}_v$ and the 3x9 matrix R_m are alternative arrangements of the row vectors in the original rotation estimate matrix R^e:

$$R^e = \begin{vmatrix} \vec{U}^{\mathrm{T}} \\ \vec{V}^{\mathrm{T}} \\ \vec{W}^{\mathrm{T}} \end{vmatrix} \qquad \vec{R}_v = \begin{vmatrix} \vec{U} \\ \vec{V} \\ \vec{W} \end{vmatrix} \qquad R_m = \begin{vmatrix} 0 & -\vec{W}^{\mathrm{T}} & \vec{V}^{\mathrm{T}} \\ \vec{W}^{\mathrm{T}} & 0 & -\vec{U}^{\mathrm{T}} \\ -\vec{V}^{\mathrm{T}} & \vec{U}^{\mathrm{T}} & 0 \end{vmatrix}. \tag{7.14}$$

Defining $\vec{R}_v$ and R_m simplifies equation 7.13.

Taking the same basic approach as in Section 7.2, the partial derivative of equation 7.13 is set to zero and new definitions for the terms in equations 7.7 through 7.10 are derived:

$$D = R_m B \tag{7.15}$$

$$F = R_m A R_m^{\mathrm{T}} \tag{7.16}$$

$$\vec{G} = -C\vec{T} - B^{\mathrm{T}} \vec{R}v \tag{7.17}$$

$$\vec{H} = -D\vec{T} - R_m A \vec{R}v. \tag{7.18}$$

The C matrix remains as originally defined in equation 7.6.

The A and B matrices are defined as follows:

$$A = \sum_{i=1}^{n} \sum_{j=1}^{4} \lambda_{ij} \left(\hat{N}_{ij} \ominus \vec{P}_{ij}\right) \left(\hat{N}_{ij} \ominus \vec{P}_{ij}\right)^{\mathrm{T}} \tag{7.19}$$

$$B = \sum_{i=1}^{n} \sum_{j=1}^{4} \lambda_{ij} \left(\hat{N}_{ij} \ominus \vec{P}_{ij}\right) \hat{N}_{ij}^{\mathrm{T}} \tag{7.20}$$

where the Kronecker product of $\hat{N}_{ij}$ and $\vec{P}_{ij}$ is

$$\hat{N}_{ij} \ominus \vec{P}_{ij} = \begin{vmatrix} n_{ij\,\mathrm{x}}\; p_{ij\,\mathrm{x}} \\ n_{ij\,\mathrm{x}}\; p_{ij\,\mathrm{y}} \\ n_{ij\,\mathrm{x}}\; p_{ij\,\mathrm{z}} \\ n_{ij\,\mathrm{y}}\; p_{ij\,\mathrm{x}} \\ n_{ij\,\mathrm{y}}\; p_{ij\,\mathrm{y}} \\ n_{ij\,\mathrm{y}}\; p_{ij\,\mathrm{z}} \\ n_{ij\,\mathrm{z}}\; p_{ij\,\mathrm{x}} \\ n_{ij\,\mathrm{z}}\; p_{ij\,\mathrm{y}} \\ n_{ij\,\mathrm{z}}\; p_{ij\,\mathrm{z}} \end{vmatrix} \quad (7.21)$$

$$\hat{N}_{ij} = \begin{vmatrix} n_{ij\,\mathrm{x}} \\ n_{ij\,\mathrm{y}} \\ n_{ij\,\mathrm{z}} \end{vmatrix} \quad (7.22)$$

$$\vec{P}_{ij} = \begin{vmatrix} p_{ij\,\mathrm{x}} \\ p_{ij\,\mathrm{y}} \\ p_{ij\,\mathrm{z}} \end{vmatrix} \quad (7.23)$$

It is only a small step from the definitions of A, B and C to a state vector $\vec{S}$, where this vector completely captures the information required to determine 3D pose. The vector $\vec{S}$ is nothing more than the union of the unique elements of A, B and C. For the sake of completeness, the 60 elements of $\vec{S}_i$ are enumerated in Table 7.1. Table 7.1 is divided into three sections, with headings indicating which elements of $\vec{S}_i$ are associated with A, B and C. To obtain the state vector $\vec{S}_c$ for a correspondence mapping c, the individual contributions from the n corresponding pairs of segments are summed:

$$\vec{S}_c = \sum_{i=1}^{n} \vec{S}_i. \quad (7.24)$$

In the context of matching, there are three distinct ways in which the state vector approach saves computation compared to using the original equations 7.6 through 7.10. The first saving is in having only to compute the state vectors $\vec{S}_i$ once when the initial set of potentially matching pairs of 3D model segments and image segments is defined. During matching, the state vectors for different correspondence mappings can be computed by summing as indicated in equation 7.24.

A greater savings is realized within the inner loop of the pose algorithm. Recall n is the number of paired features, and that the complexity of the pose algorithm using the original set of equations 7.6 through 7.10 is the number of iterations multiplied by n. This can make computing pose costly if either the number of iterations or n gets large. Removing summation over n makes the complexity of the pose algorithm independent of n. To put it another way, the time required to compute 3D pose no longer depends upon the size of the match. For larger matches this can save a great deal of time.

The final and most pronounced savings is realized when evaluating a Hamming-distance-1 neighborhood as part of local search. Using the state vector form, it is possible to incrementally compute the 3D pose for the Hamming-Distance-1 neighbors of a current correspondence c. Let $\vec{S}_c$ be the state vector for c and let neighbor c' be obtained by removing the kth pair from c:

$$\vec{S}_{c'} = \vec{S}_c - \vec{S}_k. \quad (7.25)$$

To add the kth pair, add rather than subtract $\vec{S}_k$.

Table 7.1 Fully defined 3D pose determining state vector.

C Matrix	A Matrix
$\vec{S}_i\,(1) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{x}}$	$\vec{S}_i\,(25) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{x}}$
$\vec{S}_i\,(2) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{y}}$	$\vec{S}_i\,(26) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{y}}$
$\vec{S}_i\,(3) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{z}}$	$\vec{S}_i\,(27) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{z}}$
$\vec{S}_i\,(4) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{y}}\ n_{ij\,\mathrm{y}}$	$\vec{S}_i\,(28) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{y}}$
$\vec{S}_i\,(5) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{y}}\ n_{ij\,\mathrm{z}}$	$\vec{S}_i\,(29) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{z}}$
$\vec{S}_i\,(6) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{z}}\ n_{ij\,\mathrm{z}}$	$\vec{S}_i\,(30) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{z}}$
B Matrix	$\vec{S}_i\,(31) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{x}}$
$\vec{S}_i\,(7) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{x}}$	$\vec{S}_i\,(32) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{y}}$
$\vec{S}_i\,(8) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{y}}$	$\vec{S}_i\,(33) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{z}}$
$\vec{S}_i\,(9) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{z}}$	$\vec{S}_i\,(34) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{y}}$
$\vec{S}_i\,(10) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{x}}$	$\vec{S}_i\,(35) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{z}}$
$\vec{S}_i\,(11) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{y}}$	$\vec{S}_i\,(36) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{z}}$
$\vec{S}_i\,(12) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{z}}$	$\vec{S}_i\,(37) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{x}}$
$\vec{S}_i\,(13) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{x}}$	$\vec{S}_i\,(38) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{y}}$
$\vec{S}_i\,(14) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{y}}$	$\vec{S}_i\,(39) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{z}}$
$\vec{S}_i\,(15) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{z}}$	$\vec{S}_i\,(40) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{y}}$
$\vec{S}_i\,(16) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{y}}\ n_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{x}}$	$\vec{S}_i\,(41) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{z}}$
$\vec{S}_i\,(17) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{y}}\ n_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{y}}$	$\vec{S}_i\,(42) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{x}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{z}}$
$\vec{S}_i\,(18) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{y}}\ n_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{z}}$	$\vec{S}_i\,(43) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{y}}\ n_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{x}}$
$\vec{S}_i\,(19) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{y}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{x}}$	$\vec{S}_i\,(44) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{y}}\ n_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{y}}$
$\vec{S}_i\,(20) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{y}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{y}}$	$\vec{S}_i\,(45) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{y}}\ n_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{z}}$
$\vec{S}_i\,(21) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{y}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{z}}$	$\vec{S}_i\,(46) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{y}}\ n_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{y}}$
$\vec{S}_i\,(22) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{z}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{x}}$	$\vec{S}_i\,(47) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{y}}\ n_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{z}}$
$\vec{S}_i\,(23) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{z}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{y}}$	$\vec{S}_i\,(48) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{y}}\ n_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{z}}$
$\vec{S}_i\,(24) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{z}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{z}}$	$\vec{S}_i\,(49) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{y}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{x}}$
	$\vec{S}_i\,(50) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{y}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{y}}$
	$\vec{S}_i\,(51) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{y}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{z}}$
	$\vec{S}_i\,(52) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{y}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{y}}$
	$\vec{S}_i\,(53) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{y}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{z}}$
	$\vec{S}_i\,(54) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{y}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{z}}$
	$\vec{S}_i\,(55) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{z}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{x}}$
	$\vec{S}_i\,(56) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{z}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{y}}$
	$\vec{S}_i\,(57) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{z}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{x}}\ p_{ij\,\mathrm{z}}$
	$\vec{S}_i\,(58) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{z}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{y}}$
	$\vec{S}_i\,(59) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{z}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{y}}\ p_{ij\,\mathrm{z}}$
	$\vec{S}_i\,(60) = \sum_{j=1}^{4} \lambda_{ij}\, n_{ij\,\mathrm{z}}\ n_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{z}}\ p_{ij\,\mathrm{z}}$

7.5 Conclusion

This chapter presented the 3D pose algorithm developed by Kumar [68, 69], along with a series of modifications to make the algorithm more useful for local search matching. To extend the algorithm to work on problems involving small numbers of model and image features, Section 7.3 introduced a regularization term. To make pose determination more efficient, particularly in the context of local search, Section 7.4 rederived the basic pose update equations in terms of state vectors associated with pairs of model and image line segments.

Chapter 8 will show how, using the 3D pose algorithm, local search matching can find optimal correspondence mappings and 3D pose estimates between 3D object models and image features. Where before, E_{fit} was defined in terms of the residual integrated squared perpendicular distance between model and image line segments in the image plane, a new local search algorithm will define E_{fit} in terms of the residual 3D point-to-plane distances measured between corresponding 3D world line segments and image lines. Where before, the best-fit pose was a 2D similarity transformation between model and image line segments, with the new algorithm it will be the best-fit 3D rotation R^* and translation $\vec{T}^*$ relating the 3D object to the camera.

For full-perspective matching, the initial pose estimate R^e and $\vec{T}^e$ takes on added importance. In weak-perspective matching, it was optionally used to constrain the set of possibly matching pairs of model and image features. With full-perspective matching, it is a required input to the 3D pose algorithm. For an added computational burden, the pose algorithm could be run without an initial estimate by coarsely sampling the space of possible poses [69]. However, this would both be expensive and beg the more fundamental question of determining which 3D features are visible. For both these reasons, the work on full-perspective matching will assume the initial pose of the 3D object is roughly known.

CHAPTER 8

Matching with 3D Perspective

8.1 Introduction

The matching problems considered in Chapters 5 and 6 were solved using weak-perspective matching as defined in Section 1.3. The previous chapter presented Kumar's [70, 71] full-perspective 3D pose algorithm, and this chapter will use this 3D pose algorithm to develop new full-perspective matching algorithms. These algorithms will solve optimal correspondence and pose problems without imposing the restrictive weak-perspective imaging assumption.

Problems requiring full-perspective matching arise in the context of landmark-based robot navigation. In landmark-based navigation, a robot has a partial model of its environment, and through recognizing these landmarks, the robot maintains an up-to-date estimate of where it is in the environment. Such knowledge is a prerequisite of intelligent navigation for both people and robots: it is hard to go where you want if you don't know where you are. Landmark-based navigation is further described in the following section, and the importance of accounting for full-perspective when navigating indoors will become apparent.

The terms full-perspective and weak-perspective are used to describe imaging, matching algorithms and matching problems. A review of how these terms are used will be helpful. Recall that Section 1.3 defines weak-perspective and full-perspective as types of imaging. Full-perspective imaging is defined as the mapping between 3D object features and 2D image features implied by perspective projection and associated with a pin-hole camera model. Weak-perspective imaging, or more precisely weak-perspective-2D imaging, approximates full-perspective by projecting 3D features into the 2D image plane based upon a single fixed pose estimate, and subsequently involves rotating, translating and scaling these projected features in the image plane to vary an object's appearance.

A *weak-perspective matching algorithm* is defined as one which uses only weak-perspective imaging to account for the changes in appearance of a model relative to corresponding image data. The Hamming-distance-1, steepest-descent algorithm introduced in Section 5.1, and the subset-convergent algorithm introduced in Section 5.4, are both weak-perspective matching algorithms. In contrast, a *full-perspective matching algorithm* is defined as one which uses full-perspective imaging to account for changes in the appearance of a model. To do this, the full-perspective matching algorithms introduced in this chapter will use the full-perspective fitting algorithm (3D pose algorithm) presented in Chapter 7.

A *weak-perspective matching problem* is defined as a problem which can be solved using weak-perspective matching. To be more precise, it is a problem for which weak-perspective imaging is adequate to register corresponding model and data features. Whether a matching problem is a weak-perspective matching problem depends in part upon the pose estimate used to project the 3D model into the 2D image plane; the projection must be transform to fit the image data using only rotation, translation and scaling in the image plane. If this is not already obvious, examples in the following section will help. A *full-perspective matching problem* is defined as one which can be solved using full-perspective matching, and as a matter of practice, this term is reserved for problems *requiring* full-perspective matching.

This chapter develops three different full-perspective matching algorithms, and tests each on the same set of full-perspective matching problems. The first is called the *full-perspective-inertial-descent* algorithm, and it is the most obvious and brute-force extension of the weak-perspective steepest-descent algorithm introduced in Section 5.1. Starting with the weak-perspective algorithm, this new algorithm is obtained by completely replacing the weak-perspective fitting algorithm defined in Chapter 4 with the full-perspective fitting algorithm defined in Chapter 7. In addition, steepest-descent is replaced by a strategy called here inertial-descent. Inertial descent caches potentially good moves detected when evaluating the n neighbors of a match, and tries these before again testing the entire neighborhood. This is described further in Section 8.3.1.

The inertial-descent-full-perspective algorithm uses the iterative 3D pose algorithm *every* time it tests a match, i.e. for every change in model-data correspondence. This means it runs the pose algorithm n times when testing the n Hamming-distance-1 neighbors of a match. This is fairly expensive, and the motivation behind the second full-perspective matching algorithm is to save time during neighborhood evaluation by using the weak-perspective pose update and associated match error to approximate a full-perspective pose update and match error computation. The hope is that often the cheaper weak-perspective test will suggest the same neighbor, and that the 3D pose recovery algorithm need only be run after the most promising neighbor has been selected. This algorithm is called the *hybrid-weak-full-perspective* algorithm because it utilizes both weak-perspective and full-perspective during local search.

The third full-perspective matching algorithm developed in this chapter is called the *hybrid-subset-convergent* algorithm. The motivation for this algorithm is found in the relative strength of the subset-convergent local search algorithm from Section 5.4 compared to the Hamming-distance-1 steepest-descent algorithm from Section 5.1. Recall that the subset-convergent algorithm is, in some sense, built on top of the Hamming-distance-1 steepest-descent algorithm. More precisely, the subset-convergent algorithm uses the Hamming-distance-1 steepest-descent algorithm to find paths from *subsets* of a Hamming-distance-1 locally optimal match to new, often better, matches. In similar fashion, the hybrid-subset-convergent algorithm defined in this chapter uses the hybrid-weak-full-perspective algorithm to find paths from *subsets* of a hybrid-weak-full-perspective locally optimal match to new, often better, matches.

The following section introduces geometric matching problems associated with landmark-based robot navigation. The importance of 3D perspective in these problems is emphasized. Section 8.3 presents the three full-perspective matching algorithms: inertial-descent-full-perspective local search, hybrid-weak-full-perspective local search, and hybrid-subset-convergent local search. Section 8.3.4 summarizes the salient attributes of these algorithms and of the weak-perspective-steepest-descent algorithm presented in Section 5.1. Section8.4

compares the performance of these three algorithms, plus that of the weak-perspective-steepest-descent algorithm, on the full-perspective matching problems introduced in Section 8.2.

8.2 Landmark-Based Robot Navigation

One of the many domains in which geometric matching problems arise is that of landmark-based robot navigation. For example, a robot moving through a hallway tracks its progress using vision, and it acquires images such as shown in Figures 8.1a and 8.1b. It must test and update its position and orientation estimate based upon the appearance of known landmarks. To infer its 3D pose relative to the known landmarks, it must solve for the correspondence between modeled landmark features and image features.

As illustrated in Figure 8.2, modest changes in position can introduce pronounced perspective effects. Figure 8.2a shows 9 different poses for a robot. A partial wire frame model of the hallway is shown in relation to these poses. To give a sense of scale to this illustration, pose 5 is 40 feet from the doorway at the end of the hallway. Image 1 in Figure 8.1a is taken from pose 5. Figure 8.2b shows the prominent landmark features as they would appear from each of the 9 indicated poses. These features are 3D segments from a partial wire frame model of the hallway.

