

**The Recognition of Two-Dimensional Modeled Objects
in Images**

by

David T. Clemens
B.S., Swarthmore College
(1982)

Submitted in Partial Fulfillment of the
Requirements of the Degree of

MASTER OF SCIENCE
IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 1986

© Massachusetts Institute of Technology, 1986

Signature of Author

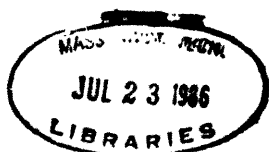
Department of Electrical Engineering and Computer Science
May 21, 1986

Certified by

Professor W. Eric L. Grimson
Thesis Supervisor

Accepted by

Arthur C. Smith
Chairman, Departmental Committee on Graduate Students



Archives

The Recognition of Two-Dimensional Modeled Objects in Images

by

David T. Clemens

Submitted on May 22, 1986 in partial fulfillment
of the requirements for the degree of
Master of Science
in Electrical Engineering and Computer Science

Abstract

Many issues remain unsettled in the field of model-based vision. In this thesis, I identify and explore several conflicting pressures in the two-dimensional subdomain. In particular, the strengths and weaknesses of a common recognition strategy, called *hypothesis-driven* recognition, are investigated. In this strategy, a sequential search through a vast space of image interpretations is avoided by making a commitment to a small set of initial hypotheses derived from the image data. The hypotheses guide further search by allowing predictions to be made, which are then verified. Verification can be much faster than most *data-driven* approaches, but involves certain risks. The investigation consisted of the review of relevant literature and the development of three recognition systems that demonstrate the differences between the hypothesis-driven and data-driven approaches. The implementation required the definition of a feature set and several matching algorithms, in addition to many detailed considerations. The implementation of one of the modules led to a probabilistic interpretation of the Hough transform. The relative strengths and weaknesses of the three systems reveal useful insights for the design of future algorithms.

This thesis contains a discussion of the issues identified in the domain, a description of the recognition systems and an evaluation their performance, and a review of four related systems in terms of the identified issues.

Thesis Supervisor: Prof. W. Eric L. Grimson
Electrical Engineering and Computer Science

Acknowledgments

Thanks first to my advisor, Eric Grimson, for discussing so many ideas with me week after week, for *so* many weeks.

Thanks to proof-readers Margaret Fleck, Eric Saund, and Suzanne Joris who rushed to my rescue as time grew short, and shared the burden of a deadline with me.

Thanks to John Canny, for his cheerful willingness to help with the probabilistic interpretation of pose indexing, his relentless pressure towards formal tactics, and his recognition of the importance of the sheep-from-shearing problem.

Thanks to the AI Lab in general, and Gerry Roynance and Chris Lindblad in particular, for providing the all-important environment of discovery, from Lisp Machines to Revolving Seminars. If it were not for the Lab's existence, I would never have had the opportunity to have conversations with the likes of Jim Mahoney, Bruce Donald, Jon Bliss (thanks for the graphs), Steve Buckley, David Jacobs, Dan Huttenlocher, and many others— I hope they know I thank them, too.

Thanks to Tomas Lozano-Pérez, who helped me get involved with the lab in the first place.

Thanks to my folks, whose attendance at the ceremonies almost got me to finish the thesis on time, but not quite.

And most dearly, thanks to Anne, who has always been supportive, and has helped me live my life happily.

Contents

1	Introduction	6
1.1	A Definition of Model-Based Vision	7
1.2	The Two-Dimensional Domain	8
1.2.1	An Example	8
1.2.2	Aspects of the Domain	9
1.3	Approach and Presentation	12
2	Issues in Machine Vision	14
2.1	Perspectives on Vision	14
2.2	General Challenges in Visual Recognition	16
2.3	Approaches and Conflicting Forces in Recognition	20
2.3.1	Feature Complexity versus Search Algorithm Complexity	20
2.3.2	Accuracy versus Speed: Completeness	22
2.3.3	Hypothesis-Driven Search	25
3	Presentation of Research	28
3.1	Edge Extraction	32
3.1.1	Edges as an Initial Abstraction	32
3.1.2	The Laplacian and Gaussian Operators	33
3.1.3	Zero-Crossing Detection	36
3.2	The Orientation Graph	38
3.2.1	Initial Formation	39
3.2.2	Advantages and Disadvantages of the Orientation Graph	40
3.2.3	Orientation Graph Examples	41
3.3	Feature Extraction	44
3.3.1	Criteria for a Feature	45
3.3.2	Choice of Features and Their Detection	46
3.3.3	Evaluation of Chosen Features	50
3.4	Pose Grouping with an Indexing Array (Hough Transform)	51
3.4.1	A Simple Hough Example	52
3.4.2	Interpreting the Pose Array	54
3.4.3	The Histogram Segmentation Issue	55

3.5	A Probabilistic Interpretation	56
3.5.1	Definition of Terms	56
3.5.2	A Bayesian Flip	57
3.5.3	Getting to the Feature Level	58
3.5.4	A Helical Distribution	60
3.5.5	Error Distributions for Lines and Circles	62
3.6	Pose Array Implementation	63
3.6.1	Using the Discrimination Values	63
3.6.2	Filling the Helical Arc	64
3.6.3	Finding the Peaks in the Histogram	65
3.7	Match Extension by Exploring Feature Neighborhoods	66
3.7.1	The Neck-and-Neck Neighborhood Search Strategy	66
3.8	Grouping Poses	69
3.8.1	Other Grouping Schemes	70
3.9	Predicting Matches from Hypothetical Poses	71
3.10	Results and Discussion	73
3.10.1	Discussion	78
4	A Review of Related Research	82
4.1	Bolles, Cain: Local-Feature-Focus	83
4.2	Grimson, Lozano-Pérez: Tree Pruning with Constraints	86
4.3	Ayeche, Faugeras: HYPER	91
4.4	Turney, Mudge, Voltz: Feature Saliency	95
4.5	Discussion	97
4.5.1	Domain Definition and Dependence	97
4.5.2	Extension to 3D Domains	101
4.5.3	Feature Choice	101
4.5.4	Search Strategy	102
5	Conclusion	105
5.1	Future Work	106

Chapter 1

Introduction

A general-purpose vision system with capabilities similar to those of humans would certainly have applications in many fields. The development of such a system has proved to be a significant challenge. In response, the general vision task has been subdivided along various axes and investigated in limited domains. The goal of research in the domain of model-based recognition is to find methods by which sensory data from the world can be used to identify objects by matching them with object models. Many of the recognition systems that exist today have been developed for specific worlds and to achieve limited goals. Methods have varied, and there does not seem to be a clear path to follow toward developing a system which can be applied to a more general set of situations. In this thesis, I present an overview of some of the issues in the domain of two-dimensional model-based recognition, in which the world is defined to contain only objects with fixed two-dimensional shapes. In particular, I focus on the distinction between hypothesis- and data-driven recognition, and on the advantages and disadvantages of these approaches. Data-driven recognition systems process the image data uniformly, without making commitments to early hypotheses. Hypothesis-driven systems use early hypotheses to limit the search for supporting evidence. My investigation has consisted of the development of three recognition systems demonstrating varying degrees of hypothesis-driven search, and the review of several current systems described by other researchers. In addition to exposing many practical considerations, the process of implementation has led me to find a usable feature set and to develop a probabilistic interpretation of the Hough transform. While all three recognition systems are successful, the central theme of this thesis is the comparison

of the methods and the general insights derived from the work.

1.1 A Definition of Model-Based Vision

A completely general model-based vision system would be quite sophisticated. It would continually sense the moving world around it and keep a three-dimensional map of the positions of identified objects. The objects would be recognized from a vast library of models that covers many classes of objects, including flexible objects, and perhaps even conceptual families, such as "tables", whose members can take on many shapes. The objects would be recognized confidently at any scale or perspective, despite significant occlusion, shadows, reflections, and uneven lighting. And all of this would be done in real time. In this extreme, the only difference between "model-based" vision and general vision is that it explicitly identifies the objects in the image with models, rather than simply determining the presence of unidentified shapes.

In practice, this kind of system is so far from realizable that it would be meaningless to use it to define model-based recognition. It may be an ultimate goal, but current research under the label of "model-based recognition" has much more modest expectations. The boldest of the current algorithms struggle to invert the projection of a static three-dimensional world onto a single two-dimensional image and recover the identity of the 3D object. Model libraries typically contain fewer than ten objects, all rigid and solid. Recognition heavily emphasizes shape; color and texture are typically not used. Disturbances such as shadows and highlights are rarely interpreted, but simply tolerated. Even with these reduced goals, computer recognition is generally slow and far less reliable than human recognition.

Thus, a definition of current model-based recognition could be stated as:

Using models of objects that are found in a particular world, a model-based recognition system interprets sensory data about the world by finding correspondences between the data and the object models, implying the identity of objects in the world.