The recognition problems suggested by Figure 8.2 are 'perspective-sensitive' because small changes in robot position introduce significant perspective effects. For example, it simply isn't possible using weak-perspective to accurately fit the projection of the landmark as it appears from pose 2 to segments extracted from image 1. An easy way to see this is to consider fitting the projection from pose 2 to the projection from pose 5. There is no way using only 2D rotation, translation and scaling to introduce the changes in relative orientation between segments evident in these two projections. It is also worth noting that these perspective effects arise with relatively minor changes in robot position. Pose 2 is only 2 feet to the left of pose 5.

The partial hallway model, the 9 pose estimates, and the two images shown above will form the basis for the experiments presented below. The goal of the experiments in Section 8.4.1 is to recover the true pose that produced the image of Figure 8.1a; each of the 9 pose estimates shown in Figure 8.2 will be used as initial estimates of robot position. The experiment presented in Section 8.4.2 treats a harder problem; that of recovering from a larger error in initial pose estimate. The robot poses from which the two images in Figure 8.1 are taken differ by over 10 feet. The problem in the second experiment is to recover the pose using the data from one image and the pose from the other as an initial estimate.

8.3 Three Full-perspective Matching Algorithms

Three distinct full-perspective local search matching algorithms are defined in this section. The first is a brute-force extension from weak-perspective to full-perspective in which the weak-perspective fitting from Chapter 4 is completely replaced by the full-perspective fitting, or 3D pose computation, from Chapter 7. The second algorithm, in the interests of saving computation, uses the less expensive weak-perspective fitting to evaluate local

(a)

(b)

Figure 8.1 Two hallway images. a) Image 1, b) Image 2.

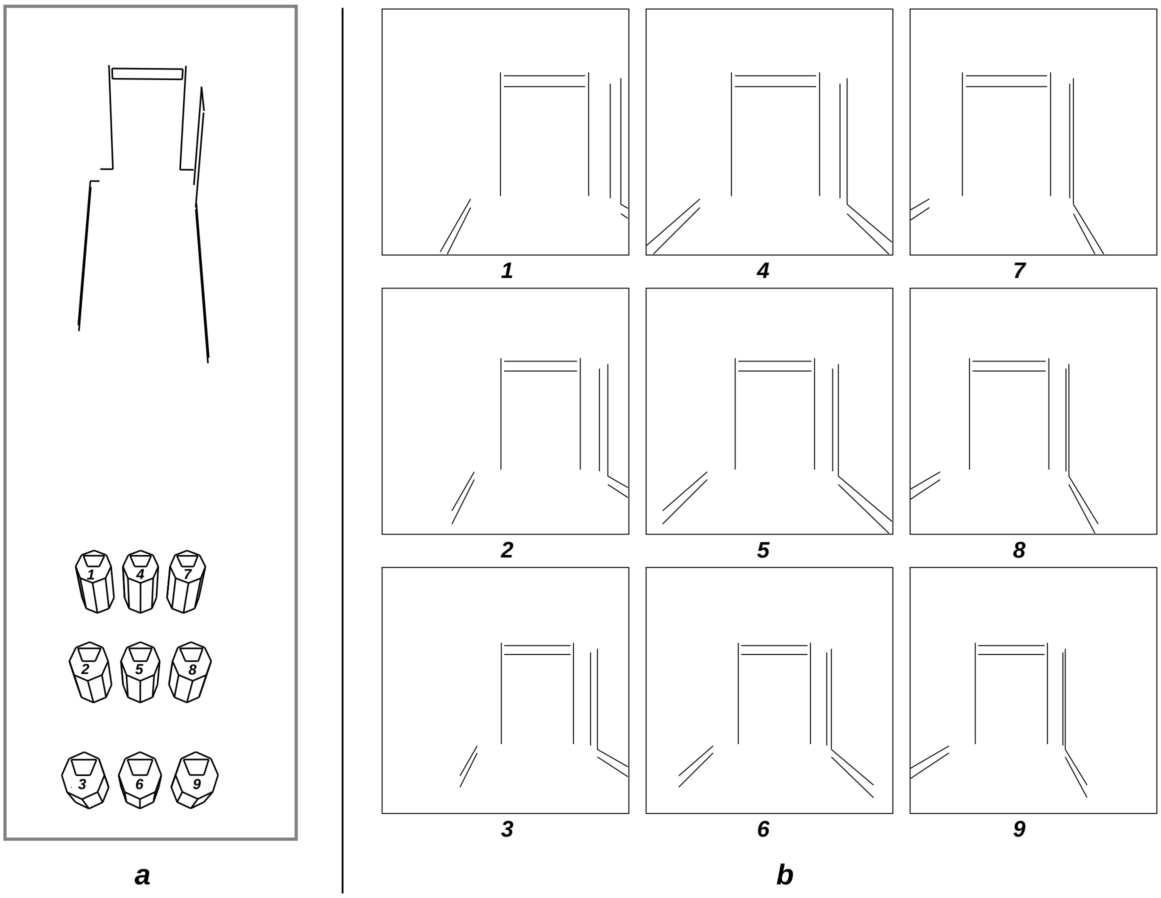


Figure 8.2 Perspective views of landmarks: a) Robot in relation to a partial model of the hallway. The image in Figure 8.1a was taken from pose 5. Each of the 9 poses shown are used as initial estimates from which landmark recognition tries to recover the robot's true pose. The eight poses around pose 5 are obtained by moving the robot forward and backward 4 feet and side-to-side 2 feet. b) Landmark features as they would appear from each of the 9 poses. The relative orientation of the baseboards and the door frame change as the robot's pose estimate shifts laterally.

neighborhoods and recomputes 3D pose only when adopting a new improved match. Finally, the third algorithm extends the second using the same subset-convergent search strategy presented in Chapter 5.

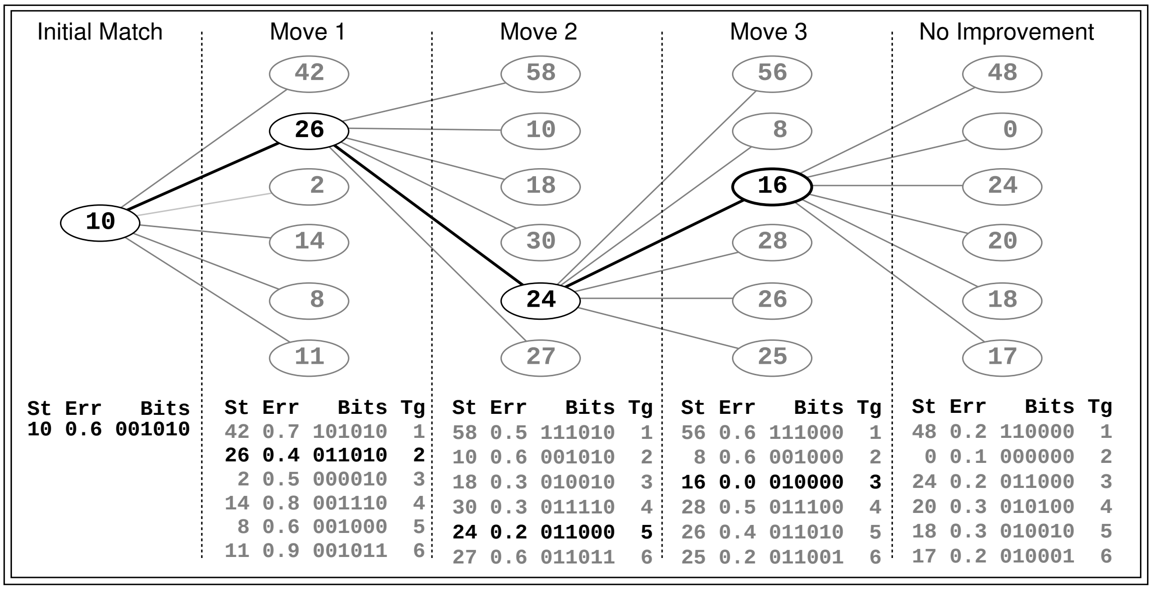
For all three algorithms, the omission error E_{om} is computed for 2D model segments *after* they have been projected into the image plane based upon the 3D pose estimates. Additionally, a clipping algorithm is used during projection to remove segments, or portions of segments, not projecting into the bounded image plane. Consequently, omission is computed only for those portions of the 3D model actually visible from the best-fit 3D pose. The rationale is that it is unreasonable to penalize a match for failing to find data outside the image, and this approach seems adequate for the experiments presented here. However, this approach appears to be a source of problems on recent and very preliminary tests for matching models to the outdoor scene shown in Figure 5.12. How best to handle omission for features projecting outside the image plane needs further study.

8.3.1 Full-perspective-Inertial-Descent Matching

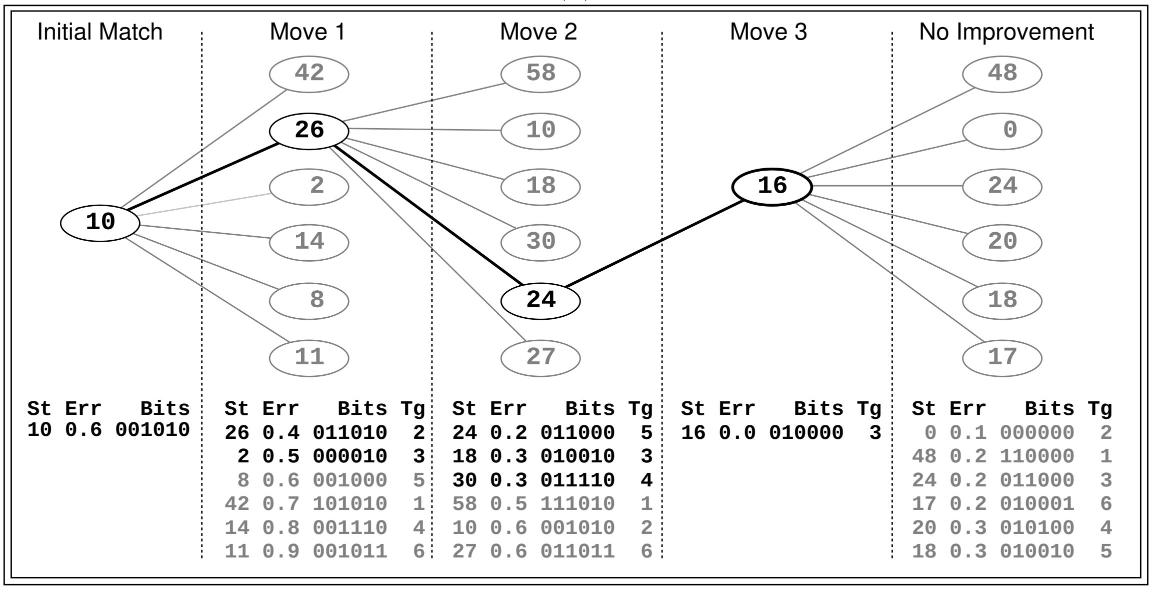
The full-perspective-inertial-descent algorithm uses a modified version of the match error in which the fit error E_{fit} is defined to be a function of the residual point-to-plane squared error terms used by the 3D pose algorithm described in Chapter 7. This means the 3D pose of the object model relative to the camera is computed for *every* match which is tested. In particular, the iterative nonlinear optimal 3D pose is run for each of the n neighbors in the Hamming-Distance-1 neighborhood.

Evaluating the n neighboring matches using the 3D pose algorithm is computationally demanding. Using the steepest-descent strategy, n 3D poses must be computed before every step in the local search process. In an effort to save computation, a modified strategy named here *inertial-descent* is used in place of steepest-descent. The idea is to exploit information obtained when testing the n neighborhoods in order to choose a sequence of moves through the search space. When the n neighbors of a current match are evaluated, those neighborhood transformations which lead to better matches are ranked in order of improvement and stored in a list. Provided this list is not empty, the algorithm applies the first transformation and thus moves to a new, and guaranteed to be improved, match state. At this point, rather than completely evaluate the n neighbors of this new match, the inertial-descent algorithm tests whether the second ranked transformation applied to this new match generates an even better match. If it does, it repeats this process with the third ranked transformation, and so on. This is done until either the list of transformations is exhausted or a transformation is encountered which no longer yields improvement. Once this occurs, then all n neighbors are tested and a new list built. Like steepest-descent, the algorithm terminates when none of the n neighbors are better than the current match.

Figure 8.3 illustrates the difference between steepest-descent and inertial-descent on a simple example. The search space contains 64 states with error values assigned at random. Figure 8.3a illustrates a steepest-descent search path. The initial state is 10, and the 6 neighbors of 10 are generated by toggling each bit in the bitstring representation of 10. The scores for all 6 neighbors are shown below move 1, with the best neighbor shown in black. For each neighbor the following are shown: the state number, the error value, the actual bitstring representation, and the bit which is toggled in order to generate this neighbor.



(a)



(b)

Figure 8.3 Illustrating the difference between steepest and inertial-descent. a) Steepest-descent performs 4 neighborhood evaluations to find the optimal state, b) inertial-descent performs 3 neighborhood evaluations on the same search problem.

In move 1, bit 2 is toggled in order to move from state 10 to state 26. Successive moves lead search to state 16, which is optimal. Observe that the *only* information used from each neighborhood evaluation is the identity of the single neighbor yielding greatest improvement.

Figure 8.3b shows how inertial-descent performs on the same simple example. Inertial-descent has found the same optima as steepest-descent, but with one less neighborhood evaluation. Move 3 is made based upon the results of the neighborhood test in move 2. This represents a 25% savings relative to steepest-descent in this example. To step through the example in Figure 8.3b, move 1 is still to state 26, but now both the transition to state 26 and state 2 are shown in black in the ranked transitions displayed under the move. This signifies that both transitions lead to better matches. This other transition, toggling bit 3, is tried after move 1. In this case, it does not yield improvement and the entire neighborhood is again evaluated. However, after this evaluation, toggles of bits 5, 3 and 4 all yield better matches than state 26. Move 2 toggles bit 5 and moves search to state 24. Then from state 24, toggling bit 3 yields a better match and hence search moves to state 16.

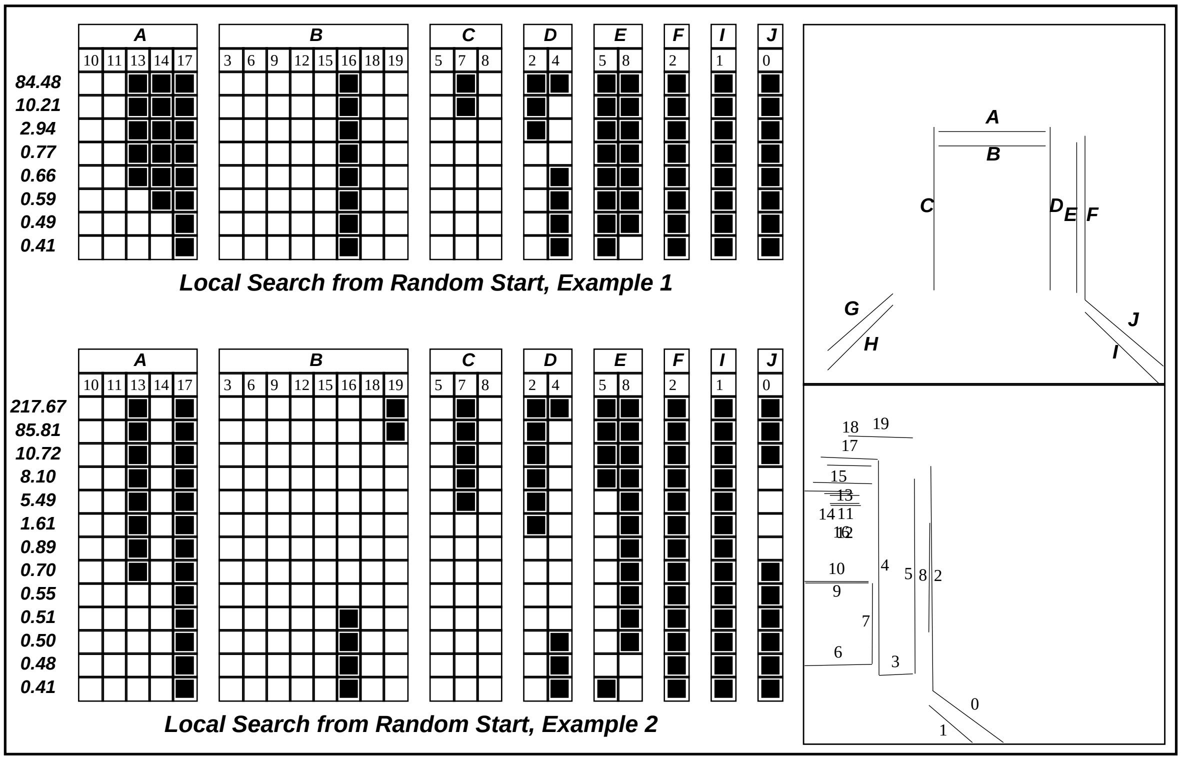


Figure 8.4 Example of the full-perspective-inertial-descent algorithm.

An example of the full-perspective-inertial-descent algorithm is presented below using Figures 8.4 and 8.5 for illustration. The matching problem is to match the landmark model shown in Figure 8.2 to data extracted from Image 2, shown in Figure 8.1b. The initial pose estimate is pose 5, shown in relation to the 3D landmark model in Figure 8.2a. The model and data line segments are shown on the right hand side of Figure 8.4. The model line segments are shown projected into the image plane as they would appear from pose 5.

Figure 8.4 also shows a complete trace of two independent runs of the inertial-descent

Table 8.1 The improvement lists generated by the inertial-descent algorithm each time it tests adding/removing a single pair. The highlighted pairs, applied in sequence, lead to improved matches.

Match	Improvements in Descending Order
84.48	$((\mathbf{D},\mathbf{4})(D,2)(E,5)(I,1)(J,0)(B,15)(B,12)\ldots$
10.21	$((\mathbf{C},\mathbf{7})(J,0)(I,1)(E,5)(B,12)(B,15)(A,11))$
2.94	$((\mathbf{D},\mathbf{2})(E,5)(E,8)(A,11)(B,12)(B,15))$
0.77	$((\mathbf{D},\mathbf{4})(\mathbf{A},\mathbf{13})(\mathbf{A},\mathbf{14})(\mathbf{E},\mathbf{8})(A,17)(B,15)(B,16))$
0.41	$()$

algorithm. As with previous illustrations, each successive row indicates a successively better match. Table 8.1 illustrates the sorted lists of transformations (bit toggles) yielding improvement for example 1 (Figure 8.4). The highlighted pairs at the head of the lists actually lead to improved matches. For the first three matches, inertial-descent didn't save any computation relative to steepest-descent. From the new match obtained by toggling the first pair on the list, the second pair no longer improved the match, and therefore all n neighbors of the new match were tested. However, for the fourth match, the list allowed the algorithm to apply four changes in correspondence in succession without expending effort testing alternatives. This reduced by nearly 50% the number of 3D pose calculations required to solve this example.

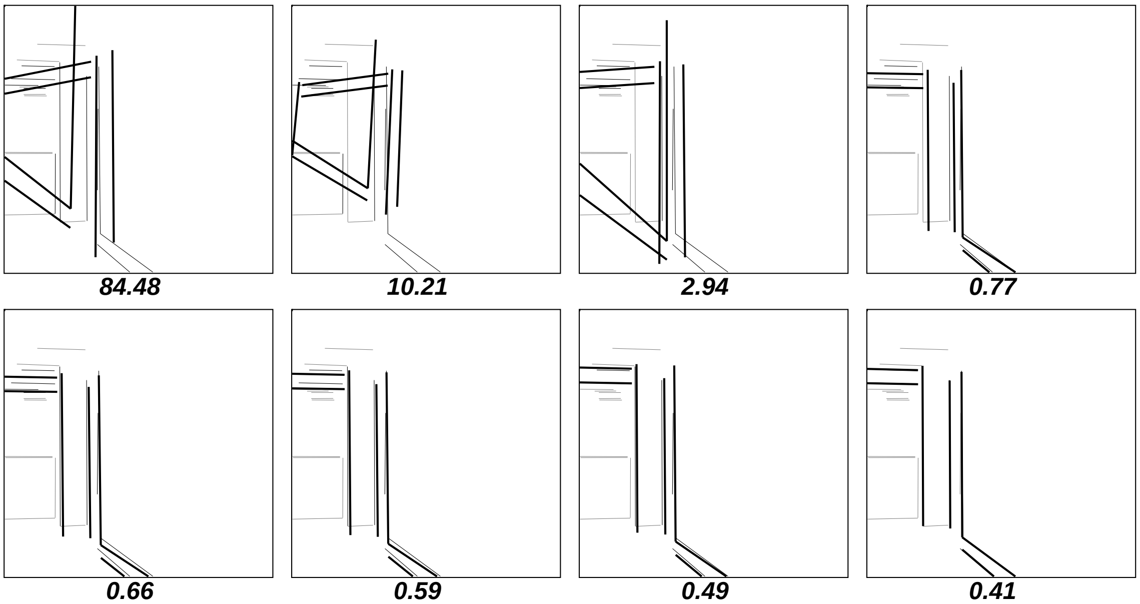


Figure 8.5 Landmark projections for full-perspective-inertial-descent example.

To emphasis that new 3D poses are generated during search, Figure 8.5 shows the projection of the landmark from these updated poses for the successively better matches found in example 1, Figure 8.4. The initial projection looks genuinely awful, as is to be expected given an initial random assignment of landmark to image features. It is the view of the landmark as it would appear to a robot downstairs and in a room off to the right of the hallway, and of course presuming the robot could see through floors and walls. One reason to show this initial projection is to emphasize that it need not itself be good, or even logical. What matters only is that local search is able to discover a path from this initial match to one which is better. This is the case here, and by the fourth match the match error has dropped from 84.48 to 0.77 and the projection is looking reasonable. The last four steps in the search path do not dramatically alter the appearance of the projected landmark. However, they do represent refinements in the correspondence mapping which improve the accuracy of the final robot pose estimate.

8.3.2 Hybrid Weak-perspective and Full-perspective Matching

Even using inertial-descent rather than steepest-descent, the cost of computing the 3D pose for all n neighbors in the Hamming-distance-1 neighborhood makes the full-perspective-inertial-descent algorithm rather slow. In rough terms, it is an order of magnitude faster to compute the pose of a neighboring match using the closed-form weak-perspective pose algorithm than it is to compute the pose with the iterative 3D pose algorihtm. Clearly, there is an incentive to see whether the weak-perspective algorithm might be substituted for the full-perspective algorithm when testing neighbors in the Hamming-distance-1 neighborhood, and this is done by the hybrid-weak-full-perspective algorithm presented in this section.

The hybrid-weak-full-perspective algorithm uses the weak-perspective match error E_{match} defined in Chapter 3. To be precise, the fitting of model to image features is done after the model has been projected into the image plane, and using the weak-perspective fitting algorithm of Chapter 4. However, and this is a major difference, *when a new match becomes the current match, 3D pose is recomputed and the model is reprojected.* In this way, the hybrid-weak-full-perspective is able to take account for full-perspective and recover from perspective distortions associated with an errorful initial pose estimate.

Figure 8.6 illustrates the essential idea of hybrid-weak-full-perspective by showing transitions to successively better match states in a search space. Nodes are drawn with the weak-perspective match error E_{match} displayed inside. These states are contrived. Their purpose is to show exactly where in the search process 3D pose is updated. To begin search, 3D pose is computed for the initial match and the model is projected into the image. Using these projected model features, the weak-perspective match error E_{match} is computed for all n Hamming-distance-1 neighbors. The neighbor with the lowest error is selected to be the next match. Search moves to this match, and in so doing, it recomputes the 3D pose and again projects the 3D model features into the 2D image, this time using the updated 3D pose estimate. This operation on the new match is indicated in Figure 8.6 by two bold lines connecting a node representing the selected match *prior* to the 3D pose update, and a node representing the match *after* the 3D pose update.

The possible *change* to E_{match} resulting from the 3D pose update puts a new twist on local search. Although generally E_{match} will still be lower than for the previous match, this need not always be the case. When E_{match} goes up, there are only a couple of options. One is to immediately terminate search. Another is to go back and start trying other neighbors in the hopes of finding one for which E_{match} stays down *after* 3D pose is updated. The third alternative, and the one used here, is to remember the match that was best and continue

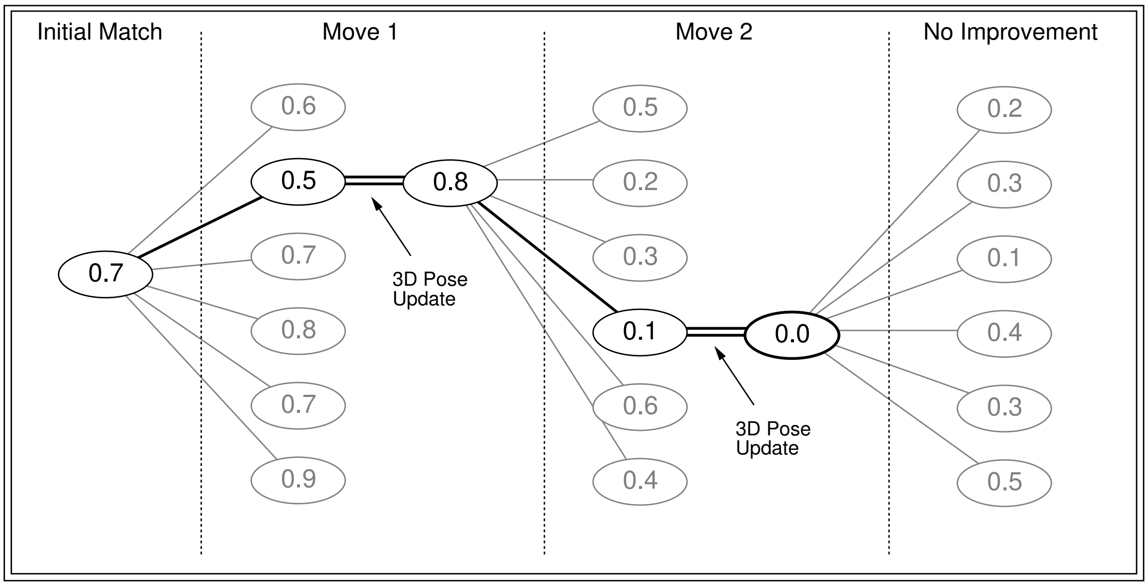


Figure 8.6 3D pose update placement in hybrid-weak-full-perspective search. The nodes in the search space are shown with the weak-perspective match error E_{match} written inside. The 3D pose is updated after a best neighbor is selected. Updating pose usually alters E_{match}.

searching from the new, worse, match.

The strategy of going on to explore matches worse than already seen is in principle somewhat antithetical to local search, which in concept *only* moves to better states. However, in practice it is quite common. In this thesis, it has already been used by the subset-convergent local search algorithm, which remembers a match as locally optimal in the Hamming-distance-1 neighborhood, and then initiates search from subsets of this match. For the hybrid-weak-full-perspective algorithm, the strategy for exploring worse matches is to continue search for some fixed number of moves. In all the experiments in this chapter, up to 10 exploratory moves are permitted. If, during this exploration, a match better than that being remembered is found, then local search continues and the previously best match is forgotten. If nothing better is found, then the best match being remembered is returned.

Checks must be put in to prevent search from cycling back to the already discovered best match. Cycling defeats the purpose of getting away from this match. There is a variant of local search called *tabu search* [40] (spelling is 'tabu') which embodies this idea: making moves which in some sense lead backwards taboo. A transformation for the hybrid-weak-full-perspective is taboo is it adds or removes a pair of model-data features already added or removed since the last best match was found. Thus, for example, it is not possible for local search to remove a pair in one move and then undo the effect by immediately adding it back in.

Figure 8.7 illustrates a single trial of the hybrid-weak-full-perspective algorithm with a randomly chosen initial correspondence is indicated by the first row in the table. The provision for adopting worse matches is seen in the increased match error between rows 1 and 2. The final solution is the correct, globally optimal, match. The landmark projected over the image segments for the first 5 matches and the optimal match are shown in Figure 8.8.

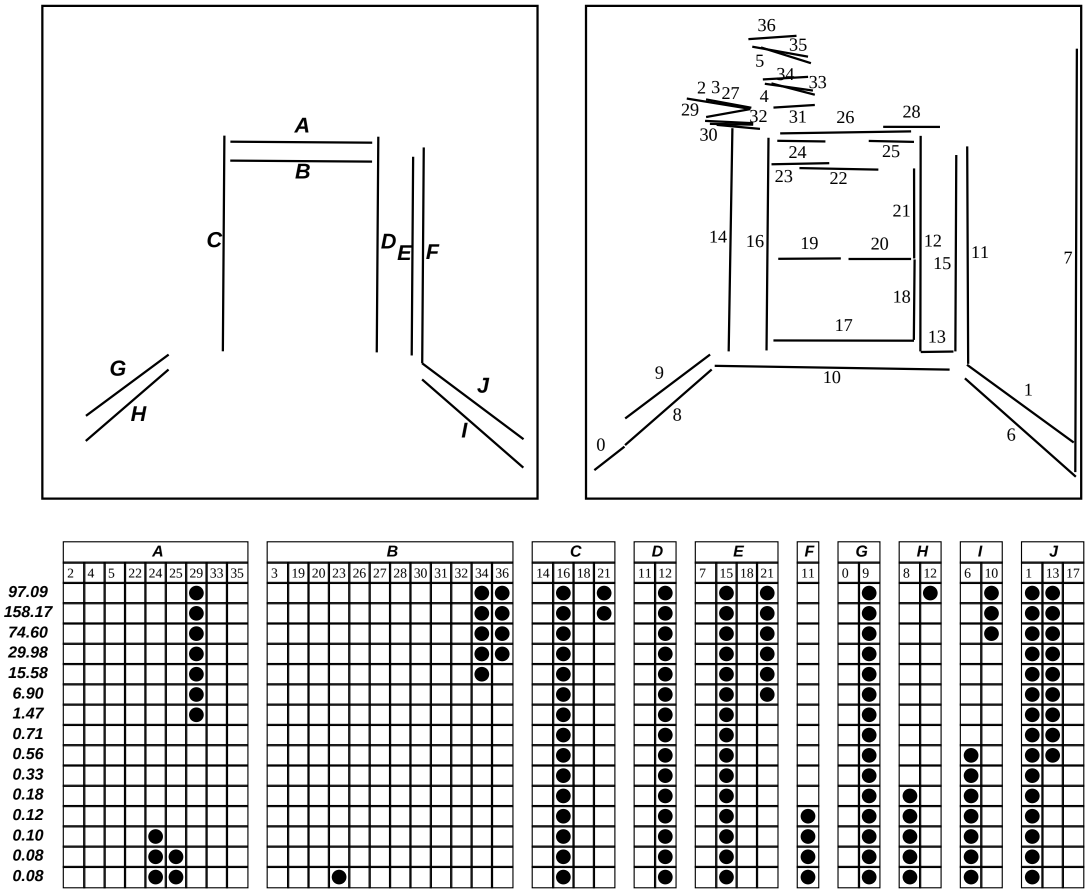


Figure 8.7 Example search trace for the hybrid-weak-full-perspective search. The match error does increase in the first move, but then decreases in all successive moves.

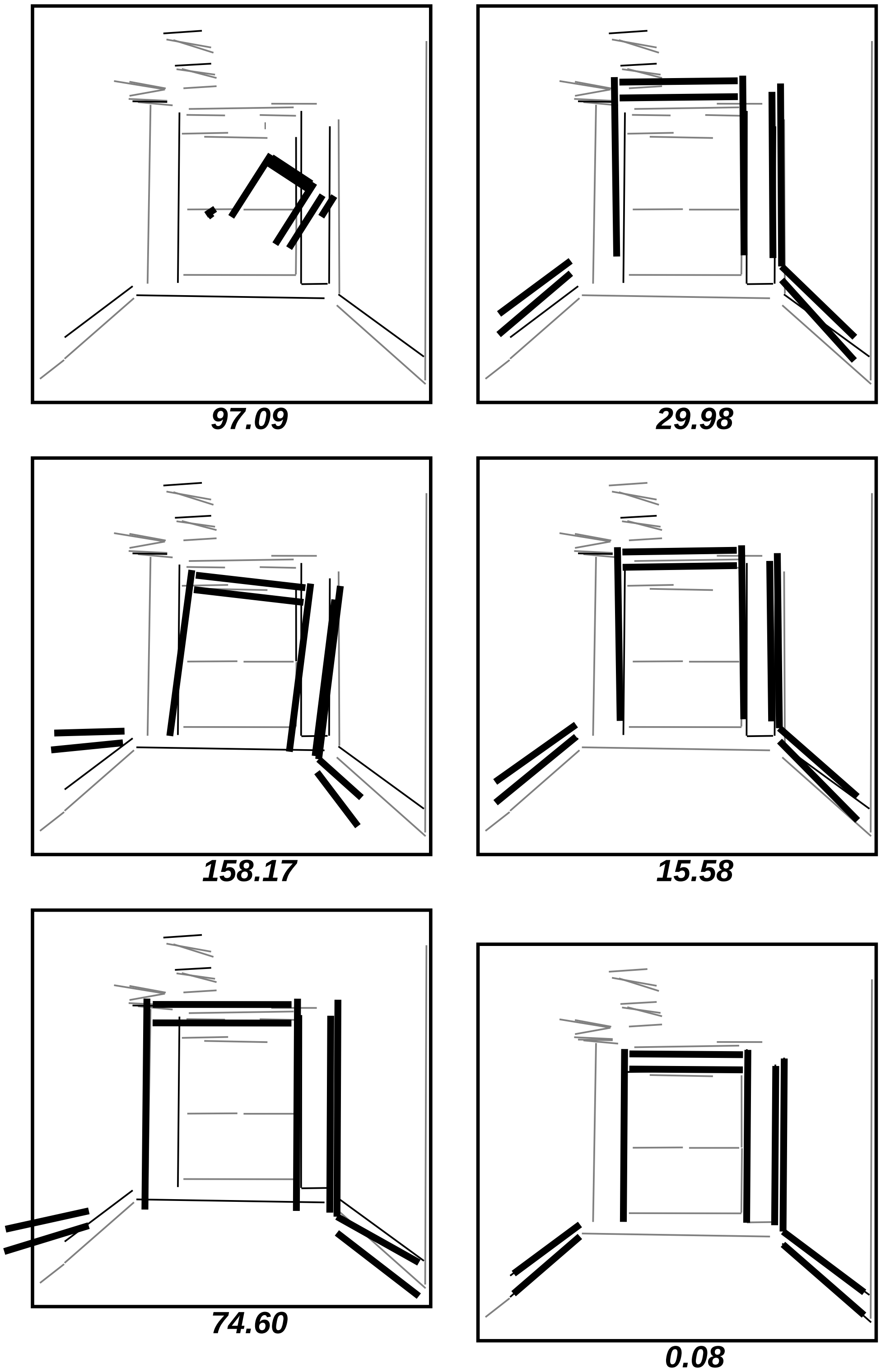


Figure 8.8 Landmark projections for the hybrid-weak-full-perspective search. The first 5 matches are shown along with the final optimal match.