While this definition is general enough to include the most general vision system, it leaves room for a realistic definition of "world". Recognition systems have been designed

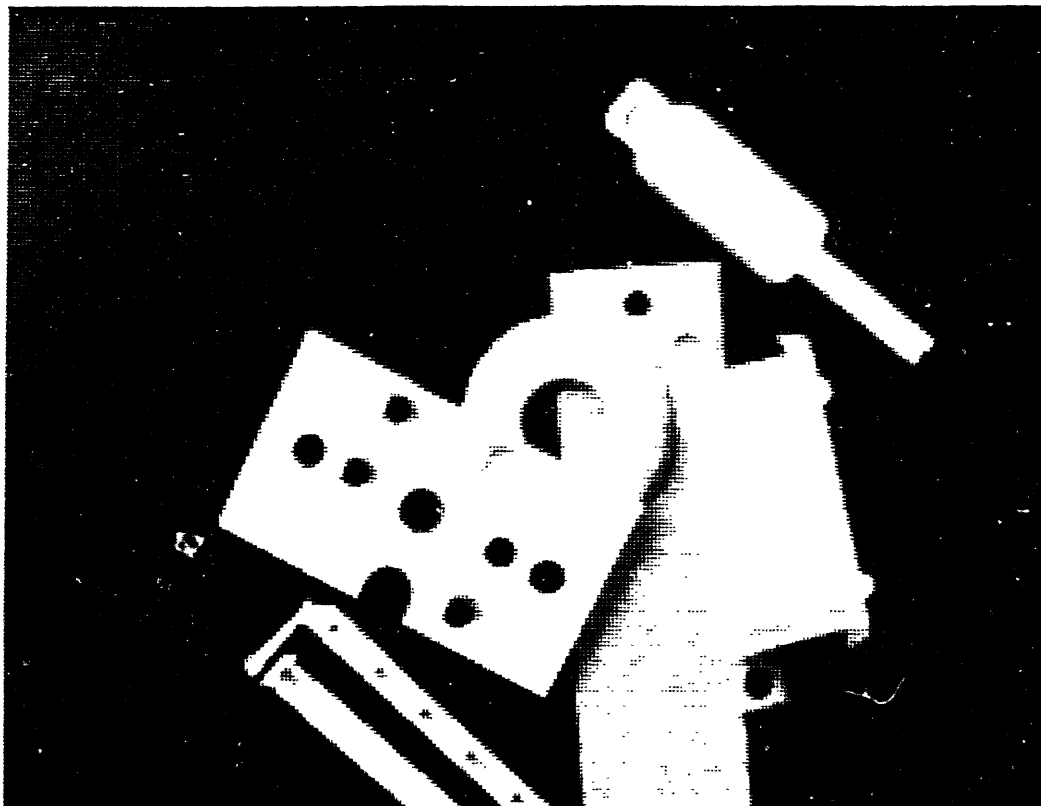


Figure 1.1: A typical image of objects in the two-dimensional domain, (halftoned for photocopy reproduction).

to operate in several different worlds, or domains, each with their own restrictions and assumptions. The domains have usually been defined by a combination of the needs of a specific task, and the assumptions required to develop a working system. In the next section, I describe the general domain used in this thesis.

1.2 The Two-Dimensional Domain

1.2.1 An Example

In the two-dimensional domain, a typical model-based vision system recognizes objects such as those shown in Figure 1.1, using some kind of model of each of the objects, as schematically depicted in Figure 1.2. The system begins with a sensed image of a two-

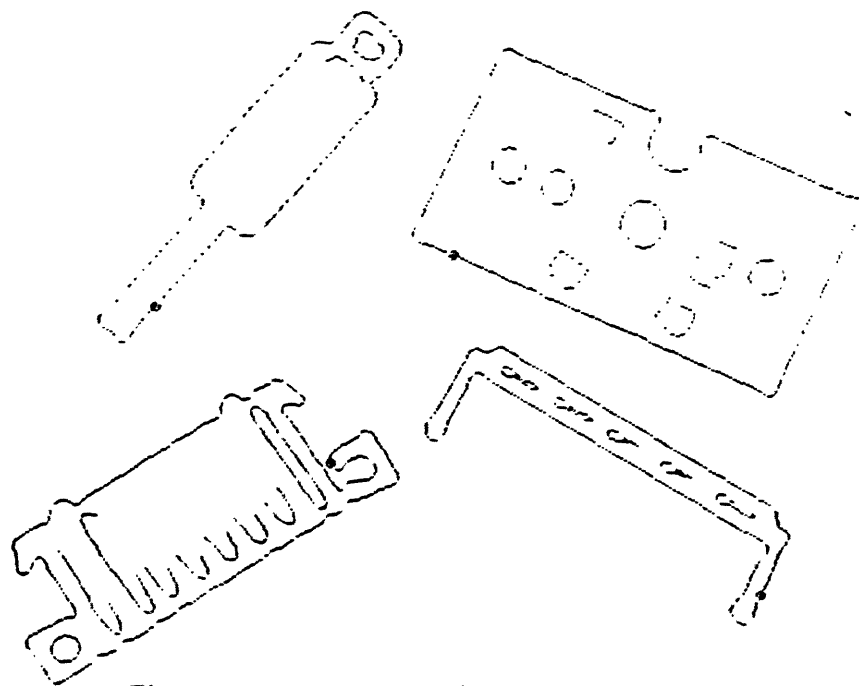


Figure 1.2: Object models (only the shapes are shown).

dimensional world, such as the grey-level image shown, containing none, one, or several of the modeled objects, possibly along with other objects. The system's task is to identify and locate the modeled objects in the image, as in Figure 1.3. This example demonstrates the basic flavor of the problem definition in this domain. A more detailed description of the domain is presented in the following sections.

1.2.2 Aspects of the Domain

Dimension: The most significant aspect of a domain is the dimension of the objects in its world, and the dimension of the sensory information from the world. First, the world is assumed to be static. No systems have yet been developed to recognize objects from image sequences of a world in motion. The world is also limited in the spatial dimensions. In a two-dimensional image of a three-dimensional world, an object can have many shapes, depending on the viewpoint of the camera. A recognition system in this domain must be able to anticipate the affects of projection, despite the fact that information is lost. There is no unique inversion of this projection: an infinite number of object arrangements could have caused the same image. In the two-dimensional domain, the projection problem is removed. Objects are restricted to those that have a small number of resting states when placed on a flat surface.

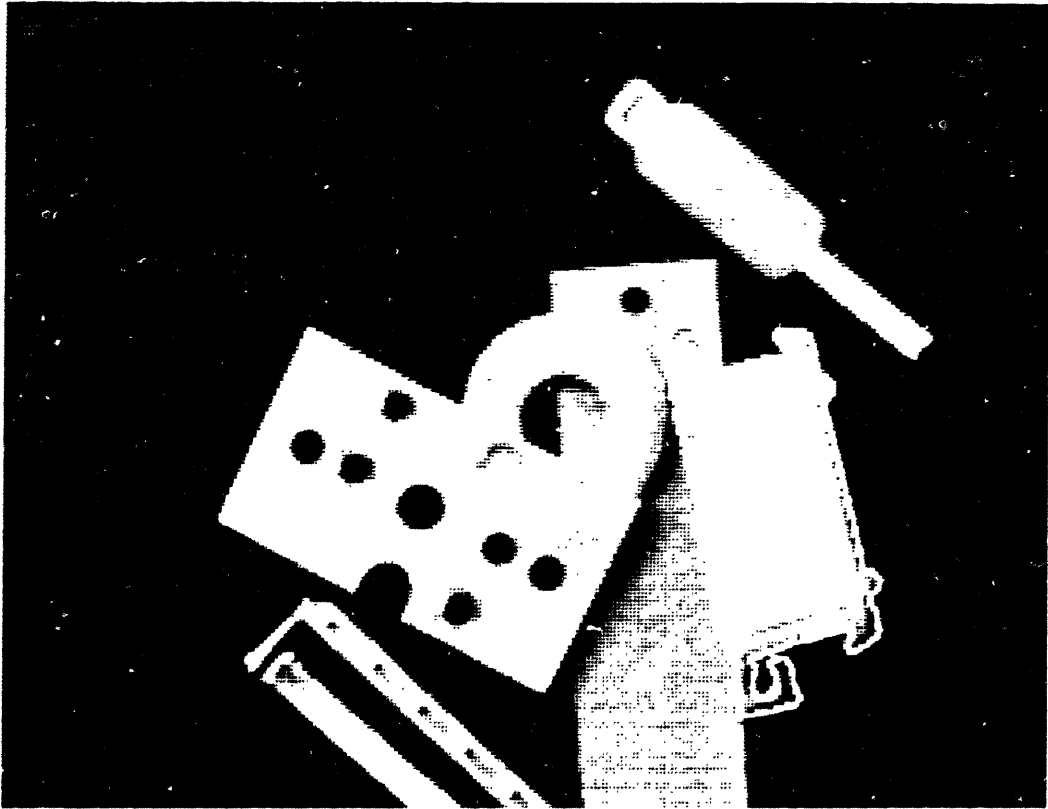


Figure 1.3: The result of recognition.

The major advantage is that when viewed from above, the shape of the outline of the object in the image will be consistent, though it may be in any position and at any rotation. Some systems also allow scale variations. Thus shape alone may be used directly from the image to identify objects from their models. This is a significant restriction.