8.3.3 Hybrid Subset-Convergent Matching

The hybrid-weak-full-perspective algorithm is based upon the Hamming-distance-1 neighborhood, and like the weak-perspective-steepest-descent algorithm introduced in Section 5.1, it can become stuck on undesirable local optima. To overcome this problem in the weak-perspective-steepest-descent algorithm, Section 5.4 introduced the subset-convergent local search algorithm. Recall this algorithm used the weak-perspective-steepest-descent to search for better matches from *subsets* of weak-perspective-steepest-descent locally optimal matches.

This section defines a hybrid-subset-convergent algorithm for full-perspective matching which is identical to the subset-convergent local search algorithm from Section 5.4, except that the weak-perspective-steepest-descent algorithm is replaced by the hybrid-weak-full-perspective algorithm. The expectation is that testing from subsets will help to eliminate undesirable local optima for full-perspective matching, just as it did for weak-perspective matching in Chapter 5. The experiments presented in Section 8.4 test this expectation, and whether the additional search pays off in terms of overall time required to reliably find globally optimal matches.

How the subsets are selected for the hybrid-subset-convergent requires some explanation. The subset selection algorithm described in Section 5.4.1 might select different subsets for different perspective projections of a 3D model. In principle, the subsets might be updated dynamically to reflect such changes during search. However, in practice this would be computationally expensive, and instead the subsets are selected based upon the projection of the 3D model from the *initial* 3D pose estimate. For example, the automatically selected 4 subsets for the landmark model as projected in Figure 8.7 are model segment pairs $(A, D), (B, C), (F, I)$ and (E, J).

8.3.4 Comparison and Review of Algorithms

Table 8.2 summarizes the major attributes of the three full-perspective matching algorithms just defined. The weak-perspective-steepest-descent defined in Section 5.1 is also included. In the next section, the performance of these four algorithms will be compared on a series of landmark recognition tasks. Referring back to Table 8.2 may prove a helpful way of recalling the salient differences between the four algorithms.

Algorithm type, either weak-perspective of full-perspective, is indicated for each. The algorithm type is a direct consequence of whether a full-perspective 3D pose update is performed after each move taken during local search. All four algorithms are shown to use the Hamming-distance-1 neighborhood. In addition, the hybrid-subset-convergent algorithm uses the larger neighborhood consisting of matches reached by searching from subsets of Hamming-distance-1 locally optimal matches. Only the full-perspective-inertial-descent algorithm uses the full-perspective 3D pose algorithm when testing matches in the Hamming-distance-1 neighborhood. All the other algorithms use the cheaper weak-perspective pose algorithm.

Each algorithm uses a somewhat different control strategy to direct search. The weak-perspective-steepest-descent algorithm uses the basic steepest-descent local search strategy. The full-perspective-inertial-descent algorithm uses the inertial-descent-strategy. The hybrid-weak-full-perspective algorithm uses the variant on steepest-descent that allows it to

Table 8.2 Comparing the attributes of four matching algorithms. The four algorithms are the weak-perspective-steepest-descent algorithm from Section 5.1 and the three full-perspective matching algorithms just presented.

		Weak-perspective-steepest-descent	**Full-perspective-inertial-descent**	**Hybrid-weak-full-perspective**	**Hybrid-subset-convergent**
Algorithm Type	Weak-perspective	X			
	Full-perspective		X	X	X
Pose Update Between Moves	Weak-perspective	X			
	Full-perspective		X	X	X
Local Neighborhood(s)	Hamming-distance-1	X	X	X	X
	Subset-convergent				X
Hamming-1 Neighbor Tests	Weak-perspective	X		X	X
	Full-perpective		X		
Local Search Stragetgy	Steepest-descent	X			
	Inertial-descent		X		
	Steepest-descent-tabu			X	
	Subset-convergent				X

move to worse matches and defines some transitions as taboo in order to avoid doubling back. Finally, the hybrid-subset-convergent algorithm uses the subset-convergent strategy to try to break out of matches which are local optima for the hybrid-weak-full-perspective algorithm.

8.4 Comparing Performance

This comparison is divided into two experiments. The first involves recognizing the hallway landmark segments in Image 1 from each of the 9 initial pose estimates shown in Figure 8.2a. This is a modestly difficult problem, in particular because of the perspective-sensitive nature of the domain. However, the positional errors in pose covered by this problem are, for the most part, rather modest. At worst, the robot's true position is no more than 5 feet from the true position.

A second experiment involves a greater error in the initial pose estimate. Image 1 in Figure 8.1 is taken from pose 5 and image 2 is taken from a pose obtained by moving the robot 11 feet closer to the door and rotating the robot several degrees to the right relative to pose 5. The task of this second experiment is to use landmark recognition to recover each pose given the other as the initial estimate.

8.4.1 Experiment 1: Recovering from Modest Pose Errors.

Each of the three algorithms introduced above is tested on the task of recovering the true pose, pose 5, from each of the estimates shown in Figure 8.2a. The true pose is 41.3 feet from the doorway and 4 feet from each of the two side walls. The estimates are obtained by introducing translation errors of 4 feet forward and backward and 2 feet side-to-side.

In addition, as a check on the importance of dealing with perspective during matching, the weak-perspective-steepest-descent algorithm from Section 5.1 is also run on each of these pose recovery problems. When using this algorithm, the 2D initial projections shown in Figure 8.2b are used as landmarks. The resulting optimal weak-perspective correspondence between model and image segments is then used once by the 3D pose algorithm in an attempt to recover the robot's true position.

In establishing the search spaces for each of the landmark matching problems, the predicted appearances of the landmarks, shown in Figure 8.2b, are used to determine the sets of candidate pairs. A pair $s = (m, d) \in M \times D$ is an element of set S if:

1) d is within 30 degrees of m.
2) d is within 128 pixels of m.
3) d is at least 1/4 the length of m.
4) d and m have the same sign-of-contrast.

The first test filters candidate pairs based upon relative orientation. The second is a rough proximity test. The image is 512 pixels across, so the threshold of 128 pixels represents one quarter of the distance across the image. The third test removes excessively small fragments from consideration. The thresholds for these three tests are picked based upon experience

Table 8.3 Size of candidate pair sets with/without the sign-of-contrast. Pairs S are tested for compatible sign-of-contrast, pairs $\bar{S}$ are not.

	Initial Pose Estimate								
	1	2	3	4	5	6	7	8	9
$\|S\|$	41	45	53	36	37	43	42	42	54
$\|\bar{S}\|$	87	94	112	75	77	92	89	94	112

with the domain, and are adequate to insure that the correct match is contained in the resultant search space.

The fourth constraint, sign-of-contrast, is very useful in a domain such as the hallway, where many of the segments are the result of surface markings rather than occlusion. For example, it is known with certainty that the top edge of the baseboards in the hallway will be white on top and black on the bottom, and therefore straight line segments extracted from the image with the opposite contrast need not be considered. However, sign-of-contrast is not a reliable constraint for occlusion edges, and so should not be used in these cases. To test the importance of the sign-of-contrast constraint in helping to solve these landmark matching problems, all the matching experiments in this section will be conducted using candidate pairs obtained with and without the sign-of-contrast constraint. The set S denotes pairs filtered on contrast and set $\bar{S}$ not filtered on contrast. Typically ignoring sign-of-contrast doubles the number of pairs, and Table 8.3 shows the exact size of these sets for all 9 initial pose estimates.

Figures 8.9 and 8.10 show specifically how changes in the initial pose estimate changes the exact set of pairs S included in a search space. Figure 8.9 shows the lettered model line segments as they appear projected from pose 5 and the numbered data segments extracted from image 1. Figure 8.10 shows explicitly which pairs of segments are considered candidate pairs based upon the initial projections. The match shown at the bottom is optimal for all 9 of the indicated search spaces. Scanning up the table one can confirm that these 11 pairs of segments are included in all 9 search spaces.

The full-perspective-inertial-descent, hybrid-weak-full-perspective, and hybrid-subset-convergent algorithms *recovered the true pose* in all the experiments conducted for this section [1]. There were a total of 18 matching problems resulting from the 9 initial pose estimates and the two sets of candidate pairs S and $\bar{S}$. All three full-perspective matching algorithms reliably found exactly the same optimal correspondence indicated in Figure 8.10. Consequently, the robot's true pose is recovered to within the accuracy bounds of our pose algorithm [71] [2]. The weak-perspective-steepest-descent algorithm did equally well for initial pose estimates 4, 5 and 6, where perspective has little effect. For the other cases performance was less reliable. Recovered pose differed from the true pose by 1 to 2 feet in 5 of the 6 cases, and by nearly 8 feet in one case.

Between 100 and 300 trials of random-start local search for each of the four algorithms

[1] Only the position portion of the pose estimate associated with a match is considered.

[2] Speaking loosely, for these problems the 3D pose algorithm appears accurate to within roughly 6 inches.

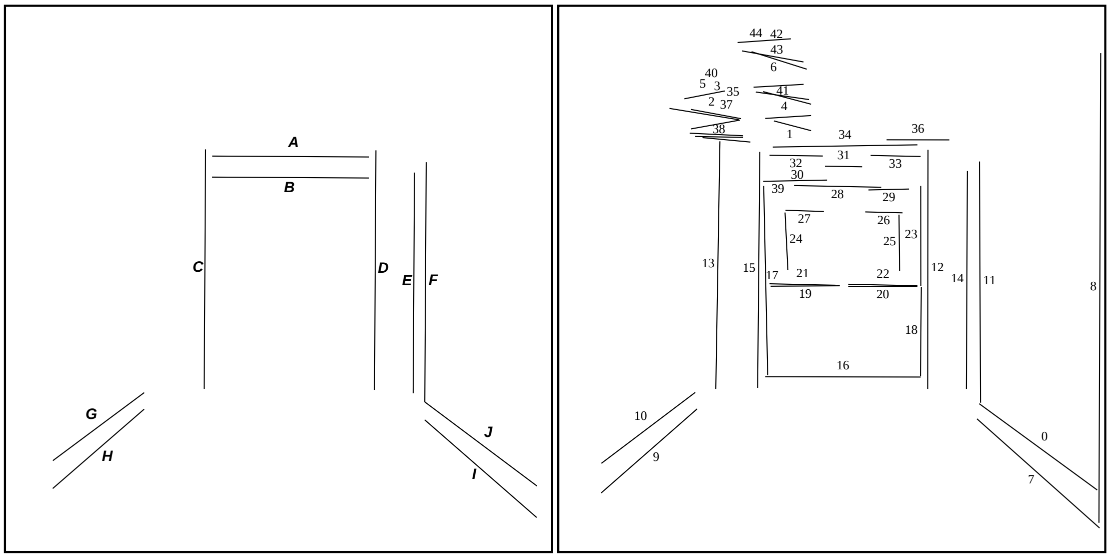


Figure 8.9 Labeled model and data segments for experiment 1.

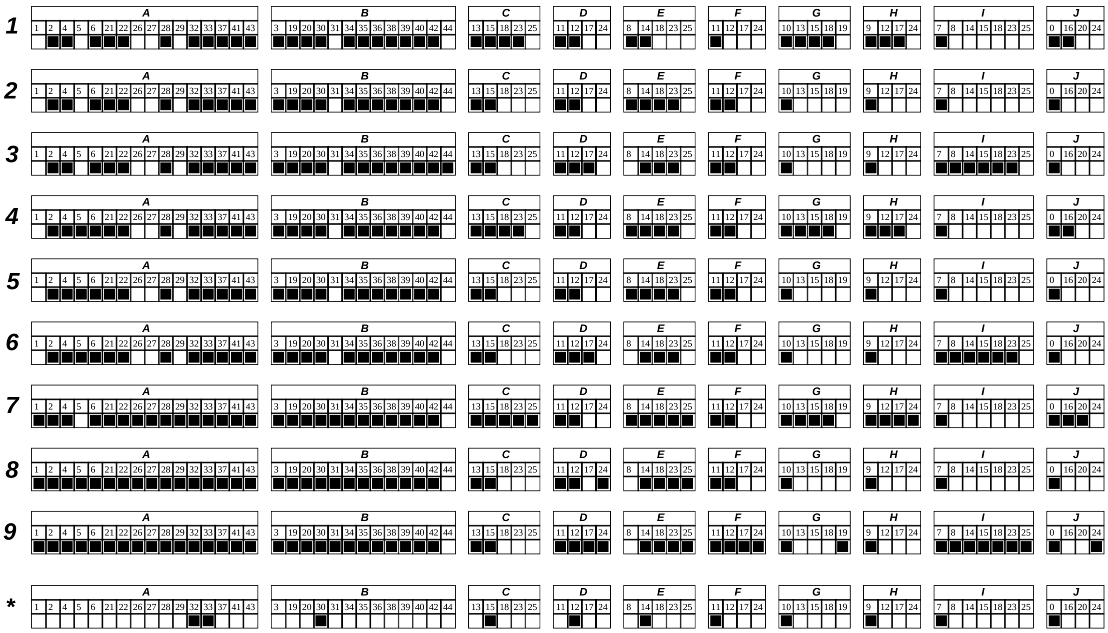


Figure 8.10 Candidate pairs for 9 poses and directed segments. The reader is asked to note that the semantics of a filled in square is different in the 9 numbered tables than in other tables seen before or the bottom table. In these 9 tables, a filled in square indicates a pair *is a member of S*. In the bottom table, indicated with and asterisk, filled in squares indicate pairs of segments belonging to the globally optimal match.

was run on each of the 18 matching problems. As was done for the study of the weak-perspective algorithms in Chapter 6, both the fractions of the runs finding the globally optimal match and the average run-time per trial were recorded for each algorithm applied to each problem. From this information, the probability of success estimates $\hat{P}_s$ for each algorithm on each problem are determined. These are plotted in Figures 8.11. Figures 8.11a shows $\hat{P}_s$ for each of the 9 pose estimates and candidate pairs S filtered by sign-of-contrast. Figure 8.11b shows $\hat{P}_s$ for the corresponding 9 problems using candidate pairs $\bar{S}$ in which the sign-of-contrast constraint is ignored.

The $\hat{P}_s$ values drop when sign-of-contrast is not taken into account. As measured by $\hat{P}_s$, matching problems not using the sign-of-contrast constraint are considerably more difficult to solve. For instance, the full-perspective-inertial-descent algorithm finds the globally optimal match on a single trial with probabilities between 0.17 and 0.55 with the sign-of-contrast constraint, while without the probabilities range from 0.06 and 0.22. A potential general explanation for the difference lies in size of the search spaces in the two cases. Without the sign-of-contrast constraint the set of candidate pairs roughly doubles in size, and the search space grows astronomically. There is another factor quite particular to this problem. Without the sign-of-contrast constraint, the upper and lower borders of the hallway baseboards are locally ambiguous. The tops of the baseboard in the model can quite readily match the bottoms in the image, and vice versa.

Another thing to observe from the plots in Figure8.11 is the relative performance of the four algorithms. Consistently in both cases, the hybrid-subset-convergent algorithm is nearly as good as or better than the full-perspective-inertial-descent algorithm. As is to be expected, the hybrid-subset-convergent algorithm outperforms the hybrid-weak-full-perspective algorithm. Finally, the weak-perspective-steepest-descent algorithm does worse than the other three in virtually all instances. Overall, this suggests that the hybrid-subset-convergent algorithm is reliably outperforming the other three in terms of the likelihood of finding the globally optimal match on a single trial.

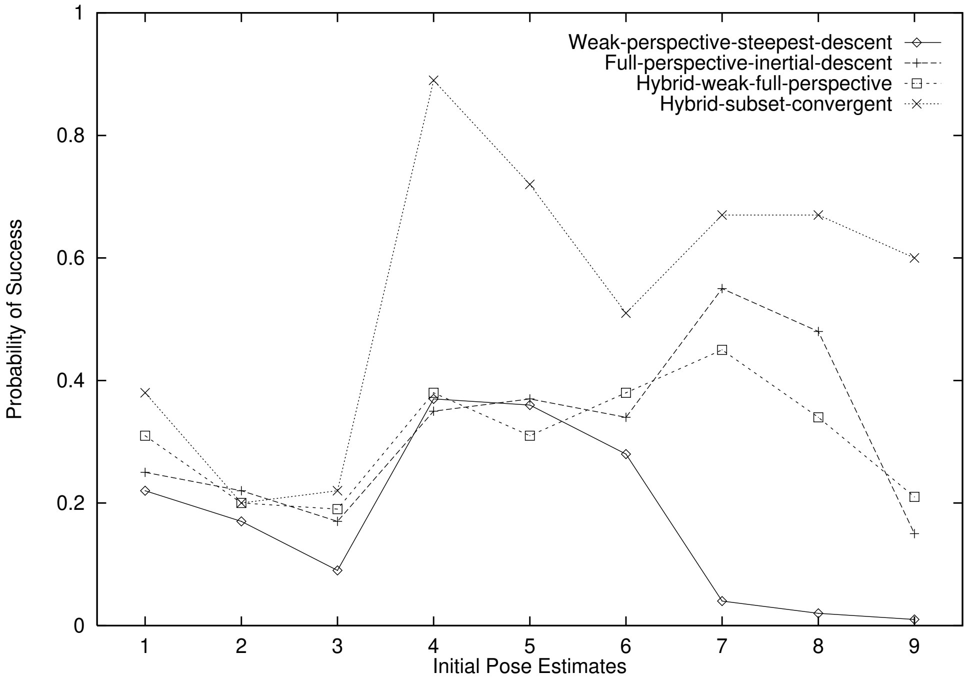
The next set of plots, Figures 8.12a and 8.12b, show the estimated run-time, $\hat{r}_s$, required to find the optimal match with probability 0.95 or better. This run-time is estimated in the same way run-times were estimated in Chapter 6, Section 6.6. The target confidence level has been lowered from 99% to 95%. Based upon the $\hat{P}_s$ values already shown, it is possible to estimate the number of required trials $\hat{t}_s$. Equation 5.5 from Chapter 5 is specialized to yield:

$$\hat{t}_s = \lceil \log_{\left(1-\hat{P}_s\right)} 0.05 \rceil . \tag{8.1}$$

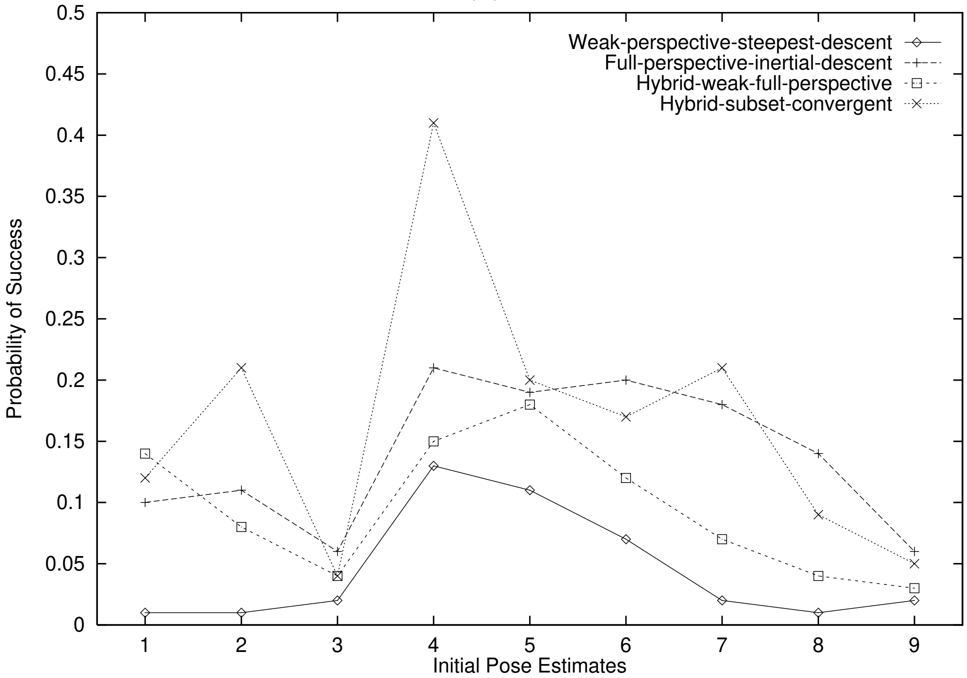
The estimated run-time $\hat{r}_s$ is the average time per trial r multiplied by $\hat{t}_s$.

As with the values for $\hat{P}_s$, perhaps the most striking difference is the increase in estimated run-time for the problems not utilizing the sign-of-contrast constraint. The range of the plot in Figure 8.12a is 50 minutes, while the range for the plot in Figure 8.12b is 5 hours. Again, it must be remembered that the search space in the latter case is incomparably larger. It should be noted that new C version of this algorithm running on a Decstation 5000 runs roughly 50 times faster than the Lisp Machine version reported here. When the sign-of-constrast constraint is used, this new version will never take longer than 5 seconds on these Hallway problems.

Overall, the hybrid-weak-full-perspective algorithm seems to be outperforming the others. The closest competition comes from the subset-convergent-hybrid algorithm. The

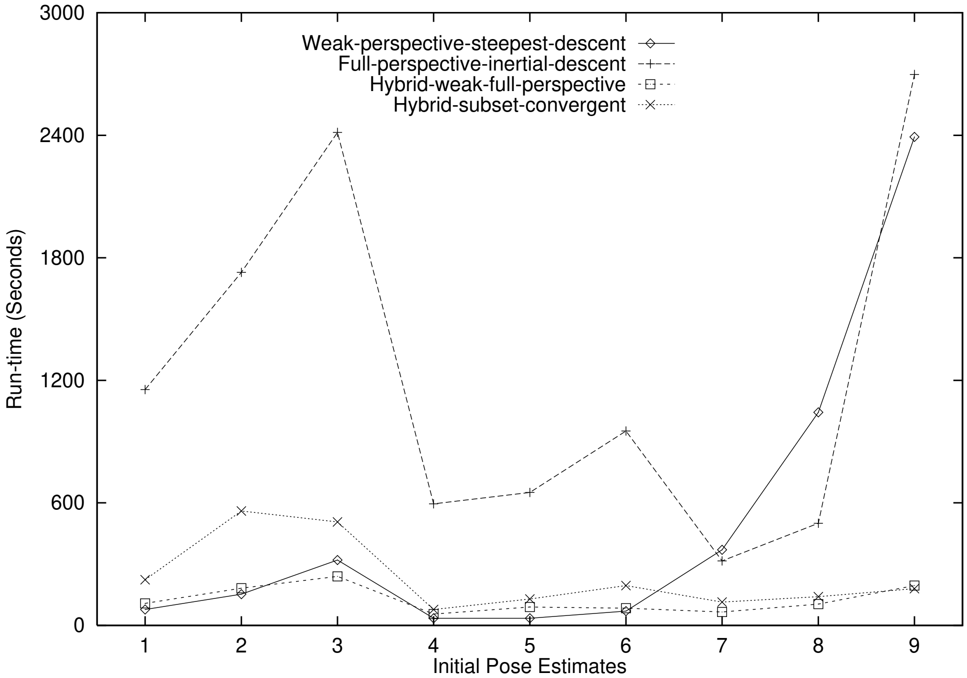


(a)

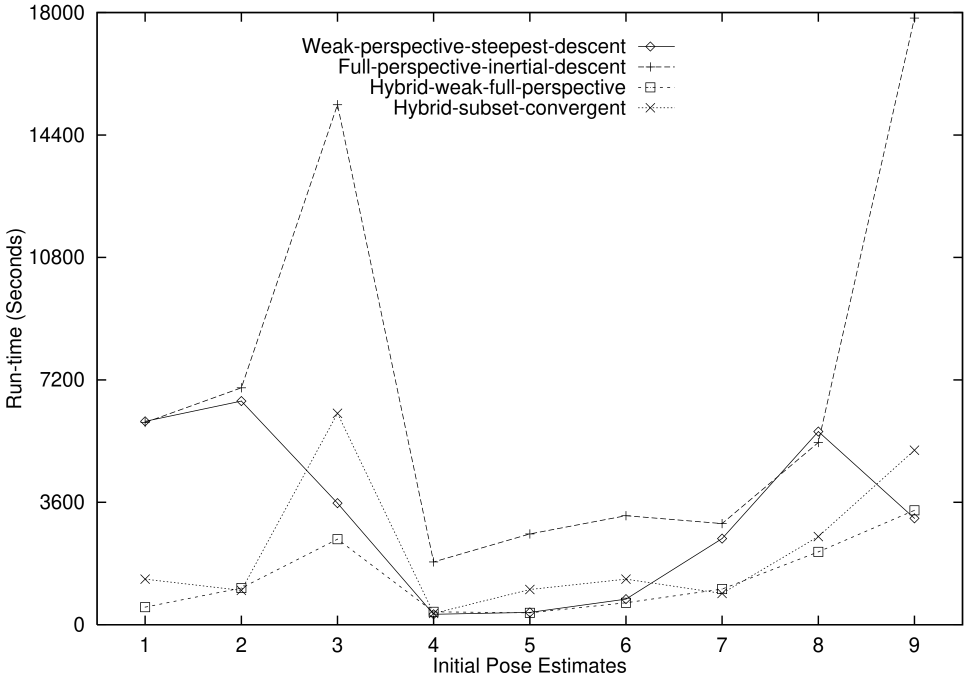


(b)

Figure 8.11 $\hat{P}_s$ for matching from 9 pose estimates. a) Candidate pairs filtered by sign-of-contrast, b) Candidate pairs not filtered by sign-of-contrast.



(a)



(b)

Figure 8.12 Estimated run-times for matching from 9 pose estimates. These times are to run sufficient trials to solve the each problem with 95% confidence. a) Using the sign-of-contrast constraint, the full-perspective-inertial-descent algorithm takes nearly an hour in the worst case. The hybrid-weak-full-perspective algorithm never takes more than 4 minutes. b) Without the sign-of-contrast constraint, the full-perspective-inertial-descent algorithm takes nearly 5 hours in the worst case. The hybrid-weak-full-perspective never takes more than an hour.

Table 8.4 Matching results when poses for Image 1 and Image 2 are confused. The probability of success $\hat{P}_s$, average run-time per trial r, required trials $\hat{t}_s$, and estimated run-time to solve the problem $\hat{r}_s$ are shown for each algorithm applied to each matching problem. Times are reported in seconds.

	Image 2 to Image 1				Image 1 to Image 2			
Algorithm.	$\hat{P}_s$	r	$\hat{t}_s$	$\hat{r}_s$	$\hat{P}_s$	r	$\hat{t}_s$	$\hat{r}_s$
Full-perspective-inertial-descent	0.29	172	9	1,548	0.18	63	18	1,134
Hybrid-weak-full-perspective	0.02	6	149	894	0.01	5	299	1,495
Hybrid-subset-convergent	0.09	25	32	800	0.06	11	49	539

worst run-time for the hybrid-weak-full-perspective algorithm is under 4 minutes with using the sign-of-contrast constraint, and just under an hour otherwise. The run-times for the hybrid-subset-convergent are tending to run roughly double that required by hybrid-weak-full-perspective alone. Run-times for the full-perspective-inertial-descent algorithm are much greater, taking up to nearly an hour with the sign-of-contrast constraint, and 5 hours otherwise.

These results tell us that although the full-perspective-inertial-descent and hybrid-subset-convergent algorithms do better on individual trials, they also take more time on each trial. Moreover, neither does sufficiently better to warrant the additional time per trial. It appears, for the set of problems tested here, that the hybrid-weak-full-perspective algorithm is the best choice.

8.4.2 Experiment 2: Recovering from Larger Pose Errors.

In this second experiment, the pose for image 1 shown in Figure 8.1a is given as the initial estimate of robot pose when the true pose is that for image 2 shown in Figure 8.1b, and vice versa. Both the landmark projections from the initial pose estimates, as well as the best matches, are shown for these two problems in Figure 8.13.

These are more difficult matching problems than those shown in the previous section, and 300 trials are used to estimate $\hat{P}_s$. Two factors make these more difficult problems. First, in one case half the expected segments are not visible while in the other many unexpected segments are visible. Second, to account for the greater uncertainty in pose, the proximity constraint used to select the set of candidate pairs S is relaxed to include all segments d within 256 pixels of a model segment m.

The results for the second experiment are summarized in Table 8.4. Only the three algorithms capable of handling full-perspective were tested on these problems. The table reports the estimated probability of success $\hat{P}_s$, the average run-time per trial in seconds r, the number of trials $\hat{t}_s$ required to find the optimal match with 95% confidence, and finally the estimated run-time $\hat{r}_s$ required to confidently find the optimal match. Comparing the run-times $\hat{r}_s$ using each algorithm, the hybrid-subset-convergent is superior to the other two. The highest of the two run-times for the hybrid-subset-convergent algorithm is still lower than the best time of either of the other two algorithms.

These results provide an interesting example of what is a basic tradeoff in local search: is

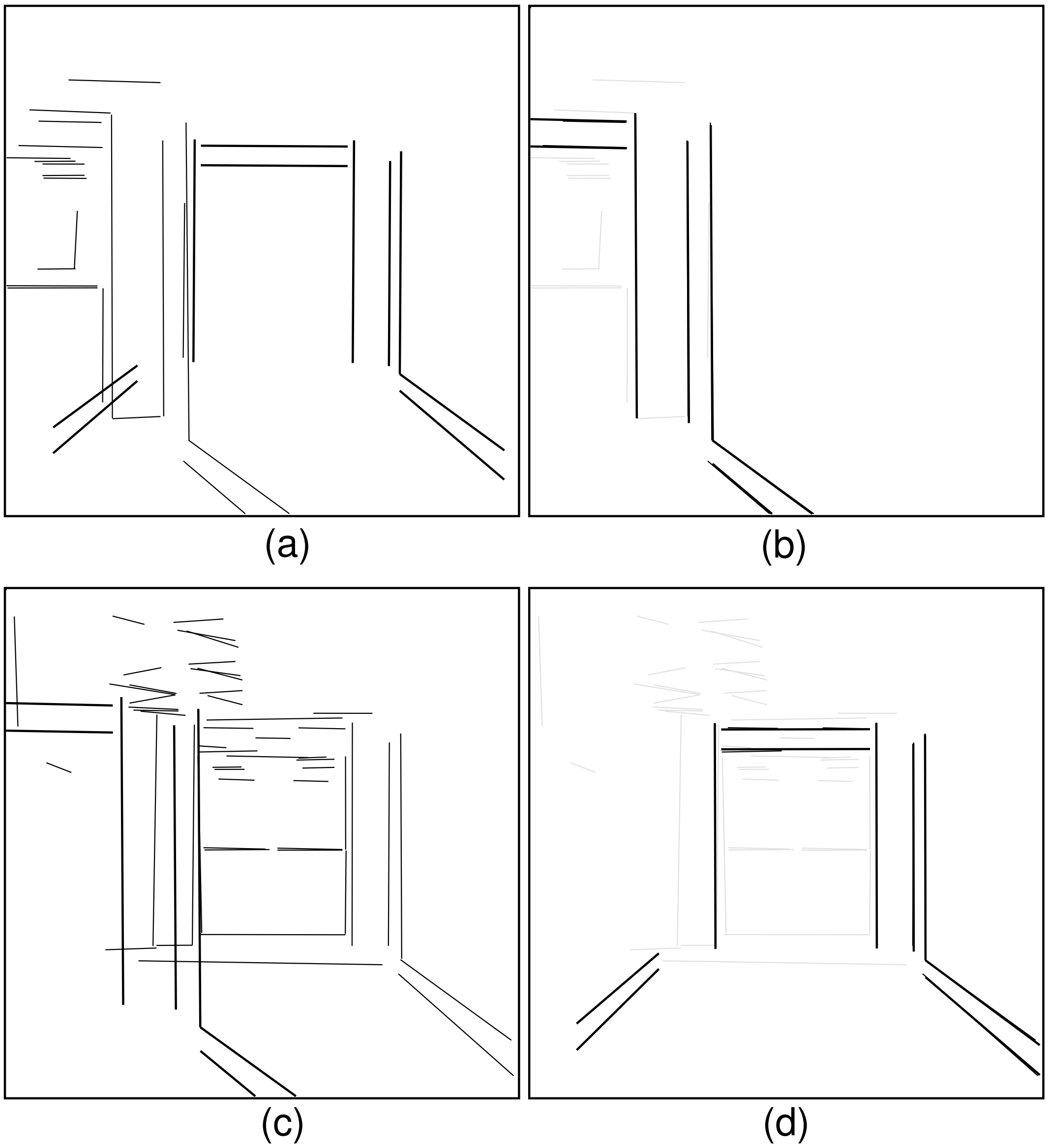


Figure 8.13 Confusing poses for images 1 and 2. a) landmark projected over image 2 data as it would appear from image 1 pose, b) successful recovery of correct match and pose, c) landmark projected over image 1 data as it would be seen from image 2 pose. d) successful recovery of correct match and pose.

it better to run a few expensive trials that are more likely to succeed, or instead to run many cheaper trals that are less likely to succeed? The full-perspective-inertial-descent and hybrid-weak-and-full-perspective algorithms seem to lie at opposite extremes of this tradeoff. The full-perspective-inertial-descent algorithm is slow but reliable, finding the globally optimal match with modestly high probability on each trial, but expending considerable computation in the process. In contrast, the hybrid-weak-and-full-perspective only finds the optimal match in 1 or 2 out of every 100 trials, but each trial runs in comparatively little time. In this particular case, the two factors balance, and the run-time $\hat{r}_s$ is lower for the full-perspective-inertial-descent algorithm on one of the two matching problems, and higher on the other problem.

In terms of the tradeoff just mentioned, the hybrid-subset-convergent lies in between the extremes staked out by the other two. In comparing the hybrid-subset-convergent algorithm and the hybrid-weak-and-full-perspective algorithm , the increase in $\hat{P}_s$ more than makes up for the increased time taken per trial. For the hybrid-subset-convergent compared to the full-perspective-inertial-descent algorithm, each trial is sufficiently faster to make up for the relative drop in $\hat{P}_s$.

The hybrid-weak-and-full-perspective algorithm performed best on the modest pose error problems studied in the previous section, while the hybrid-subset-convergent algorithm performed best on the larger pose error problems in this section. This is a rather small amount of data upon which to draw general conclusions; however, it does seem to suggest a trend. As matching problems become more difficult, in that there is greater uncertainty in the initial pose, the hybrid-subset-convergent algorithm will probably perform better relative to the hybrid-weak-full-perspective algorithm. This observed trend suggests a general conjecture: for relatively pose constrained problems the simpler forms of local search are adequate and even preferable to subset-convergent search, while for less pose constrained problems the subset-convergent algorithms appears to be superior.