Object Type: In order to be able to identify an object from the shape of its outline, the objects in this domain must be rigid. They also tend to have smooth boundaries such that a continuous line can describe the outline. This rules out fuzzy or extremely complicated objects. Some systems explore the recognition of objects with one degree of freedom such as scissors, but none have been successful with flexible objects such as cats.

Occlusion: Some early vision systems required that objects be separate from each other in the image, so that simple global measurements of objects, such as the number of holes, could be used for identification. When this condition is relaxed and objects are allowed to touch and overlap, the recognition process becomes much more complicated. The effect of allowing partial covering is that identification must be based on local measurements, and it is not known which parts of an object will be visible. This greatly increases the size of the search for a model match. The inclusion of overlapping objects is one of the ways in which the domain is general. However, any tilting that might occur when objects overlap must be small enough that the shape distortions are not significant.

Image Type: Earlier vision systems often used binary images, in which every pixel is either "on" or "off". A binary image can be produced by backlighting objects or by placing them against a white background and using special lighting to reduce shadows. A brightness threshold is then applied. In such images, the object boundaries are easy to find, and it is clear which parts of the image are background and which are occupied. If the system is extended to recognize objects under natural lighting, a grey-level image is usually needed. Grey-level images simply store a number proportional to the brightness of the scene at each pixel. Other sensing techniques can also be used as input, such as sonar readings of a room, or robot manipulator touch-points.

Model Libraries: Many systems are designed to find only a single model in an image. Some have included several models, but consider each one sequentially. A few systems have incorporated knowledge about relationships among the models in their recognition strategy. These have typically worked with libraries of less than ten objects. The important problems of very large libraries have not been investigated in depth.

The domain I have chosen is two-dimensional, contains rigid objects, has only a single model, but allows occlusion and natural lighting. In this domain, model-based vision consists of finding the best consistent correspondence between parts of the model and parts of the image. I refer to these object parts as *features*. The choice of a useful feature set is an important part of developing a recognition algorithm within a domain. The features typically form the basis of the object model, or *representation*. The structure of the object model and what information it makes easily available is a second important aspect of a recognition algorithm. The *matching scheme* is a third. This is the emphasis of model-based recognition research, in which a correspondence between image and model features is found. The first two aspects are important enough on their own that they have received isolated investigation by other researchers. However, no conclusive discoveries have been made and thus feature definition and object representation are still important unknowns. Because the implementation decisions made at the earlier levels have such an effect on the matching scheme, model-based recognition research often includes significant work at all levels.

1.3 Approach and Presentation

My approach has been to work within the limited domain described above, with the goal of discovering insights that will apply to many algorithms and possibly can be generalized beyond the domain. The investigation has included the development of various algorithms, in which several types of features and several matching strategies were tried. Through the implementation and from investigating related algorithms, the need became clear for a uniform perspective which could provide a more general understanding of the trade-offs and concepts in the field. In Chapter 2, I present some of the issues relevant to model-based vision. One of the identified trade-offs involves the matching scheme. Because the set of all possible correspondences between image features and model features is large, recognition

algorithms are designed to avoid trying every mapping. Some algorithms simply use the data in a uniform manner to produce the most likely hypotheses (by some criteria). Many approach the problem by developing early partial hypotheses about feature matches or the position and orientation (pose) of the model in the image. These initial hypotheses are then used to guide the search for completing the hypotheses. These two strategies led me to develop three recognition systems. The first of these systems is based on the Hough transform. In order to make this method work, several alterations were needed, which were guided by a probabilistic investigation of the method. The second uses a hypotheses-directed local search for features, but then combines these feature groups in a data-driven module. The third system uses the same feature groups as seeds in a classic hypothesis-driven approach to predict the location of all model features in the image. The systems and related details are presented in Chapter 3. In Chapter 4, reviews of four related algorithms by other researchers are presented and discussed in terms of the issues introduced in the earlier chapters. Finally, Chapter 5 summarizes the major points of the thesis.

Chapter 2

Issues in Machine Vision

The field of Machine Vision is still in its infancy, and as such, even the determination of the issues is itself an issue. It is currently a field of construction by trial and error, rather than a science based on discoveries of natural truths. Results are often in the form of a partially successful heuristic algorithm for a limited domain, rather than conclusive research which can serve as a building block towards clear higher goals. I believe that this is a natural phenomenon for this kind of field, but I also believe that there is much to be gained by attempting to develop a broader perspective. In addition to building limited vision systems for specific tasks, it is useful to try to form generic models of processes, to categorize approaches, and identify common trade-offs. In this chapter, I outline my interpretation of where model-based recognition fits into general vision research, describe some of the major challenges in vision research, and then present the issues within my domain which are the focus of this thesis.

2.1 Perspectives on Vision

Because the problem of developing a vision system with human-like capabilities is overwhelming, it is typically attacked by trying to divide the problem into manageable parts. The task of determining a natural division is very difficult, since it requires a clear understanding of the operation of the whole system. Furthermore, the problem may not be subdividable. That is, it may be that the interaction among functions which seem conceptually distinct is, in fact, so extensive that none of the parts can be usefully considered

independently of the whole. If this is so, it will be very difficult to develop any of the parts successfully without significant global understanding, since the inputs from all other parts cannot be determined. (Marr [76] discusses two types of theoretical approaches in consideration of this problem).

Despite this possibility, the investigation of vision has been partitioned in many ways. Neurobiologists have tried to understand vision by investigating the only successful examples in existence: biological systems. Due to the complexity of animal neural systems, this work is mostly limited to early vision: the processing which occurs on the retina or immediately thereafter. Psychological experiments have also been performed to understand how biological examples work, but through external probing. This work can help determine what human vision systems do, and hence indirectly support or discourage hypotheses about how it is accomplished, but is limited to external stimuli and measurements. Computational approaches try to develop executable algorithms to perform the same (or related) functions as biological vision systems, but usually without explicitly imitating the mechanism. These are limited by the state of technology, and more importantly, by the absolute demand for explicit descriptions of any proposed mechanism. This demand can be an asset, since it automatically limits the investigation to viable solutions. However, it is also responsible for diverting attention from general approach questions to detailed algorithmic questions, and for forcing vision systems into limited domains.

Within the computational approach, vision is often categorized by level. Levels are ordered by the sequence in which it is assumed that they are performed in the human visual system. It seems clear that the first processing done on images is at the retinal level. Generally, it is assumed that this earliest vision is very local, and therefore performs tasks such as smoothing and edge-detection. Intermediate levels might find locations in the image on which to focus attention [Ullman 83], find related portions of the image, find features of some type, or parse the image into separate objects. Higher levels recognize objects in the image, find navigable paths, inspect parts for defects, and so on. Ultimately, higher levels of perception merge with cognitive processes, and require an understanding of the representation of knowledge. It should not be assumed that information is only passed up the hierarchy— more likely, information flows in both directions, allowing knowledge about objects to help parse the image, for example.

Another perspective divides computational vision into categories by the type of clue

that can be found in visual data to determine the shapes of objects in the world. One can find shape from stereo images, for instance, and determine the depth of objects by finding corresponding points in the two images. Adding another dimension to the problem, in finding shape from motion, a continuous sequence of images from changing perspectives provides a quite different kind of information about the world. Shape is also implied by shading. By inspecting smoothly varying intensity levels in a single image, with some assumptions about the surface and the light source, one can infer continuous curvatures. These modules are often referred to as “shape from” routines. They require dealing with all levels of vision, concentrating on the middle- to higher- levels. It seems that natural vision systems accomplish all of these tasks, though not necessarily in distinct modules. Limited success has been achieved in several of these categories, but only recently has any attempt been made to combine the parts. It is also not clear that detailed shape or a depth map for the entire image is a necessary intermediate representation of the world. Higher-level modules may never need to know the distances of every part of every object from the sensor in order to accomplish their tasks.

While it is not known which, if any, of these divisions of the vision problem are optimal, they are not arbitrary. By using them to guide the attack, the understanding of vision is allowed to proceed.

2.2 General Challenges in Visual Recognition

Because humans process visual information so well, and subconsciously, it is easy for us to take it for granted, and difficult to realize what might be hard for a machine to do. The problem is distinctive in the large amount of information that must be processed, and in the surprising fact that the information is very poorly structured. There is no explicit association of parts of an image with particular objects, and no explicit representation of depth. The image intensities do not even directly indicate the color of the object, since that also depends on the lighting and angle of viewing. From an array of reflected intensities, the shapes, identities and locations of the objects in the world must be deduced. The problem is not even well posed, since the projection of three-dimensional data onto a two-dimensional retina is not invertible. Many possible worlds could theoretically form the same image, but from our knowledge of the kind of shapes that are in the world (and stereo vision, and

motion) we are able to confidently infer the most likely world that could cause the image. This is truly an outstanding accomplishment of evolution. The magnitude of the challenge to develop vision systems should not be underestimated.

In this section, I briefly describe several of the general hurdles faced by model-based recognition. The aspects of recognition domains described in Section 1.2 highlight many of the difficulties presented in more detail below. In the next section, I focus on three issues in the two-dimensional domain which motivated the development of the algorithms presented in Chapter 3.