8.5 Conclusion

Weak-perspective matching algorithms are inadequate for landmark-based robot navigation in a hallway, at least for the types of line-segment-based geometric matching problems studied here. The failure of the weak-perspective-steepest-descent algorithm has been demonstrated. By implication, the other basic approaches to geometric matching reviewed in Chapter 2, and which have yet to be demonstrated on perspective-sensitive matching problems, can also be expected to fail.

It was the indoor robot navigation problems presented here which provided powerful incentives to develop quantitatively accurate full-perspective matching algorithms. These algorithms are needed, in part, because often those features which most accurately determine robot pose are the same features which are most sensitive to perspective effects. This is illustrated here by the baseboards in the hallway. It is the baseboards, far more than any other features visible in the images shown in Figure 8.1, which must be found in order to estimate the placement of the robot left-to-right in the hallway. Finding the doorway at the end of the hallway constrains the distance of the robot from the doorway, but offers little side-to-side constraint.

The experiments with different full-perspective matching algorithms in this chapter have shown that it is the hybrid algorithms, which blend the use of weak-perspective and full-perspective evaluation of matches *during* search, which perform best. They categorically beat the weak-perspective algorithm, which fails when the initial estimate of the 3D object's pose introduces perspective distortion into the initial projection of the model. In terms of run-time, the hybrid algorithms dramatically outperform the more brute-force full-perspective-inertial-descent algorithm. The hybrid-weak-full-perspective algorithm appears to perform somewhat better in terms of overall run-time for the modest pose error problems in Section 8.4.1, while the hybrid-subset-convergent version seemed more economical on the problems with larger initial pose error tested in Sections 8.4.2.

The experiments presented here systematically explored performance for typical hallway navigation problems, and show conclusively that the local search approach to full-perspective matching works on these problems. Determining how these results will generalize to other domains, such as outdoor navigation, will require further study. It is expected that the local search matching approach will generalize well. However, there are many details in need of further study.

CHAPTER 9

CONCLUSION

This thesis has developed a new approach to finding the best match between a geometric object model and noisy, fragmented, cluttered image features. The approach adapts a simple and powerful idea from combinatorial optimization to a common problem in computer vision. The simple idea is local search, which is one of the most practical ways of finding near optimal solutions to difficult combinatorial optimization problems. The common problem is matching a geometric object model to features extracted from an image. Matching has been further defined here as the problem of finding the optimal correspondence mapping between model and image features subject to the constraint that model and image must be related by a best-fit pose transformation. Local search has been shown to reliably solve these optimal matching problems under a wide range of circumstances. This conclusion will summarize what has been presented, the strengths and weaknesses, and the opportunities for future work related to local search matching.

9.1 Review

Chapters 3 and 4 formalize the geometric matching problem as a combinatorial optimization problem. They emphasize the factors which go into ranking potential matches, and especially the fundamental need to estimate object pose as part of match evaluation. An original and highly efficient closed form solution to the problem of fitting a line model under weak-perspective permits evaluation of globally registered matches. The match error developed here, and in particular the omission term, is a valuable and original contribution.

Chapter 5 presents new local search algorithms for efficiently finding globally optimal geometric matches. Steepest-descent local search using the Hamming-distance-1 neighborhood is introduced, as is random random sampling as a way of overcoming local optima and of estimating the difficulty of a geometric matching problems. Of special importance, the subset-convergent local search technique is introduced and demonstrated on quite difficult matching problems; including the Deer and Giraffe problem at the close of the chapter.

Chapter 6 explores the range of problems that can be solved using random-start subset-convergent local search. A suite of weak-perspective matching problems test the algorithm under a variety of adverse conditions, many of which have been selected specifically because they are known to cause problems for other matching techniques. The object models are corrupted by fragmentation and skewing. Random clutter is added in some cases, and multiple instances of the same object in others. The object models are both very simple and modestly complex, and in one case the objet model is highly symmetric. Under all these conditions, subset-convergent local search reliably finds the globally optimal match, and in all but a few cases, 100 independent trials of local search is sufficient to find the globally optimal match with better than 99% confidence.

Chapter 6 also estimates run-time growth as a function of problem size. Regression analysis finds that the best-fit polynomial to the run-times as a function of problem size n has an exponent of roughly 1.5. Rounding up to the next whole integer, and extrapolating from these measurements, it is conjectured that average case computational complexity is $O(n^2)$. It is readily conceded that this might be viewed as an optimistic conjecture based upon extrapolation from empirical data. However, failure to observe exponential behavior does not imply a lack of a practical bound on problem size. For the TI Explorer II lisp Machine implementation of subset-convergent local search, this threshold is reached for n around $1,000$. The largest problem tested in this thesis has an n of $1,296$.

The final step in this thesis generalizes local search matching to solve full-perspective matching problems. These are matching problems which cannot be solved without quantitatively accounting for full 3D perspective during matching. The matching algorithms presented in Chapter 8 appear to be the first to be able to find optimal matches under these conditions. This extension would not have been possible without the use of the 3D pose algorithm developed by Kumar [70, 71]. Thanks to this algorithm, it is possible to perform landmark-based robot navigation in a hallway: a domain with pronounced perspective effects.

Three different full-perspective matching algorithms are presented in Chapter 8, and their performance compared on a set of landmark recognition problems. As a check on the importance of accounting for full-perspective, the Hamming-distance-1 steepest-descent algorithm from Chapter 5 is also tested. Several significant conclusions can be drawn from these tests. First, the pronounced perspective effects encountered in the hallway make the weak-perspective matching algorithm unreliable for robot navigation, and in general any problem where there is significant disparity in depth of model features and varying viewpoint. Second, all three of the full-perspective matching algorithms solved all the problems presented to them, and any of them could be used for robot navigation in this domain. A hybrid algorithm, utilizing both weak-perspective and full-perspective fitting during matching, is more efficient than an algorithm which always uses full-perspective, and this hybrid algorithm requires no more than twice the time of the fast but unreliable weak-perspective matching algorithm.

9.2 The Problems

9.2.1 Complexity and Object Indexing

The biggest problem facing someone wishing to use the algorithms developed in this thesis is essentially the same as would be faced with any of the known general geometric matching techniques: average run-times grow roughly as a function of the number of paired features squared, if not faster. Worse yet, in the absence of an initial constraint on object pose, increasing model size m linearly increases n quadratically, and run-times tend to grow as function of m^4. For local search matching, the failure to find evidence of exponential run-time growth in Chapter 6 is of considerable theoretical interest, and it makes the approach attractive for problems of modest size. However, it does not help someone wishing to find a model with 100 segments in an image containing $10,000$ data segments.

A problem with $n = 1,000,000$ is three orders of magnitude larger than any problem solved in this thesis, and by extrapolation from equation 6.6 in Chapter 6, it would take nearly a century to solve on a TI Explorer II Lisp Machine. Even accounting for faster and parallel implementations, projected run-times indicate multiple trials of Subset-convergent local search initiated from random matches is, by itself, not appropriate for finding large models in even larger data sets. As seen in many of the problems shown in this thesis, it is best applied to problems for which there is an initial object pose estimate, or some other source of ancillary information which limits possible pairings between features.

Asking where an initial pose estimate comes from leads to the second problem associated with local search matching: what prompted that algorithm to look for this particular object in the first place. Like most of the previous work on geometric matching, local search matching assumes a particular object is worth looking for in an image. Local search matching does not deal with the problem of indexing into a database of geometric object models and selecting an object likely to be visible in an image. As covered in Chapter 2, geometric hashing is the only one of the four main approaches to geometric matching to deal with object indexing.

The lack of indexing, and the apparent average case complexity of local search matching, means it may be used successfully *by itself* when:

- Object indexing is not necessary, and either
- models and image data sets are of modest size, or
- there is an initial estimate of the object's pose, or
- ancillary information limits the possible pairing of features.

One reason local search matching is well suited to landmark-based robot navigation is that, as a problem, it typically satisfies these conditions. When these conditions are not met, then local search matching must be used in *conjunction* with other tools. Local search matching may be coupled with any algorithm which generates object hypotheses, and used to test and rank hypotheses. It has been used in this capacity as part of Draper's work on learning knowledge directed scene interpretation strategies [31, 30].

9.2.2 Algorithm Tuning

Presuming someone wishes to apply local search matching to a problem satisfying the above conditions, there is another area of concern, and that can be briefly described as algorithm tuning. Despite efforts to keep the local search approach to matching simple, success lies in the details, and there are a number of important details. For example, success can depend upon proper selection of terms to include in the match error, and the relative weights assigned to each term. Recall the telephone pole matching example discussed in Section 5.4.3 and shown in Figure 5.13. To have the correct match be the globally optimal match, it was necessary to evenly weight omission error for each of the 3 segments forming the telephone pole model. Using omission error weighted by relative length not surprisingly discounted the importance of the shorter cross bar and caused the match error function to prefer a match to the street lamp. Ideally, one set of match error weights and terms would be appropriate to all the problems presented in this thesis. However, this is not the case, and thought must be given to how match error ranks competing matches.

In addition, as the match error case 1 versus case 2 performance on the multiple instance Dandelion problems in Chapter 6 showed, variations in the form or parameterization of the match error function not only change the ranking of locally optimal matches, they change the very matches found by local search. This second order effect means treating the match error as a simple statement of what constitutes a desirable match may be inadequate. It may be necessary to also analyze how the match error leads the algorithms *during* search. In defense of local search matching, all significant matching algorithms have parameters, and while tuning issues for local search matching may be different than those for other algorithms, local search matching is not alone in being sensitive to details.

There is a very specific tuning issue not fully dealt with in this thesis, and it concerns how a match with only half a model projecting into the scene should be compared to one with the model entirely in view? This is different from omission due to errorful feature detection or occlusion by an unidentified object, and concerns the case in which the best-fit pose indicates portions of the model project outside the bounds of the image. On the one hand, it seems odd to penalize a match with an omission score for line segments clearly not in view. On the other hand, not introducing a penalty leaves the algorithm open to finding good matches by forcing much of the model out of view.

How to treat features projecting out of the image is a practical concern. Recall from Section 8.4.2 an experiment where half the model projected outside the image. A clipping algorithm included in the 3D-to-2D projection was used in this experiment, and therefore no omission penalty was exacted for portions of the model projecting outside the image. In this case, success on the problem suggests this was the right way to handle the problem. However, in recent and very preliminary experiments with full-perspective matching on the scene shown in Figure 5.12, local search found the ground-truth correct match, but ranked it lower than alternatives for which most of the near telephone pole projected outside of the image. Local search had found several correspondences which, although incorrect compared to ground-truth, let it discount the importance of the nearest telephone pole by forcing it to project outside of the image. In general, further work on the match error function is required, both to better understand and handle this problem, as well as to facilitate comparisons between matches to different object models.

9.2.3 Random-Start Local Search Probably Succeeds

The probabilistic nature of random-start local search matching is a potential source of trouble. As developed and tested in this thesis, local search matching is a way of finding the globally optimal match with high confidence. The algorithm will not *always* find the globally optimal match, and when run on a sufficient number of problems, it *will* find sub-optimal matches on some small number. These probabilities can be characterized for an application domain, and intelligent choices made to trade run-time costs for confidence of match. However, the approach does not guarantee an optimal solution to every problem.

For many of the foreseeable applications the lack of guaranteed success is not excessively troubling. For instance, on industrial inspection tasks low probability inspection failures at known rates are an accepted fact of life. More generally, it is not a problem for semi-autonomous systems where occasional human intervention is acceptable. It is also not a problem for fully automated systems with adequate failure recovery mechanisms. However, local search matching must not be used under circumstances where finding occasional sub-optimal matches will have catastrophic consequences.

9.3 The Strengths

9.3.1 Robust Performance

Probably the most attractive aspect of local search matching is its robust performance when working with poor quality data, multiple model instances and symmetric models. It is the only demonstrated technique for finding optimal many-to-many mappings between geometric object and image features, and in part this accounts for its strength on problems involving highly fragmented image features. Recall the examples in Figures 1.4 and 1.6. Many-to-many mappings also handle polygonal approximations to curved objects as demonstrated with the Leaf example in Chapter 6.

Local search matching is robust also because it seeks the best match or matches present in an image regardless of their absolute quality. This contrasts with tree search [48], which through its need for a termination criteria, relies on a prediction of expected match quality; if the criteria is set too high, then search will expand an exponential space of partial matches, if the criteria is too low, then matches of higher quality will be ignored. Random-start local search discovers a set of locally optimal matches, and with high probability this set will contain the globally optimal match.

Local search matching randomly samples the space of possible matches in a way that does not rely upon locally distinctive key-features, and this too makes the technique robust. This manifests itself in two ways. First, local search matching excels at problems involving distracting background clutter such as the Deer and Giraffe matching problems presented at the close of Chapter 5. Second, as demonstrated in Chapter 6, internal symmetries (the Dandelion), or excessive simplicity (the Pole), do not prevent subset-convergent local search from picking out the best match against random clutter, and with additional effort, from picking out the best match from among several distinct model instances. Key-features are problematic in highly cluttered scenes and when using either very simple models or highly symmetric models.

9.3.2 Trivial to Run in Parallel

With massively parallel computers becoming available, one of the most important questions about a computer vision algorithm is how it will perform on these machines. Here lies another major strength of local search matching. The algorithms presented in this thesis have an obvious parallel forms; each independent trial is run in parallel. Looking only several years into the future, a geometric matching algorithm used for an important application will have access to hundreds of moderately powerful processors. A quick sketch of what this promises is encouraging.

For the sake of argument, assume a production version of the current TI Explorer Lisp

Machine implementation were ported to a parallel machine. Presume a 200 fold speedup running on each processor. This is conservative, since a recently implemented C version on a Decstation 5000 appears to be between 50 and 75 times faster than the Lisp Machine version. Next, presume a 100 processor parallel machine. Under these assumptions, the algorithm could run up to 20,000 times faster than reported for the TI Explorer. The hardest Hallway matching problem using directed line segments (Chapter 7) required 240 seconds on the TI Explorer, and could therefore require as little as 12 milliseconds running on this hypothesized parallel machine. On such a machine, these algorithms could easily run at frame rate as part of a real-time application.

9.3.3 Broad Applicability

The matching problems solved in this thesis were selected in part to show the broad applicability of local search matching. It has been demonstrated on weak-perspective model-to-image matching problems in aerial photography (Figure 1.4) and outdoor scenes [11]. It has been demonstrated on weak-perspective model-to-image matching problems such as those that arise in outdoor (Figure 5.13) landmark-based vehicle navigation [12, 34], and it has been demonstrated on full-perspective landmark identification (Chapter 8) on indoor scenes [9, 10, 8]. Despite the match error being designed for model-to-image matching problems, it has been used for image-to-image matching in order to track a moving object (Figure 1.6). It has been tested on synthetic data with large amounts of clutter (Figure 5.14), multiple model instances (Figures 6.4 and 6.5) and highly symmetric models (Figure 6.10).

Even with the breadth of problems presented in this thesis, they represent only a fraction of all the problems to which the algorithm has been a applied. It has been used to support the work of Draper [31, 30] on learning object recognition strategies for outdoor scenes, where it has been used to derive ground-truth pose estimates and as a knowledge source for verifying object hypotheses. Recent work with Collins [24] has used local search matching in conjunction with vanishing point analysis to register aerial photographs taken from widely different viewpoints. Vanishing point analysis is used to unwarp the perspective views of the ground plane, and then local search matching finds the optimal correspondence mapping and similarity transform relating features in the two images.

9.3.4 Quantitatively Accurate Full-Perspective Matching

Of all the strengths of local search matching, perhaps the most striking is the relative ease with which it generalizes from essentially 2D weak-perspective matching problems to full-perspective matching. As reviewed in Chapter 2, a great deal of work has been done on weak-perspective matching, while there has been almost no work on quantitatively accurate methods for performing full-perspective matching. With the possible exception of the work of Lowe [82], which accurately accounts for full-perspective but considers only modest correspondence problems arising in the context of object tracking, there appears to be no precedent for the full-perspective matching results presented in this thesis.

9.4 The Future

This thesis presents the first concerted effort to apply the principles of local search to the problem of geometric matching. As just reviewed, the approach has some problems and many strengths. However, much remains to be done to both refine and extend the approach. This section will consider six areas in which additional work promises major advancements. Briefly, these areas are:

1. Future work should further explore several of the more important algorithm design choices and parameter settings used in this thesis.

2. Initializing local search with 'better than random' initial matches may lead to utilizing model-directed feature grouping as part of matching.

3. The match evaluation should be tied into probabilistic models of image formation.

4. Partial symmetries in objects should be detected off-line and used as neighborhood permutations during local search.

5. Alternative optimization techniques should be tested as substitutes for local search: for example stochastic relaxation and parallel genetic algorithms.

6. The full-perspective matching work presented in Chapters 7 and 8 should be extended to consider alternative measures of fit and alternative forms of hybrid matching.

7. Local search matching promises to facilitate model-directed sensor fusion: for example using LADAR and CCD image data.

9.4.1 Detailed Exploration of Algorithm Parameters

A number of the factors which go into tuning the local search algorithms appear to be important. For instance, the results for the two different match error cases in Chapter 6 raise basic questions which deserve further attention. The change in performance between match error case 1 and 2 is quite dramatic on some problems, and future experiments should be conducted in order to explore whether the change in maximum allowable displacement between matched segments σ or the pairwise error term is more responsible for these differences. At the least, differences observed suggest effort should be put into better understanding how such changes can improve the performance of local search. It can also be expected that the bias introduced into the random selection of initial matches will significantly alter performance, and a future study should systematically test how performance varies as a function of the average number of data line segments initially bound to a model segment.

Alternative heuristics for selecting the subsets used by the subset-convergent algorithm are not explored in this thesis, and future work ought to consider alternatives. For example, the subsets selected automatically for the Hallway model in Chapter 8.3.3 (page 164) are not necessarily the best one could imagine. Two of the subsets, pairs (F, I) and (E, J), consist of pairs of segments which are very close to each other and have the same basic geometric form. One might wonder if having picked two less similar subsets might not have improved performance.

Before it terminates, the tabu search component of the hybrid-weak-full-perspective matching algorithm is allowed to explore up to 10 states without seeing improvement. The choice of 10 states was made because it seemed sufficiently large to preclude premature termination. However, it is quite possible that 10 is needlessly large. Future experiments should profile the algorithm in order to determine if a smaller number might accomplish the same goal with less total computation.

9.4.2 Local Search as Model Directed Feature Grouping

The use of randomly selected initial matches is largely responsible for the robust performance of local search matching. However, this robustness comes with a price. All but the hardest of the matching problems presented in this thesis could probably be solved more quickly if effort were put into carefully selecting 'better than random' initial matches. There are risks in attempting to focus search in this way: if the bias introduced does not suit the specific problem at hand, then the algorithm will not only be slow, it will fail. However, the potential pay-off is a much faster algorithm for many, but not all, problems.

Future work might couple local search matching with key-feature detection, and so initiate local search from hypothesized key-feature matches. The overall approach would resemble that laid out by Lowe [80] in his SCERPO system. Given reliable algorithms for finding key-features, coupling these to local search matching would yield a reliable and fast optimal model matcher. In many practical applications this would be worthwhile. However, this idea still suffers from all the normal problems associated with defining and reliably finding key-features. To date, no general mechanisms have been demonstrated for learning key-features, and therefore this approach assumes someone is willing to invest the time and energy to identify key-features and program key-feature detectors. Alternatively, simple heuristics such as used to identify model subsets in the subset-convergent local search algorithm might suffice in some cases, but these can be expected not to be distinctive under some conditions.

A more intriguing prospect is to use a variant of local search as a uniform control structure in which to blend bottom-up feature grouping and top-down model matching. In the past, bottom-up feature grouping and top-down model matching have been largely separate activities. In concept, the two activities meet gracefully in the middle, with bottom-up grouping stopping and model directed matching taking over. Efforts to realize this goal encounter several problems. First, bottom-up grouping often fails to extract features of sufficient richness to forestall the combinatorial problems of top-down matching. Second, attempts to lessen the combinatorics of matching by extracting more complex bottom-up features are hampered by the combinatorics of grouping, and these explode when more complex groups are sought. Finally, when seeking ever richer bottom-up features, choosing which groups to form out of the combinatorial space of possibilities becomes almost very difficult in the absence of some top-down goal.

A promising way around these problems is to view matching and feature grouping as one uniform activity. Model-directed search would construct ever more complex assemblies of features until ultimately a match is found. A single objective function, such as the match error function defined in this thesis, would direct search throughout this process. Here is a sketch for such an algorithm. Begin with a set of n candidate pairs of model-data segments and use spatial proximity constraints to filter the set of n^2 pairs-of-pairs down to a more manageable set of size $k\,n$, where k is a constant much smaller than n. Treat these $k\,n$ pairs-of-pairs as initial matches and compute the match error for each. Search begins by initiating one cycle of local search for the best of the $k\,n$ matches. In other words, evaluate the Hamming-Distance-1 neighborhood once and update this match accordingly. Typically this will make the best match better. The algorithm proceeds until it can no longer improve this match, and then moves on to the second best match, and so on.

Future work will explore this matching-as-grouping concept. There are several basic issues which require attention. For example, does the process run to some kind of natural completion, or should it be terminated at some point. Checks to avoid redundantly traversing paths to local optima will also be important. Fortunately, this can be done by remembering past matches and the local optima to which they lead. If found again, these matches can be immediately equated with the already identified local optima. Finally, care must be taken to see that the match error guides the initial matching in ways which lead to the detection of the globally optimal match with as little computation as possible.

9.4.3 Match Error and Formal Image Formation Models

There are two quite different ways of designing a match error function. One is to argue directly, as this thesis has done, for a function which quantifies the difference between a perfect match and an observed imperfect match. The other is to formally and fully define a probabilistic error process, and from this process model derive the match error function as either a maximum likelihood estimate or a maximum a-posteriori (MAP) estimate. This thesis asserts that the fit-plus-omission match error is good because: 1) data can fail to fit, 2) data can fail to appear. Future work should augment this with work on formally defined error process models.

This should not be seen as a repudiation of the direct approach taken in this thesis. It is a mistake to assume either approach is fundamentally superior, and it is best to draw insight from both. It is instructive to consider that Gauss, who invented the least-squares method and laid the ground work for maximum likelihood estimation, appears to have preferred the more direct approach. Consider this quote from the paper "Least-squares estimation: from Gauss to Kalman" by Sorenson [100].

> Gauss proceeded by noting that the maximum of the probability density function is determined by maximizing the logarithm of this function. Thus he anticipated the maximum likelihood method, ... It is interesting that Gauss rejected the maximum likelihood method [7] in favor of minimizing some function of the difference between estimate and observation, and thereby recast the least-squares method independent of probability theory.

It is also important to recognize that properly developing a formal error process model is a major undertaking. An error process model should begin with a perfect model instance, or instances, and formally characterize each step leading to the ultimately observable and corrupted image data. It must account for the corruption of model features through skewing, fragmentation, occlusion and outright unexplained omission. It must account for the appearance of unmodeled clutter features, ideally in both random and correlated configurations. Finally, if the resulting numbers are to be meaningfully called likelihoods or probabilities on real problems, then the entire error process model must be validated, including every choice of probability density function and associated parameters.

Precedents for doing this are few and partial. Wells [57] goes part way for point rather than line segment models. He defines probability density functions for observable points in an image. However, he does not specify a complete forward process beginning with a perfect geometric object model and ending with corrupted image data. Recent work for point features by Morgan [86] proposes a complete probabilistic error process model including formal models for noise, omission and clutter.

Error process models for geometric matching may highlight different assumptions underlying different objective functions. For instance, preliminary work on a maximum likelihood interpretation of the fit-plus-omission error suggests it derives from an assumption that all possible clutter configurations are equally likely. This may be seen as justification for why segments not included in a match do not influence the match error. In contrast, Wells's [57] measure appears to derive from an assumption that all possible omissions of model features are equally likely, but differing amounts of clutter appear with differing probability. Consequently, two identical matches score differently in Wells's framework, depending upon the amount of accompanying clutter.

9.4.4 Partial Symmetry and Local Search Matching

Object symmetry has many important ramifications for robotics and computer vision, and significant progress has been made in developing an analytic framework to aid in the development of robotic systems able to reason about symmetry [78, 77]. The performance of random-start local search when matching partially symmetric object models is interesting because local search explicitly finds self-similar mappings between object features. This is evident in the experiments with the Tree model, as shown in Figure 6.6 and 6.7, and is pronounced for the Dandelion model, as shown in Figure 6.10. This tendency to find self-similar mappings between different features on a geometric object model has two important implications.

One implication is that local search is solving a subtle problem: that of detecting self-similar transformations in models. It may not be immediately apparent, but the geometric object models used in this thesis present both minor and major problems for traditional analytic approaches to symmetry. The minor problem is that the models are represented simply as sets of straight line segments, and this brute force and highly general representation is too low-level to support many analytic techniques.

The major problem is that traditional techniques are geared to perfect, not partial, symmetry. The notion that an object is somewhat symmetric raises a host of issues relating to what it means to be somewhat self-similar. It should come as no surprise that the match error presented in this thesis provides a practical definition of self-similarity, and largely solves the definitional problem. The practical problem of discovering feature transformations representing self-similar matches can be solved by matching the model to itself.

Detecting self-similarities with local search matching could be useful in a variety of applications. For example, partial symmetries are important in robot manipulation tasks, and the self-similarities discovered by local search could provide information important to a robot manipulation system. A more immediate application is to use self-similarities as the basis for an augmented local search neighborhood.

In the experiments presented in Chapter 6, local search matching can be described as tolerating the matching problems involving the highly symmetric Dandelion model. It does not exploit the symmetry. In future work, local search matching should be used to discover the feature transformations associated with partial symmetries of models, and these transformations should augment the local neighborhood definition. In this way, local search would move freely between these different matches, and finding one would be equivalent to finding the best. Using this idea, matching problems with highly symmetric object models should become no harder, and perhaps easier, than problems involving non-symmetric object models.

9.4.5 Alternative Optimization Techniques

Having formalized the geometric matching problem as a combinatorial optimization problem, this thesis only explored local search as a way of finding optimal matches. Other noteworthy alternative were not considered, and in the future two in particular ought to be tried: 1) stochastic relaxation, and 2) parallel genetic algorithms. It seems likely local search will turn out to be advantageous under some conditions, while alternatives will be better under other conditions. Recall how Johnson [61], discussed in Section 2.4.2, found no clear winner when comparing simulated annealing and random-starts local search on the graph partitioning problem.

The Hamming-Distance-1 neighborhood local search algorithm can be transformed into a stochastic relaxation algorithm. If, during neighborhood evaluation, the next correspondence mapping selected during search were picked probabilistically based upon the relative increase/decrease in error, then the result would be a stochastic relaxation algorithm. Intuitively, this does not seem a promising option for the simpler problems presented in this thesis; those for which between 5 to 20 trials of local search is sufficient to find the globally optimal match. However, stochastic relaxation might perform better on problems requiring 50, 100 or even 200 trials of local search.

Parallel genetic algorithms typically combine local search and recombination of nearby solutions in order to solve combinatorial optimization problems [29, 87]. It would be useful to see if this is a good way of finding optimal geometric matches. The ideas of matching as model-directed grouping presented earlier defined what might be considered a 'population' of matches. The ideas from the previous section might be further extended using some of the techniques emerging from research on parallel genetic algorithms. For example, matching-as-grouping has removed randomness from the search process, and may make the algorithm less robust. Adding a randomized recombination phase might correct this potential failing.

9.4.6 Full-Perspective Matching

The 3D point-to-plane measure of spatial fit has at least two known problems. This measure was the first tested by Kumar [70, 71], and in subsequent work by Kumar [69] it was found to be inferior to a measure defined in the image plane, and based upon infinitely extended model rather than data line segments. Measuring fit in the image plane treats measurement errors introduced in the image plane uniformly. Recall, as illustrated in Figure 7.2 on page 144, that the point-to-plane measure exaggerates errors for points farther from the image plane. Infinitely extending the model rather than data segments is better because, as discussed in Section 4.3, data segments are often fragmented and skewed.

Future work on local search matching should incorporate Kumar's more recent 3D pose algorithm. A 3D pose algorithm might also be developed which minimizes the integrated-squared-perpendicular-distance (ISPD) measure defined in Section 3.3.1. Unfortunately, these measures preclude the use of the state vector formulation developed in Section 7.4. To compute 3D pose with these measures, weights of terms in the sum-of-squared errors must be modified inside the inner loop of the iterative pose algorithm. This makes the new algorithm computationally unattractive for the algorithms which compute 3D pose for all the n neighbors of a match, but it is a less serious problem for the hybrid-weak-full-perspective algorithm. Using an ISPD-like measure could improve the performance of the hybrid algorithm. In the current version of the algorithm, the 3D pose and weak-perspective measures of fit are not identical: one infinitely extends the data segments while the other infinitely extends the model segments.

Other ways of combining the cheaper weak-perspective and the more expensive full-perspective fitting procedures within a single local search strategy should be investigated. The hybrid-weak-full-perspective algorithm from Chapter 8 is only one of many possible hybrid algorithms. An interesting variation would be to modify the hybrid-weak-full-perspective algorithm so that the match error is computed based upon the updated 3D pose for some number of the most promising neighbors in the Hamming-distance-1 neighborhood. This variant would require more computation than the hybrid-weak-full-perspective algorithm, less than the full-perspective-inertial-descent algorithm, and just might perform better than either.

The full-perspective matching work presented in this thesis is limited by the need for an initial pose estimate. Full-perspective matching algorithms free from this constraint might be developed by combining the matching-as-grouping idea described above with recent work on analytically determining 3D pose for small sets of features. For instance, Horaud's [51] analytic method for 3D pose determination from four sets of matched points does not require an initial pose estimate. It could be used to generate initial pose estimates under certain circumstances and thus provide the estimate needed to initiate full-perspective local search matching. Some insight on how to go about accomplishing this task may be found in the work of Burns [19]. His system does not pursue this exact idea, but it does aggregate features through a sequence of controlled matching steps which provide successively tighter matching constraints.

9.4.7 Model-Based Sensor Fusion

Geometric matching problems arise whenever features on a geometric object model must be simultaneously matched and registered to features extracted from an image. As shown in this thesis, problems of this form arise in CCD imagery. They also arise when attempting to identify modeled objects in other types of imagery, such as 3D LADAR imagery [13] and

SAR imagery [86]. The algorithms developed in this thesis, or variations of them, ought to be useful in these other domains.

Local search matching may also provide the basis for model-directed fusion of features extracted from multiple sensors. The problem, although more involved than matching to features from a single image, is similar in basic form. An optimal correspondence mapping between model features and features from multiple sensors must be found. Optimality will depend upon how well corresponding features fit and account for the object model. Fit will depend upon the estimated pose of the object relative to the different sensors. These pose estimates, in turn, express the registration between images, and hence image registration and pose estimation become inseparable problems. Consequently, local search matching on problems involving multiple sensors will simultaneously search for the optimal correspondence mapping as well as the associated best pose and image registration estimates.

At first, it might appear that problems involving multiple sensors will be harder to solve than those presented in this thesis. Often past efforts have seperated the data from different sensors; for instance using data from one sensor for object indexing and another for verification. Using all the features from multiple sensors simultaneously will tend to increase the combinatorics of the problem. In addition, solving the combined pose and image registration problems essential to match evaluation could prove difficult without initial estimates for both. However, in many domains such estimates will be available, and the additional constraints associated with multiple sensor data might well make local search simpler despite the growth in combinatorial possibilities. Consider, for example, the problem of recognizing a modeled object in approximately registered CCD and LADAR data. These two sensors provide complementary information. While LADAR provides direct but often noisy 3-D surface information, CCD data provides surface boundary information distorted by perspective projection. Combining constraints from each sensor may create distinct and easily found optimal matches in the combinatorial spaces of possibilities. Future work should test this conjecture.

9.5 In Closing

Local search matching is an original contribution to the field of computer vision, and as a consequence of its originality, its performance profile is different from that of the other basic approaches to geometric matching. Unlike most other approaches, it does not rely upon local geometric constraints associated with small subsets of model and image features to direct and constrain search. This, more than anything else, accounts for its robust performance in the presence of highly cluttered and corrupted image data. It also explains the relatively easy and largely unprecedented extension to full-perspective matching.

Weak-perspective matching problems with high quality image data are probably best solved using key-feature, generalized Hough or geometric hashing style algorithms. However, problems involving poor quality data, significant amounts of clutter, occlusion, omission, symmetric models, or full-perspective, should probably be solved using local search matching. Given the unique strengths of local search matching, it promises to play an important role in the future development of ever more capable computer vision systems.

BIBLIOGRAPHY

[1] Ansari, Nirwan and Delp, Edward J. Partial shape recognition: A landmark-based approach. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(5):470 – 483, May 1990.

[2] Arkin, Esther M., Chew, L. Paul, Huttenlocher, Daniel P., Kedem, Klara, and Mitchell, Joseph S. B. An efficiently computable metric for comparing polygonal shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(3):209 – 223, March 1991.

[3] Ayache, N. and Faugeras, O. D. Hyper: A new approach for the recognition and positioning of 2-d objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(1):44 – 54, January 1986.

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

[5] Ballard, Dana H. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111 – 122, 1981.

[6] Ballard, Dana H. and Brown, Christopher M. *Computer Vision.* Prentice-Hall, Inc, Englewood Cliffs, New Jersey, 1982.

[7] Berkson, J. Estimation by least-squares and by maximum likelihood. In *Proc. Third Berkeley Symp. on Mathematical Statistics, and Probability*, volume 1, pages 1 – 11, 1956.

[8] Beveridge, J. Ross. A maximum likelihood view of point and line segment match evaluation. Unpublished draft., 1992.

[9] Beveridge, J. Ross and Riseman, Edward M. Can too much perspective spoil the view? a case study in 2d affine versus 3d perspective model matching. In *Proceedings: Image Understanding Workshop*, pages 665 – 663, San Mateo, CA, January 1992. Morgan Kaufmann.

[10] Beveridge, J. Ross and Riseman, Edward M. Hybrid weak-perspective and full-perspective matching. In *Proceedings: IEEE 1992 Computer Society Conference on Computer Vision and Pattern Recognition*, pages 432 – 438. IEEE Computer Society, June 1992.

[11] Beveridge, J. Ross, Weiss, Rich, and Riseman, Edward M. Optimization of 2-dimensional model matching. In *Proceedings: Image Understanding Workshop*, pages 815 – 830, Los Altos, CA, June 1989. DARPA, Morgan Kaufmann Publishers, Inc (Also a Tech. Report).

[12] Beveridge, J. Ross, Weiss, Rich, and Riseman, Edward M. Combinatorial optimization applied to variable scale 2d model matching. In *Proceedings of the IEEE International Conference on Pattern Recognition 1990, Atlantic City*, pages 18 – 23. IEEE, June 1990.