Edge Detection

At the lowest level, the intensity data can be misleading in several ways. The human vision system is remarkably good at “subtracting out” the effects of uneven lighting, reflections, shadows, and other effects which make the image intensity inaccurately indicate the color (greyness) of objects, and the locations of object boundaries. Early systems used backlighting through a diffusive material, so that the outlines of objects were very clear. Features in the middle of objects were made invisible, however, and the method was obviously restricted in its application. The detection of edges in an image has received much attention, and many different algorithms have been proposed. But none seem to be able to find all the object boundaries and markings that our human vision system can find, at least without finding a large number of less important intensity fluxuations in addition. When too many are found it is very difficult to rank the importance of the edges so that higher levels can make use of them. Thus, even the lowest-level modules remain elusive.

Features and Representation

At the middle level, a very fundamental problem is exactly how to capture the essence of what makes one object different from another, and how to explicitly represent this information. Because of occlusion, simple global parameters are not acceptable, so local features are usually used. The global shape of the object is important, however, so the features must be kept in a structure which maintains global geometric references. Whatever features are used, it must be possible to extract them repeatably despite rotations and translations of the object. In a three-dimensional world, there is usually no part of an object that does not cause a different intensity pattern to appear in an image as the object is rotated. There is clearly a need to abstract the data to a more invariant form than intensities. This ab-

straction must be able to circumvent the effects of perspective and occlusion, as well as whatever weaknesses the edge-finding process may have.

The feature extraction process must accomplish a great deal, and hence features are not typically easy to extract. Even in the two-dimensional world it is difficult to determine which features would be useful to extract. They must capture whatever information distinguishes an object from all others, and yet must be general enough that the same set of features can be used in different arrangements to represent a broad class of objects. They must be able to be extracted without any knowledge of the identity of the object, as they must be found in the image before the object is matched with any model. Section 3.3 presents a more detailed description of the requirements of a feature.

Matching and Occlusion

Once the features are extracted, they must be matched with the correct model. The first problem that all algorithms must confront is the sheer size of the search space. It seems unlikely that every possible mapping of the model into the image can be tried sequentially. For example, if there are one hundred image features and only a single model with 20 features, then there are $\binom{100}{20} = 5.36 \times 10^{20}$ complete mappings (allowing only one-to-one mappings). To check a single possibility requires checking each of the 20 feature matches in the mapping. Many of the mappings will be completely wrong, and it may be possible to rule them out after checking only a few matches. Note, however, that individual matches may be perfectly acceptable, and only the combination of particular matches can rule out a mapping. More progress can be made by organizing the mappings by families, or in a tree (as Grimson, Lozano-Pérez have done [84, 85]), and pruning entire branches of mappings at once by checking a single match.

The job of reducing the search space is further complicated by the introduction of occlusion. By allowing the possibility that a model feature may not be visible, a few bad matches can no longer be taken as evidence that the rest of the mapping is incorrect. One way to account for occlusion is to add a null feature to the set of image features, and allow matching to it in the same manner as if it were any other image feature. But this addition is not just a slight increment of the size of the search, since matching to the null face is always possible, whereas matching to a particular image feature can usually be ruled out fairly easily. The significance of this can be seen in a matching tree. Consider a tree in

which each level represents the assignment of a particular model feature to one of the image features. If every match but one could be eliminated at each level (an extreme example), then the tree could be traced down to a single leaf. If, instead, the possibility of occlusion existed for n of the model features, there would be 2^n possible leaves.

Pose Determination

While the mapping of the model features into the set of image features is important to determine, and is the basis of recognition, another part of the hypothesis is the position and orientation (pose) of the model in the image. By making explicit the pose in a hypothesis, separate local feature matches can be evaluated in terms of the global hypothesis without requiring checks against all other matches in the feature mapping. The pose can act as a focus for a hypothetical mapping as it evolves. Some recognition algorithms emphasize the pose instead of the mapping. For example, since each match makes some restrictions on the pose, one can search for poses which are consistent with the most matches. Again, the size of the space of all poses can be large (and is not discrete), and a sequential search is probably not appropriate. There may be some way to direct the search to the most likely pose, but it will not be simple hill-climbing. A model feature may have many peaks of good matches in the space of all poses, and a function indicating the "goodness" of the combination of the matches of all the features in a model will be very complex, with many local maxima. It is also not clearly defined, and most definitions are time-consuming to evaluate.

Image Reasoning

Even viewing the problem as a pose search does not directly express other aspects of recognition. Recognition may also require limited reasoning about the image, probably before object models are consulted. One of the reasons that the mapping search space is so large is that separate objects are not identified, so all possible combinations of features from two objects are tested despite the fact that a single model cannot be the cause of features in two objects at once. If the recognition algorithm could partially parse the image, the combinatorics of matching would be drastically reduced. In addition, when accounting for occlusion, it should be possible to include hypotheses about which object is causing the occlusion. Random omissions of model features in the image are much less acceptable than consecutive ones, since consecutive occlusions could all be caused by a single object. If

that object is identified, the extent of the occlusion could be predicted, allowing omissions beyond the object to be taken as evidence against the hypothesis. Image reasoning has not yet been developed to a great extent, and work within the limited domains has been able to be partially successful without appreciable reasoning. In broader domains, however, I believe limited image reasoning will be a very important part of recognition.

Model Libraries

A further complication is introduced when a large library of models is to be used. Multiple models introduce the possibility that any particular model is not anywhere in the image, just as occlusion introduces the possibility that a model feature is not visible in the image. As with null feature matching, null object matching is always more difficult to rule out than a match with a specific model. In addition, a sequential search again seems infeasible if the library is large enough. Models might be arranged in trees, so that certain evidence could rule out entire classes of objects, but such a tree is difficult to actually organize. There does not seem to be anything fundamental enough about an object to place it in any clear position in a tree. More likely, a graph arrangement would reflect similarities of many types among the models. The elimination of every model but one is still a large problem. At some point, it would probably be necessary to consider many models individually, and determine exactly what to look for in the image in order to confirm or discard each model. This notion of verification is considered in greater detail below, in section 2.3.3.

2.3 Approaches and Conflicting Forces in Recognition

The previous sections have introduced several factors to consider in the design of a model-based recognition system. In this section, three discernible trade-offs are identified and discussed. These issues in particular are central to this thesis and recur throughout. They are the motivating factors behind the development of the algorithms presented in Chapter 3.

2.3.1 Feature Complexity versus Search Algorithm Complexity

I have purposefully left the definition of a feature vague, because it can actually be any abstraction of the lowest-level information in the image. In order to be useful, a feature

must have certain qualities, but one aspect of the feature set in particular pertains to the overall algorithm strategy, and that is its complexity. In an extreme example, one could consider the identification of a portion of the image with an object to be a feature. That is, the entire complexity of the recognition algorithm could be pushed down into the feature extraction module. In the other extreme, consider a set of features which are very simple, such as a list of the edge pixels, which do not make explicit much of the important information of the image. Not only will there be a greater quantity of data to process, but it will not be in a form which is easy to reduce, since the data is less closely related to the higher level concepts of hypotheses.

Along the continuum between the drastic extremes, the more complicated the features, the more time will typically be spent on their extraction. The extraction process is typically performed without reference to model data, though I believe a greater exploration of the flow of information from higher to lower levels would be worthwhile. Thus it seems that too much computation during feature extraction may be wasteful, since the additional model information available to the matching module should be able to eliminate some of the work. However, if the information available in a feature is too sparse, it may needlessly increase the work of the matching module. For example, consider a feature which identifies a distinctive point along the edge of an object. If only the point is given, the image feature could be matched to any model feature. The elimination of matches could only come from pairing the match with other matches, since that is the only way shape constraints could be applied. Simply including an indication of the type of the feature reduces the number of possible matches, but the type must be reproducible. If an indication of the tangent angle at the point were included, an (error-prone) pose hypothesis could be made from a single match, allowing individual elimination of the match with respect to any other pose hypothesis. If a further parameter is calculated, such as the rate of curvature or some other local indication of shape, matches could be evaluated in complete isolation, comparing only the feature parameters, since they are independent of any pose hypothesis.

How much more information will be useful? If a new set of features is proposed, where each new feature is a pair of old features, matches could be accurately and effectively pruned using the multiple pose-independent degrees of freedom. If old features are paired by adjacency along the contour, for example, a new problem is introduced. If an old feature is omitted in the image, or if the order of two close features is reversed, three of

the new features will not be preserved. Thus, the new feature set is less repeatable. If this weakness is to be accounted for rather than ignored, the matching module will have to spend additional time undoing the pairing and trying likely recombinations. Or, in order to assure that the pairings of the old features were the same as in the model, all possible pairings might be used. Suddenly, instead of perhaps 100 old image features, there would be $\binom{100}{2} = 4950$ new image features to match with the new model features. Increased feature complexity does not usually imply an increased quantity of features, but at some level, in some form, increased feature complexity will exceed its utility.

2.3.2 Accuracy versus Speed: Completeness

As mentioned in Chapter 1, the design of existing recognition algorithms has depended heavily on the application; there are many sub-domains, each with a different definition of recognition. In particular, the required accuracy of the pose and even the required accuracy of the identity of recognized objects varies from task to task. Some algorithms have been able improve execution times by achieving more modest goals. Others have taken shortcuts without acknowledging a reduction in accuracy. A common place for making over-zealous reductions is in the matching module, where the search space must be drastically truncated in some manner, but the best hypothesis must be preserved. Unless care is taken, parts of the supporting evidence for the correct hypothesis can be lost, reducing the probability of finding the best result.