[13] Bevington, James, Johnston, Randy, Lee, Joel, and Peters, Richard. A modular target recognition algorithm for ladar. In *Proc of the 2nd Automatic Target Recognizer Systems and Technology Conference*, pages 91 – 104, Fort Belvoir, VA, March 1992.

[14] Bolles, R. C. and Cain, R. A. Recognizing and locating partially visible objects: the local-feature-focus method. *International Journal of Robotics Research*, 1(3):57 – 82, 1982.

[15] Bray, Alistair J. Object recognition using local geometric constraints: A robust alternative to tree search. In Faugeras, O., editor, *Proceedings: First European Conference on Computer Vision*, pages 499 – 515, New York, April 1990. Springer-Verlag.

[16] Breuel, Thomas M. An efficient correspondence based algorithm for 2d and 2d model based recognition. A.I. Memo 1259, MIT, MIT AI Lab, October 1990.

[17] Brooks, Rodney A. Symbolic reasoning among 3-d models and 2-d images. *Artificial Intelligence*, 17:285 – 348, 1982.

[18] Brown, Christopher M. Bias and noise in the hough transform 1: Theory. Technical Report TR - 105, Computer Science Department, University of Rochester, June 1982.

[19] Burns, J. B. *Matching 2D Images to Multiple 3D Ojbects using View Description Networks*. PhD thesis, University of Massachusetts, Amherst, MA, February 1992.

[20] Burns, J. B., Hanson, A. R., and Riseman, E. M. Extracting straight lines. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI–8(4):425 – 456, July 1986.

[21] Burns, J. B., Weiss, R., and Riseman, E. R. The non-existence of general-case view-invariants. In Mundy, J. and Zisserman, A., editors, *Geometric Invariance in Computer Vision*, page (to appear). MIT Press, Cambridge, 1992.

[22] Cass, Todd A. Polynomial-time object recognition in the pressence of clutter, occlusion, and uncertainty. In *Proceedings: Image Understanding Workshop*, pages 693 – 704, San Mateo, CA, January 1992. DARPA, Morgan Kaufman.

[23] Chen, C. H. and Kak, A. C. A robot vision system for recognizing 3d objects in low-order polynomial time. *IEEE Trans. on Syst., Man, Cybern.*, 19(6):1535 – 1563, November/December 1989.

[24] Collins, Robert T. and Beveridge, J. Ross. Matching perspective views of coplanar structures using projective unwarping and similarity matcing. In *Proceedings: 1993 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, page (to appear), New York, NY, June 1993.

[25] Costa, Mauro, Haralick, Robert M., and Shapiro, Linda G. Optimal affine - invariant point matching. In *Proceedings of the IEEE International Conference on Pattern Recognition 1990, Atlantic City*, pages 233 – 236. IEEE, June 1990.

[26] Cox, I. J. and Kruskal, J. B. Determining the 2- or 3-dimensional similarity transformation between a point set and a model made of lines and arcs. In *IEEE Proceedings of the 28th Conference on Decision and Control*, page 1167, Tampa, December 1989. IEEE Control Systems Society, IEEE.

[27] Davis, Larry S. Hierarchical generalized hough transforms and line-segment based generalized hough transforms. *Pattern Recognition*, 15(4):277 – 285, 1982.

[28] Davis, Larry S. and Yam, S. A generalized hough-like transformation for shape recognition. Technical Report TR–134, University of Texas, Computer Science, 1980.

[29] Davis, Lawrence, editor. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[30] Draper, Bruce A. *Learning Object Recognition Strategies*. PhD thesis, University of Massachusetts, Amherst, May 1993.

[31] Draper, Bruce A., Hanson, Allen R., and Riseman, Edward M. Learning knowledge-directed visual strategies. In *Proceedings: Image Understanding Workshop*, pages 933 – 940, San Mateo, February 1992. DARPA, Morgan Kaufmann.

[32] Ellis, R. E. A tactile sensing strategy for model-based object recognition. Technical Report COINS TR87-96, University of Massachusetts, Amherst, 1987.

[33] Faugeras, O. D. and Hergert, M. The representation, recognition, and locating of 3d objects. *The International Journal of Robotics Research*, 5(3):27 – 51, 1986.

[34] Fennema, Claude, Hanson, Allen, Riseman, Edward, Beveridge, J. R., and Kumar, R. Model-directed mobile robot navigation. *IEEE Trans. on Syst., Man, Cybern.*, 20(6):1352 – 1369, November/December 1990.

[35] Fischler, Martin A. and Bolles, Robert C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography (reprinted in readings in computer vision, ed. m. a. fischler. *Comm. ACM*, 24(6):381 – 395, June 1981.

[36] Foley, J. D. and Dam, A. Van. *Fundamentals of Interactive Computer Graphics*. The Systems Programming Series. Addison–Wesley, Reading, Massachusetts, 1982.

[37] Gaston, P. C. and Lozano-Pérez, T. Tactile recognition and localization using object models: The case of polyhedra on a plane. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI – 6:721 – 741, May 1984.

[38] Geman, Donald, Geman, Stuart, Graffigne, Christine, and Dong, Ping. Boundary detection by constrained optimization. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12:609 – 628, 1990.

[39] Geman, Stuart and Geman, Donald. Stochastic relaxation, gibbs distribution and the bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI – 6:721 – 741, November 1984.

[40] Glover, F. Tabu search – part i. *ORSA Journal on Computing*, 1(3):190 – 206, 1989.

[41] Gottschalk, Paul G., Turney, Jerry L., and Mudge, Trevor N. Efficient recognition of partially visible objects using a logarithmic complexity matching technique. *International Journal of Robotics Research*, 8(6):110 – 131, December 1989.

[42] Grimson, W. E. L. On the recognition of curved objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(6):632 –643, June 1989.

[43] Grimson, W. E. L. On the recognition of parameterized 2-d objects. *Internation Journal of Computer Vision*, 2(4):353 – 372, April 1989.

[44] Grimson, W. E. L. The combinatorics of object recognition in cluterred environments using constrained search. *Artificial Intelligence*, 44(1):121 – 165, July 1990.

[45] Grimson, W. E. L. and Huttenlocher, D. P. On the sensitivity of the hough tranform for object recognition. In *Proc. of the International Conference on Computer Vision*, pages 700 – 706, 1988.

[46] Grimson, W. E. L. and Lozano-Pérez, T. Localizing overlapping parts by searching the interpretation tree. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9(3):469 –482, 1987.

[47] Grimson, W. Eric L. The effect of indexing on the complexity of object recognition. In *Third International Conference on Computer Vision*, pages 644 – 651. IEEE, IEEE Computer Society Press, December 1990.

[48] Grimson, W. Eric L. *Object Recognition by Computer: The Role of Geometric Constraints*. MIT Press, Cambridge, MA, 1990.

[49] Grimson, W. Eric L. and Huttenlocher, Daniel P. On the sensitivity of geometric hashing. In *Proccedings: ICCV 3*, pages 334 – 338, Osaka Japan, December 1990. IEEE Computer Society, IEEE Computer Society Press.

[50] Holland, John H. *Adaptaion in Natural and Artificial Systems*. University of Michigan Press, 1975.

[51] Horaud, Radu, Conio, Bernard, and Leboulleux, Oliver. An analytic solution for the perspectve 4-point problem. *Computer Vsion, Graphics, and Image Processing*, 47:33 – 44, July 1989.

[52] Horn, B. K. P. Relative orientation. *International Journal of Computer Vision*, 4:59 – 78, 1990.

[53] Huertas, A., Cole, W., and Nevatia, R. Detecting runways in complex airport scenes. *Computer Vsion, Graphics, and Image Processing*, 51(2):107 – 145, August 1990.

[54] Huttenlocher, Daniel P. and Ullman, Shimon. Recognizing solid objects by alignment. In *Proc. of the DARPA Image Understanding Workshop*, pages 1114 – 1124, Cambridge, April 1988. Morgan Kaufman Publishers, Inc., New York.

[55] Huttenlocher, Daniel P. and Ullman, Shimon. Recognizing solid objects by alignment with an image. *International Journal of Computer Vision*, 5(2):195 – 212, November 1990.

[56] Hwang, Vincent S. S. Recognizing and locating partially occluded 2-d objects: Symbolic clustering method. *IEEE Trans. on Syst., Man, Cybern.*, 19(6):1644 – 1656, November 1989.

[57] III, William M. Wells. Map model matching. In *CVPR–91*, pages 486–492, 1991.

[58] Ikeuchi, Katsushi and Hong, Ki Sang. Determining linear shape change: Toward automatic generation of object recognition programs. *Computer Vsion, Graphics, and Image Processing – Image Understanding*, 53(2):154 – 170, March 1991.

[59] Illingworth, J. and Kittler, J. A survey of the hough transform. *Computer Vsion, Graphics, and Image Processing*, 44:87 – 116, 1988.

[60] Johnson, David S., Aragon, Cecila R., McGeoch, Lyle A., and Schevon, Catherine. Optimization by simulated annealing: an experimental evaluation; part i, graph partitioning. *Operations Research*, 37(6):865 – 893, November–December 1989.

[61] Johnson, David S., Aragon, Cecila R., McGeoch, Lyle A., and Schevon, Catherine. Optimization by simulated annealing: an experimental evaluation; part ii, graph coloring and number partitioning. *Operations Research*, 39(3):378 – 406, May–June 1991.

[62] Johnson, David S., Papadimitriou, Christos H., and Yannakakis, Mihalis. How easy is local search. *Journal of Computer and System Sciences*, 37:79 – 100, 1988.

[63] Kalvin, Alan, Schonberg, Edith, Schwartz, Jacob T., and Sharir, Micha. Two-dimensional, model-based, boundary matching using footprints. *The International Journal of Robotics Research*, 5(4):38 – 55, 1986.

[64] Kernighan, B. W. and Lin, S. An efficient heuristic procedure for partitioning graphs. *Bell Systems Tech. Journal*, 49:291 – 307, 1972.

[65] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. Optimization by simulated annelaing. *Science*, 220:671 – 680, 1983.

[66] Knoll, T. F. and Jain, R. C. Recognizing partially visible objects using feature indexed hypotheses. *IEEE Journal of Robotics and Automation*, 2:3 – 13, 1986.

[67] Krentel, Mark W. Structure in locally optimal solutions. In *Proceedings: Symposium on Foundations of Computer Science*, pages 216 – 221. IEEE Computer Society, Computer Society Press, October 1989.

[68] Kumar, Rakesh. Determination of camera location and orientation. In *Proceedings: Image Understanding Workshop*, pages 870 – 881, Los Altos, CA, June 1989. DARPA, Morgan Kaufmann Publishers, Inc.

[69] Kumar, Rakesh. *Model Dependent Inference of 3D Information From a Sequence of 2D Images*. PhD thesis, University of Massachusetts, Amherst, February 1992.

[70] Kumar, Rakesh and Hanson, Allen. Robust estimation of camera location and orientation from noisy data having outliers. In *Proc. of IEEE Workshop on Interpretation of 3D Scenes*, pages 52 – 60, Austin, TX, 1989. IEEE.

[71] Kumar, Rakesh and Hanson, Allen. Analysis of different robust methods for pose refinement. In *Proc. of IEEE Workshop on Robust Methods in Computer Vision*, pages 161 – 182, Seattle, WA, 1990. IEEE.

[72] Lamdan, Y. and Wolfson, H. J. Geometric hashing: A general and efficient model-based recognition scheme. In *Proc. IEEE Second Int. Conf. on Computer Vision*, pages 238 – 249, Tampa, December 1988.

[73] Lamdan, Yehezkel, Schwartz, Jacob T., and Wolfson, Haim J. Affine invariant model-based object recognition. *IEEE Transactions on Robotics and Automation*, 6(5):578 – 589, October 1990.

[74] Li, S. Z. Matching: Invariant to translations, rotations and scale changes. *Pattern Recognition*, 25(6):583 – 594, 1991.

[75] Lin, S. Computer solutions of the traveling salesman problem. *Bell Syst. Comput. J.*, 44:2245 – 2269, 1965.

[76] Lin, S. and Kernighan, B. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498 – 516, 1973.

[77] Liu, Yanxi. *Symmetry Groups in Robotic Assembly Planning*. PhD thesis, University of Massachusetts, Amherst, September 1990.

[78] Liu, Yanxi and Popplestone, Robin. A group theoretic formalization of surface contact. *International Journal of Robotics Research*, page to appear, 1993.

[79] Lowe, David G. Solving for the parameters of object models from image descriptions. In *Proc. ARPA Image Understading Workshop*, pages 121 – 127, 1980.

[80] Lowe, David G. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, 1985.

[81] Lowe, David G. The viewpoint consistency constraint. *Iternational Journal of Computer Vision*, 1(1):58 – 72, 1987.

[82] Lowe, David G. Fitting parameterized three-dimensional models to images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(5):441 – 450, May 1991.

[83] Lowe, David G. and Binford, T. O. The perceptual organization of visual images: Segmentation as a basis for recognition. In *Proc. Image Understanding Workshop, Stanford*, pages 203 – 209, June 1983.

[84] Merman, Martin and Kanade, Takeo. Incremental reconstruction of 3d scenes from multiple, complex images. *A. I. Journal*, 30(3):289 – 341, December 1986.

[85] Mohan, Rakesh and Nevatia, Ramakant. Perceptual grouping for the detection and description of structures in aerial images. In *Proceedings: Image Understanding Workshop – 1988*, pages 512 – 526. DARPA, Morgan Kaufmann, April 1988.

[86] Morgan, Douglas. Point feature detection algorithm explicit state integral version. Technical Memorandum ADS-TR-06327-025-001, Advanced Decisions Systems, September 1992.

[87] Mühlenbein, Heinz. Evolution in time and space – the parallel genetic algorithm. In Rawlins, Gregory J. E., editor, *Foundations of Genetic Algorithms*, pages 316 – 337. Morgan Kaufmann, San Mateo, California, 1991.

[88] Nalwa, Vishvjit S. and Pauchon, Eric. Edgel-aggregation and edge-description. In *Proceedings: Image Understanding Workshop*, pages 176 – 185, Los Altos, CA, 1985. DARPA, Morgan Kaufmann Publishers, Inc.

[89] Nevatia, R. and Babu, R. Linear feature extraction and description. *Computer Vsion, Graphics, and Image Processing*, 13:257 – 269, 1980.

[90] Newton, Tyre A. A simple algorithm for finding eigenvalues and eigenvectors for 2x2 matrices. *The American Mathematical Monthly*, 97(1):57 – 59, January 1990.

[91] Noble, Ben and Daniel, James W. *Applied Linear Algebra.* Prentic-Hall, Inc, Englewood Cliffs, N.J., 2 edition, 1977.

[92] Papadimitriou, Christos H. and Steiglitz, Kenneth. *Combinatorial Optimization: Algorithms and Complexity*, chapter Local Search, pages 454 – 480. Prentice–Hall, Englewood Cliffs, NJ, 1982.

[93] Press, William H., Flannery, Brian P., Teukolsky, Saul A., and Vetterling, William T. *Numerical Recipes in C.* Cambridge University Press, Cambridge, 1988.

[94] Rawlins, Gregory J. E., editor. *Foundations of Genetic Algorithms.* Morgan Kaufmann, San Mateo, California, 1991.

[95] Roberts, L. G. Machine perception of three–dimensional solids. In Tippett, James T., editor, *Optical and Electro-Optical Information Processing*, chapter 9, pages 159 – 197. MIT Press, Cambridge, MA, 1965.

[96] Seigel, Andrew F. and Benson, Richard H. A robust comparison of biological shapes. *Biometric*, 38:341 – 350, June 1982.

[97] Siegel, Andrew F. Geometric data analysis. In Launer, Robert L. and Siegel, Andrew F., editors, *Modern Data Analysis*, pages 110 – 122. Academic Press, Inc, New York, 1982.

[98] Silberberg, T. M., Harwood, D., and Davis, L. S. Object recognition using oriented model points. *Computer Vsion, Graphics, and Image Processing*, 35:47 – 71, 1986.

[99] Sitaraman, R. and Rosenfeld, A. Probabilistic analysis of two stage matching. *Pattern Recognition*, 22(3):331 – 343, 1989.

[100] Sorenson, H. W. Least-squares estimation: from gauss to kalman. *IEEE Spectrum*, pages 63 – 68, July 1970.

[101] Stein, Fridtjof and Medioni, Gérard. Graycode representation and indexing: Fast two dimensional object recognition. In *Proc of 10th ICPR*, page ?, Atlantic City, June 1990.

[102] Stein, Fridtjof and Medioni, Gérard. Recognition of 3-d objects from 2-d groupings. In *Proceedings: Image Understanding Workshop*, pages 667 – 674, San Mateo, January 1992. DARPA, Morgan Kaufmann.

[103] Stockman, George. Object recognition and localization via pose clustering. *Computer Vsion, Graphics, and Image Processing*, 40:361 – 387, 1987.

[104] Swain, M. J. and Ballard, D. H. Indexing via color histograms. In *Third International Conference on Computer Vision*, pages 390 – 393. IEEE, IEEE Computer Society Press, December 1990.

[105] Thompson, D. W. and Mundy, J. Three-dimensional model matching from an unconstrained viewpoint. In *Proc. IEEE International Conference on Robotics and Automation*, pages 208 – 220, 1987.

[106] Tovey, Craig A. Hill climbing with multiple local optima. *SIAM J. Alg. Disc. Meth.*, 6(3):384 – 393, July 1985.

[107] Weiss, Richard and Boldt, Michael. Geometric grouping applied to straight lines. In *CVPR–86 Proceedings*. IEEE Computer Society, June 1986.