In particular, I believe the *completeness* of a search is an important consideration to introduce in evaluating the accuracy of a recognition algorithm. I define completeness relative to an *acceptance criterion*. A search through the set of all possible mappings of the model features into the image features is complete if it can guarantee to have found all the mappings which satisfy the acceptance criterion. A separate issue is how many extraneous mappings are also found, but if none of the acceptable mappings are discarded, the search is complete. In the interest of speeding up a search, a severe acceptance requirement may be used, so that only very good mapping hypotheses are found. Typically, however, there is a practical limit on how severe the criterion can be. This is due in part to the presence of distortions and noise in the image and approximations and errors in the feature extraction process. Mostly, the severity is limited by the ability to define an accurate and efficient

acceptance criterion. Many versions have been tried, but it is difficult to determine a test which can be easily applied to a mapping, or more importantly to a partial mapping, that will indicate whether the modeled object could have caused the image features to appear, especially allowing for occlusion and arbitrary lighting.

Another restriction on the acceptance criterion is that in some images, the best interpretation will be weak, while in others it will be strong. Thus, a fixed criterion must accept weak interpretations in case the object is highly occluded, for example. Since processing time has a tendency to increase with the number of accepted interpretations, this can slow down the search. A common response to this problem is to have a floating acceptance criterion, which adjusts its severity to the clarity of the image. Rarely is any *a priori* estimation of the image quality made. Instead, the criterion is adjusted dynamically during the search, using the "best" interpretation that the search has produced so far to set the severity of the criterion for the rest of the search. This approach still results in a complete search, and can potentially cause drastic reductions in the total search time. However, it runs a significant risk of increasing the severity too much, and thus counting too heavily on the correctness of the acceptance criterion. For example, consider using the total length of the modeled object's boundary found in the image to be a heuristic measure of the quality of the interpretation. A simple acceptance criterion would be a threshold on the matched length. Start the threshold at zero length, then reset it to largest value obtained so far. It is quite possible for an erroneous interpretation to be found early with length slightly greater than the correct interpretation. In this case, the correct mapping would be lost. This approach fails because the length is not a perfect indicator of the correctness of an interpretation, though it is relatively easy to implement. Typically, a much more rigorous verification of hypothetical interpretations must be used in a later stage to discriminate among many reasonably good hypotheses. This verification process is usually much too slow to use as the acceptance criterion in the initial search, so sufficient error margins in the severity of less accurate criteria must be included.

Using a floating acceptance criterion with sufficient severity restraint, the search is commonly further improved by organizing it in a "best"-first order. Again, the definition of "best" is quite restricted. It is difficult to make arbitrary changes in the order of the search, so typically only very weak estimates can be used. It is clear that a perfect definition of "best" cannot be found, since the entire purpose of the search is to find the best

interpretation. A better way of expressing the ordering is “most-likely”-first. This can be mathematically defined using probability, but I do not know of anyone who has found it worth determining. The goal of the ordering is simply to find the best interpretation earlier, so that the acceptance criterion can be tightened for a greater portion of the search.

Even with a severe criterion, fixed or floating, a complete search takes a long time. A common tendency is to count heavily on “best”-first ordering, and simply stop the search as soon as the first interpretation above a pre-defined quality threshold is found. This could be interpreted as a complete search with an infinitely severe acceptance criterion, but I will call it *incomplete*. Incomplete searches can be very fast, and can even have acceptable failure rates for some applications. But there is no guarantee that the best interpretation is found by any criterion, unless the best-first ordering can be shown to be strictly correct, in which case the search would be unnecessary. Several recognition systems rely on this method, but few seem to make a distinction between this kind of potential error, due to the search being ordered by a heuristic measure, from error due to inaccuracies of the acceptance criterion.

Completeness and the severity of the acceptance criterion can be even more important in situations where it is possible that either multiple or no instances of the modeled object will be visible. In both cases, finding the “best” mapping is no longer sufficient. Each mapping must be evaluated on its own, rather than relative to other mappings. All initial hypotheses must be considered, whether or not a good interpretation has been found. Therefore, the criterion for acceptance cannot float, and must be accurate enough to both exclude close but incorrect interpretations and include very weak correct interpretations. The major difference from finding a single model is that the recognition module is forced to prove that all hypotheses which are not clearly correct are certainly incorrect, rather than simply proving that they are worse than the best hypothesis. Even a very weak initial hypothesis could possibly be grown into a correct interpretation of the image. Proving that it cannot requires a much deeper investigation of the vast majority of initial hypotheses. The increased accuracy of the criterion, the requirement that it be fixed, and the need for a complete search can make an otherwise fast algorithm very slow, or possibly infeasible.

The incomplete search only fails by its willingness to commit to an early hypothesis above some threshold of quality measure. Ultimately, all recognition methods suffer from a dependence on a quality measure, but the measure is allowed to be arbitrarily accurate. The greater accuracy is achieved through verification, as mentioned above, and need not require

a complicated acceptance criterion during the search. Alternatively, this can be viewed as a multi-stage acceptance criterion, where quick but relatively conservative tests are applied during most of the search, and more complicated tests are reserved for interpretations which pass the simple tests. Thus, with slight modification, the incomplete strategy can be converted into a search technique which is still incomplete, but no more prone to failure than complete searches. As each sufficiently good hypothesis is found, rather than simply terminating the search, the hypothesis is verified extensively. If it fails on close inspection, the search is continued. It may be that a better very good interpretation will be missed, but if the verification process is good enough, the earlier interpretation will simply be one of several acceptable interpretations. The only difference is that a complete search can be used to find all the acceptable interpretations in order by a quality measure, whereas the incomplete search will still only find the first. With this modification, incomplete searches are also similar to complete searches in execution time. The identification of these recognition search properties is, however, an important step in the understanding and development of future recognition systems.

2.3.3 Hypothesis-Driven Search

At the end of many recognition algorithms, or during the search stage as described in the previous section, a verification routine is employed, in which a strong hypothesis is extensively checked by looking for supporting evidence in the image. This is the test stage of the hypothesize-and-test paradigm. In the previous section, verification is presented as merely the application of a more complete acceptance criterion. While it is possible to simply use more stringent tests on the matches already made, verification usually involves finding new evidence for the hypothesis. Thus, verification and normal fixed-structure search differ in a more important way than just precision. During verification, the development of the hypothetical interpretation of the image is driven by the current hypothesis. Together with the model, the hypothesis is used to predict what the image should hypothetically contain. I call this kind of search *hypothesis-driven*. In a systems perspective, hypothesis-driven search is something like a feedback loop. It uses the output of the first recognition stages to form a prediction of the input, and then responds to the error between the predicted input and the actual image. Verification usually does not involve search. The test knows what

to look for, and simply checks to determine if it is there. With the search space being such an important problem in recognition, why is verification not used at an earlier stage? The most common reason is that it is expensive in its own way, and at earlier stages there are too many hypotheses to check. However, if the concept behind verification is generalized a little, hypothesis-driven search may be taken as a feasible approach to recognition.

The distinction between hypothesis-driven and straight search methods is not always clear when trying to categorize actual implementations. One way of distinguishing verification is that it drives the processing of image data from hypotheses, rather than simply crunching through the image data and reporting what hypotheses are left. (For this reason, normal search techniques will be called *data-driven*). Hypothesis-driven search attempts to find evidence to *develop* a hypothesis, rather than trying to *eliminate* all of the incorrect hypotheses. Naturally, hypothesis-driven recognition must start from somewhere— it must have some seeds, and the best hypothesis must not be excluded from the seeds. Thus, there is a trade-off between hypothesis- and data-driven recognition. How far in the elimination process should one go before switching to development? How can one be sure that the best hypothesis has not been excluded? In order to be sure, is it necessary to start with such a large number of seeds that the process is not, in fact, more efficient?

This issue is closely related to the accuracy versus speed issue presented in the previous section. Data-driven recognition techniques are often complete, since they make no commitments to early hypotheses. Hypothesis-driven recognition can be complete, but often is not since it is difficult to guarantee that the partial hypotheses which are used as seeds will always include the best final hypothesis. This is because a partial hypothesis typically does not contain enough information to evaluate it accurately. To be complete, hypothesis-driven recognition must attempt to develop all of the hypotheses which are not eliminated by the initial data-driven stage.

One advantage of verification is that it is free to use a different set of features than those which are used to make initial hypotheses. In fact, it is free to use any form of comparison between the predicted and actual images, even forms which are quite removed from the concept of features. For example, the shape of the areas in the image which are occupied, based on some distinction between background and objects, can be used to rule out hypothetical transforms which place the model in unoccupied regions. In this case, the “features” would be some encoding of the occupied areas, but the notion of features

is not really needed. Another possibility is to place the model features in the image and then check the image to see if the model features could be formed. The checking procedure can be more liberal than the formation procedure, since it will not cause a combinatoric explosion of possible matches. Or, a more feature-to-feature kind of comparison can be made, but the criterion for choosing the comparison features may be different.

There can be some ambiguity in the distinction between the two approaches. It is not always possible to determine whether an algorithm is implicitly eliminating hypotheses through explicitly developing others, or vice versa. The concept of verification is not as concrete as it may seem at first, but it does provide another perspective on the problem-one which can lead to concrete changes in an algorithm.

This thesis explores this particular issue through the development of three algorithms which use identical features to facilitate comparison, but separate in their approach to hypothesis formation. The first algorithm does a large amount of feature processing before ever determining any hypotheses. The second grows hypotheses from the set of all initial feature matches, using the connectedness of the contours to guide the formation. The longest grown hypotheses are then collected in a data-driven module. The third algorithm begins with the extended contour matches of the second algorithm, but then continues in a hypothesis-driven approach using the matches to predict the locations of further matches in the image. The details are presented in the next chapter.

Chapter 3

Presentation of Research

To develop and confront the issues presented in Chapter 2, I have implemented several recognition algorithms. I present the latest two versions in this chapter. The process of implementation has had many stages, and has forced me to find workable solutions amidst conflicting criteria. Implementation has consisted of finding:

- A subdomain of the general vision problem, as described in Chapter 1, which restricts the dimension of the viewed world and the types of objects in it.
- The form of the information to be extracted from images, and a set of features to further abstract the data.
- A representation for objects in the subdomain, which uses the features in a structure.
- A method for matching parts of the image representation with the model representation.

One of the contributions of this thesis is simply the development of the two recognition systems. Parts of the systems are new, while other parts are based on well-known approaches. In particular, I have developed a probabilistic interpretation of the Hough transform [Hough 62], and have implemented several improvements to the method based on the interpretation.

Another goal of this thesis has been to explore hypothesis-driven recognition (described in Section 2.3.3). To this end, the recognition algorithms are based on the same feature extraction module and model representation, to facilitate comparison, but differ in the way

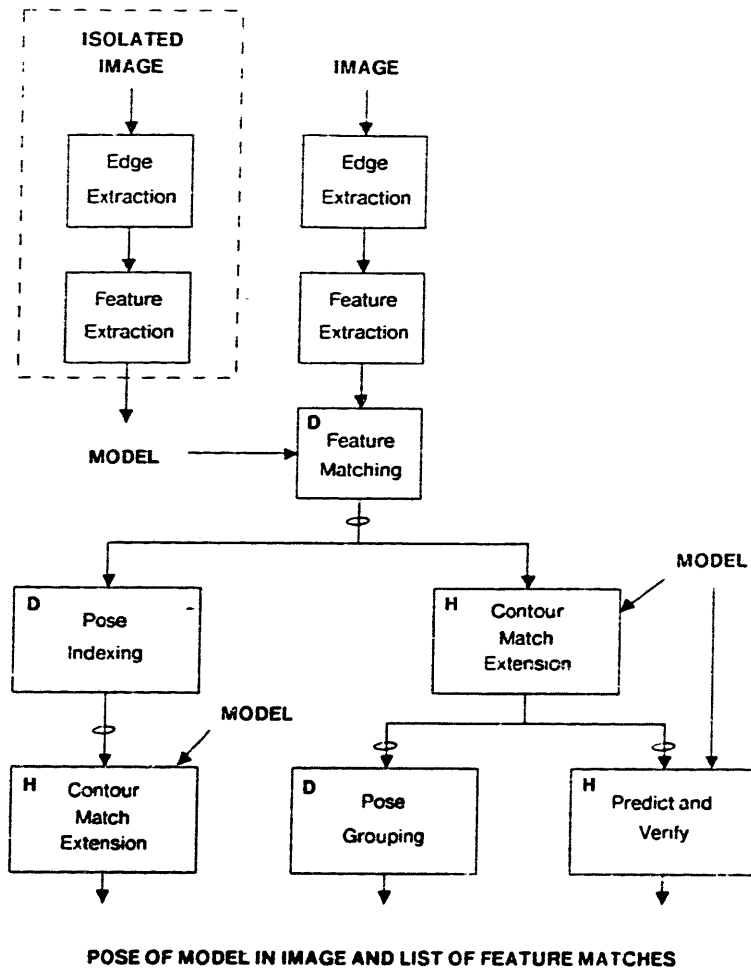


Figure 3.1: Modules of the recognition algorithms. Left: Data-driven. Middle: A combination. Right: Hypothesis-driven

that they identify the model in the image. The first algorithm is driven primarily by the image data, while the third directs the search dynamically, using intermediate hypotheses. The second is a combination of both strategies. The algorithms share similar modules for grouping hypotheses and for searching local neighborhoods of features, but use the modules differently to achieve the different goals. A comparison of the algorithms and their relative strengths gives some insight into the trade-offs between the two approaches.

Figure 3.1 shows block diagrams of the three algorithms. The dashed box indicates that the model is constructed before recognition begins, using the same feature extraction modules as are used on the image during recognition. After initial matching, the algorithms

diverge. The following outlines briefly describe the algorithm modules, in order to give some overall perspective of the detailed explanations in later sections. The first algorithm consists of three parts:

1. Initial matches are made between all similar types of image and model features. A parameter associated with each feature is used to eliminate particularly bad matches. For each match, a hypothetical pose of the model in the image is formed, using the position and orientation of the features.
2. The hypothetical poses are structured so that they may be indexed by the value of the pose. This is done by placing pointers to a match at all those locations in an array which correspond to a pose consistent with the match. (A variation of the Hough transform). Pose cells with the most matches are found, indicating likely poses.
3. For each of the most likely poses, each of the matches stored at that pose are used as seeds in a verification process. The neighbors of the match along its contour are checked for further support of the hypothesis.

Finally, the hypotheses are ordered using the number of matches as a measure of quality. Currently, the first match is taken as the conclusion. This algorithm is heavily data-driven, with a verification stage at the end.

The second algorithm uses model information about the connectivity between features together with the hypothetical initial matches to control the order of image data processing at an earlier stage. The resulting extended matches are grouped by pose. It essentially tries to perform step 3 before step 2:

1. Initial matches are made as above and pruned, but hypothetical poses are not formed.
2. Contour neighbors are explored for each initial match. Correct matches will have more matching neighbors. This can be thought of as extending the size of the feature. For each extended match, a good estimate of the hypothetical pose can be made.
3. Likely matches are grouped by pose. This could be done with an array as in step 2 of the first algorithm, but because there are fewer matches, they can be grouped by cluster accretion instead.

Again, the largest group is taken as the conclusion. Step 2 is hypothesis-driven, but step three is still data-driven.

The third algorithm uses the same feature extension module as the second algorithm, and then proceeds with a further hypothesis-driven stage:

1. Initial matches, as above.
2. Feature extension along contours, as above, including an estimation of the pose.
3. For each hypothetical pose, the positions of other model features in the image can be predicted. For each prediction, the closest image feature of the proper type is found and matched if similar enough to its predicted value. This technique finds matches that are not on the same contour, so could not be found using connectivity. The closest features are tried first, and the estimate of the pose is updated as new matches are found.

Again, the hypothesis with the most matches is taken as the conclusion. Both of the final stages of this matching strategy are hypothesis-driven.

The following sections describe the modules of the algorithms in detail, in the order that they are executed during the recognition process. In section 3.1, the edge extraction process is discussed. It is not an emphasis of this thesis, but the effects of edge detection on the rest of the algorithm can be important, and are not necessarily obvious. The method used is standard: the image is convolved with two Gaussians, and the zero-crossings of the difference indicate edges. Section 3.2 describes how the edge contours are traced to construct orientation graphs, and why an orientation graph is a very useful abstraction. In section 3.3, the feature finding process is described, and a set of criterion for features is presented. Section 3.4 introduces the pose-indexing process with an example of a simple Hough transform, followed by a discussion of several aspects of the method. In Section 3.5, the topic continues with a detailed presentation of a probabilistic interpretation of pose-indexing. By taking a probabilistic approach, many approximations and assumptions are revealed in the standard Hough approach, and a few of the implied improvements are used. Section 3.6 then describes the actual implementation of the pose array. Section 3.7 presents a method for extending feature matches by exploring the contour neighborhoods of the model and image. In the first algorithm, this process serves to verify hypotheses and

find further supporting evidence. In the second algorithm, these extended matches provide accurate pose estimations which are grouped using a simple cluster accretion method, described in Section 3.8. In the third algorithm, the extended match poses are used to predict other potential matches, as described in Section 3.9. All of the algorithms have been implemented, and are successful. They also provide some insight into the distinctions and trade-offs between two general concepts: data- and hypothesis-driven recognition. Results and a discussion are presented in Section 3.10.

3.1 Edge Extraction

The edge-extraction module is not the focus of this thesis, but is worth discussing. It is a standard implementation of the difference-of-Gaussian (DOG) approximation to the Laplacian-of-Gaussian operator ($\nabla^2 G$) [Marr, Hildreth 80], and was largely copied from the recognition system used by Grimson and Lozano-Pérez [84,85].

3.1.1 Edges as an Initial Abstraction

To be able to recognize objects using relatively natural images of the world, as opposed to images of objects under special lighting conditions, is considered to be an important and viable aspect of a vision system. Most earlier systems have worked from binary images, in which the presence of objects is unmistakable due to backlighting or other special preparation of the background or objects. Such requirements restrict the applicability of the system, but make the job of finding the outlines of objects very simple. When images with natural lighting are considered instead, the problems of shadows, low contrast, and uneven illumination make simpler methods unacceptable. It is no longer clear how to define an edge. Instead, places in the image where the intensity is changing quickly (how quickly?) are taken as edges. In a more general language, they are the first abstraction of the image, or the lowest-level features. The name “edges” has been kept because a physical object edge usually causes a sudden change in intensity in an image. However, sometimes a physical edge does not cause an image edge feature, and more often, the so-called edges are caused by shadows, highlights, or changes in reflectivity on the object’s surface such as scratches, colorings, or texture. Highlights and shadows, in particular, are not part of the object and will not usually be in the same position relative to the object. This kind

of extraneous information makes recognition harder for an algorithm that assumes that all edges are physically real. While they contribute to the size of the problem, most algorithms are not forced to explicitly distinguish between physical and non-physical edges, since more extraneous edges are introduced simply by physical edges not belonging to modeled objects.

A more fundamental aspect of edges as a primary abstraction is what information they do not capture. The use of edges in some form is one of the few common threads among vision algorithms, probably because the highest concentration of information in images occurs at sharp changes in intensity. It is presumed that areas of constant or slowly varying intensity do not tell much about the world. Therefore, algorithms which use only edge data will not be able to analyze smoothly curving surfaces from shading. Also, since it is not usually safe to assume that an object will be either lighter or darker than its background, the direction of the intensity changes across the edges are often discarded, making foreground/background region sorting difficult or impossible. This is some of the information thrown away by the edges used in my algorithm, with no better reason than for the sake of simplicity.

3.1.2 The Laplacian and Gaussian Operators

The Laplacian is the sum of the squares of the second derivatives in the x and y directions. It is a second derivative, so in order to find the edges along which the image intensity is changing most rapidly, the points at which the value of the Laplacian cross through zero must be found. The Laplacian is not the only operator which can be used to detect changes in a two-dimensional scalar field, such as an image¹. Other possibilities include the second derivative along the gradient, which is the direction in which it is maximum, or other combinations of the derivatives in the x and y direction. For a complete discussion of the issues of edge detection, see Canny [83]. The Laplacian does have several properties which are notable. One is that it can be implemented by a convolution of the image with several impulse functions. Also, the zero-crossings are guaranteed to form a closed contour if the image is "smooth" enough to allow a second derivative. Practically, the smoothness requirement is not just a mathematical detail. Any second derivative operator is going to

¹In this section, I use mathematical language which is intended for continuous functions, when in fact an image is a discrete set of intensity values. The actual implementation uses differences, for example, instead of derivatives.

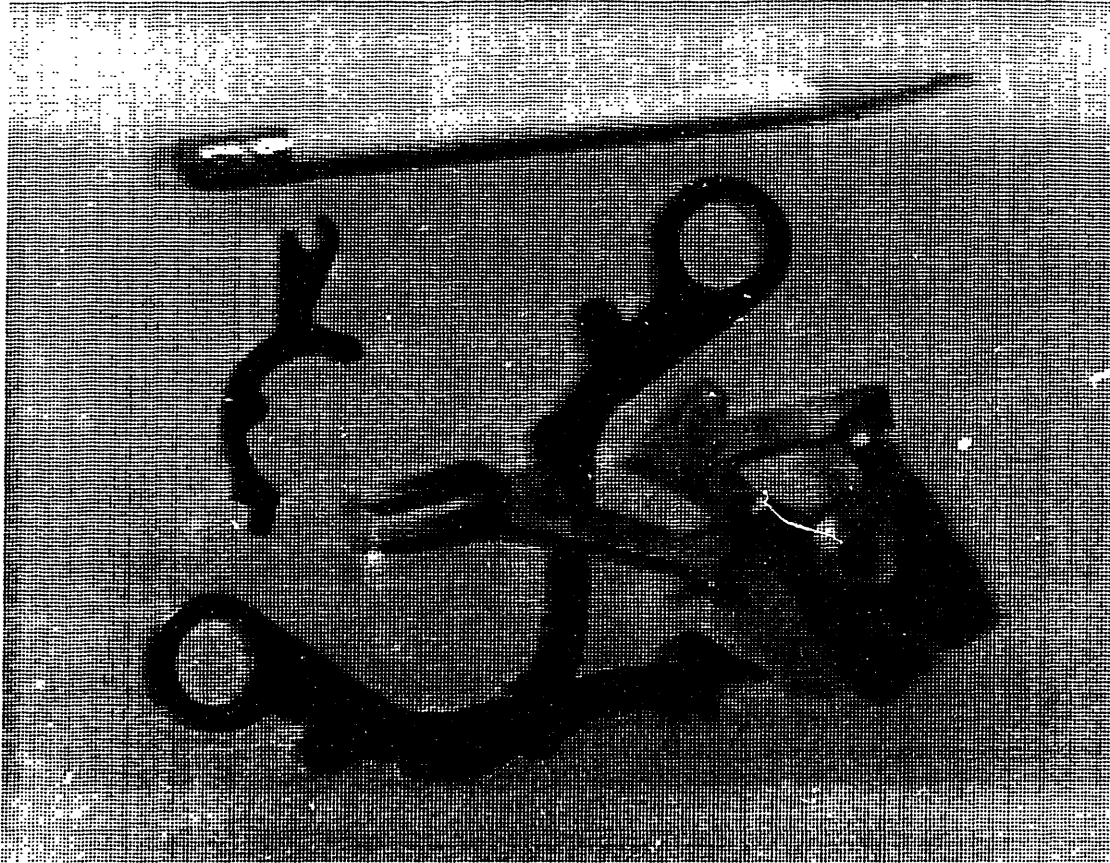


Figure 3.2: Original image (halftoned for photocopy reproduction).

amplify white and other kinds of noise, creating a serious problem. The standard response to this problem is to smooth the image by convolving it with a Gaussian operator, which causes each pixel to take on a value equal to a weighted average of its neighborhood. Since convolution is associative, the Laplacian can be applied to the Gaussian once and for all, and the result can then be applied to each image to perform both the smoothing and edge-finding at once. The resulting operator is often described as looking like an inverted Mexican hat (see Figure 3.3). This shape can be closely approximated by the difference of two Gaussian curves of different sizes (DOG). Because there are mathematical tricks which make a Gaussian particularly easy to convolve with an image, the DOG approximation is very easy to calculate.

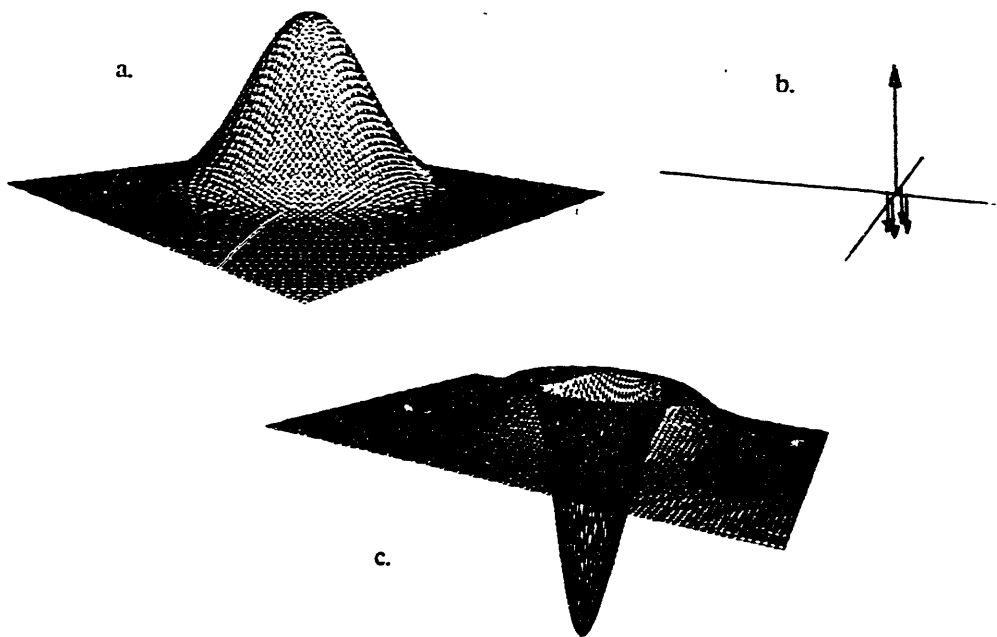


Figure 3.3: a. Gaussian, b. Laplacian, c. Convolution of Gaussian and Laplacian, resembling a Mexican hat.

In order to apply smoothing, however, there is an important parameter which must be determined, and that is the choice of the size of the Gaussian. While some smoothing is needed to prevent the entire image from being covered with edges, the more smoothing that is done, the less detail is detected. A popular way of approaching this problem is to find the edges using Gaussians of many different scales, and then to use the whole set of resulting zero-crossing images to form a “scale-space representation” of the image. There has been some work using features found in this space to identify objects in images [*], but so far it is very susceptible to highlights and occlusion since it depends on uninterpreted image intensities to represent objects. In the implementation I am using, the size of the Gaussian is fixed at a value which makes a reasonable trade-off, preserving details which are the size of a few pixels and yet not interrupting connected edges the size of half the image. This fixed value would probably cause more problems if my algorithm was trying to perform recognition of objects at different scales. But because the objects are always the same size, obscured details are consistently invisible, and the arclength of the boundary (which is used to index higher-level features) is not significantly affected.

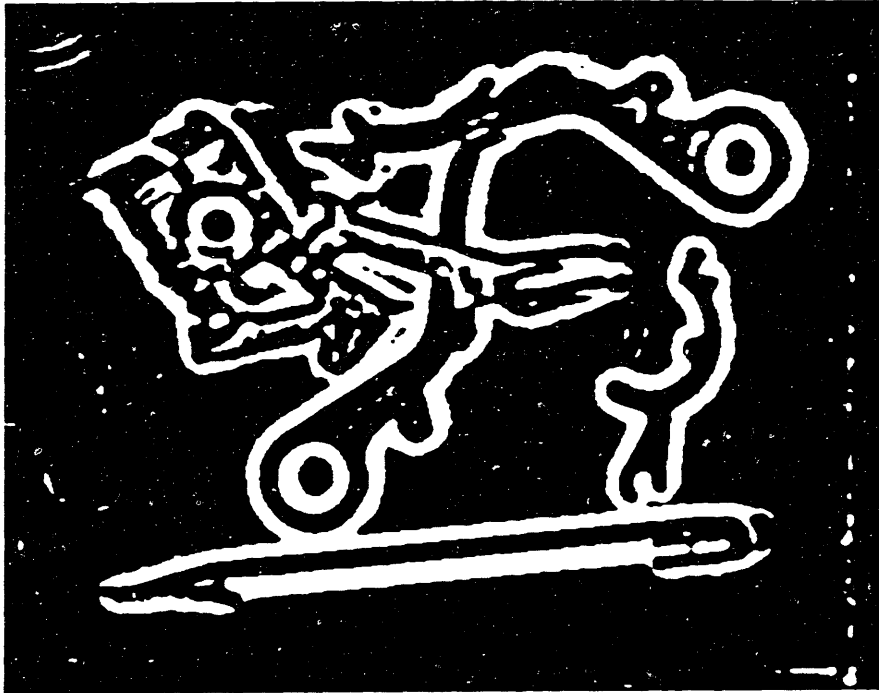


Figure 3.4: *The sign of the convolution of the image with the DOG operator.*

3.1.3 Zero-Crossing Detection

The result of the DOG operator is an array of values which indicate the magnitude of the second derivative of the smoothed intensity of the original image at that pixel (the Laplacian array). The sign of this array is shown in Figure 3.4. Where the value of the Laplacian crosses zero, an edge is indicated, as shown in Figure 3.5. In order to assure the proper connectivity of pixels which are designated as being on the edge, the four immediate neighbors of each pixel are checked to see if the sign differs either vertically or horizontally. These pixels are set to one in a zero-crossings array.

There are several problems with this edge detector, as can be seen in Figures 3.2 through 3.5. While it is nice that isolated objects will usually cause a closed connected contour around their boundary, in more complicated situations, the closed contours will be forced to wander away from strong edges in order to be closed. At sharp corners, such as the pencil tip, and “T” junctions, where three different grey-levels come together, significant separation is notable [Berzins 83]. Perhaps the most noticeable problem is the “reset” contour, a second outline outside of each object which occurs as the Laplacian returns to zero in uniform areas of the image. This may be a problem of the specific implementation

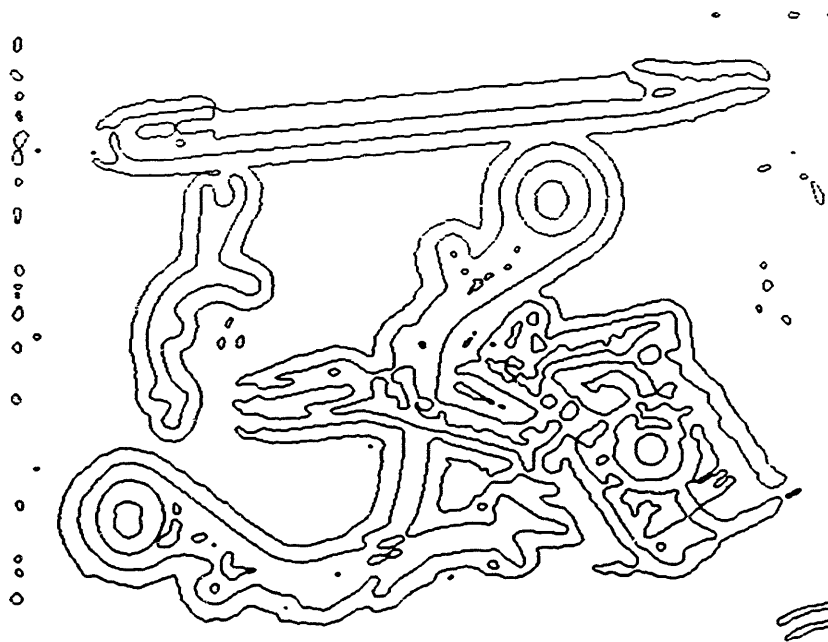


Figure 3.5: *Zero-crossings before thresholding. Note: extra "reset" contours, deviation at pencil tip, small circles in uniform areas.*

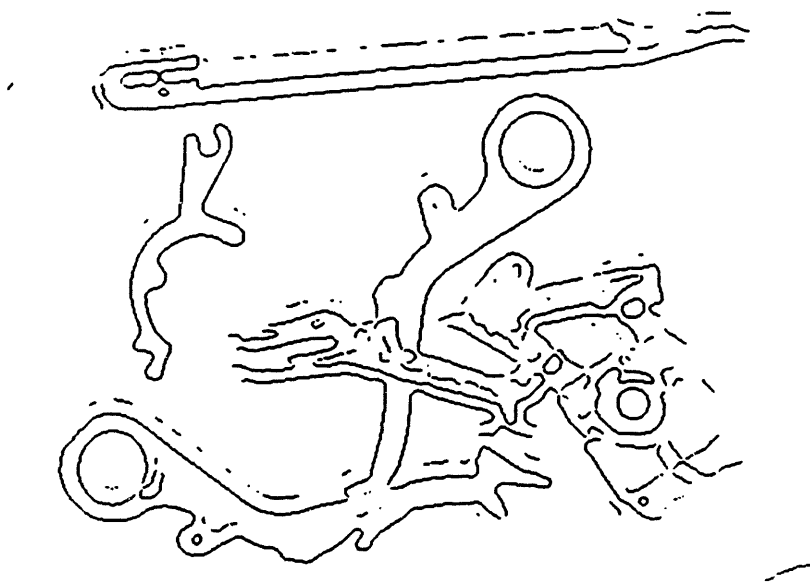


Figure 3.6: *Zero-crossings after thresholding. Note small breaks at high curvatures along useful edges, and remnants of unwanted edges.*

that is avoidable, since it does not usually appear in other zero-crossing detectors. A more commonly acknowledged problem is that in relatively constant areas, even very weak fluctuations will cause zero-crossings. Further discrimination is needed to use the zero-crossings as edges.

A simple but imperfect way to eliminate some of the unwanted zero-crossings is to measure the magnitude of the third derivative, indicating the “strength” of the zero-crossing. This is approximated at each zero-crossing pixel by the sum of the squares of the horizontal and vertical differences between the neighbors in the Laplacian array. Edges where this value is below a fixed threshold are removed. (If no thresholding were to be done, only the sign of the Laplacian would be needed). As can be seen in Figure 3.6, a simple threshold can leave some undesirable remnants, while also breaking real edges where they are weak. There is definitely a need for a more adaptive method of eliminating weak zero-crossings, particularly because a fixed threshold tends to break good contours near sharp corners. Several methods are proposed by Canny [83], and should be used in future implementations.

3.2 The Orientation Graph

At this stage, some image information has been lost and the edge information has been isolated in the zero-crossings array, but not in a very compact form. Many algorithms summarize the edges by fitting straight line segments to the connected contours. Depending on how this summary is used, it may be adequate since the segments can be made to fit the curves very closely. However, no one has yet proposed a canonical method for segmenting curves such that the segmentation would be repeatable under changes in orientation, occlusion, and small perturbations. Therefore, one-to-one matching of segments between model and image segments would either have to be carefully forgiving, or it will fail. In response to this problem, I keep information from every pixel along the contour, and find features in it to summarize the contour information as a discrete set of symbols, rather than segmenting it. Since the information in the zero-crossing array is indexed by position in the image, and does not explicitly indicate the connectedness of the contours, the contours are turned into a graph of orientation versus arclength. A similar encoding has been used by several researchers [Turney, Mudge, Volz 84; Asada, Brady 84], but for different purposes.

3.2.1 Initial Formation

The entire zero-crossing array is scanned from top to bottom. When a set bit is encountered, the connected contour is traced along the outside, pixel by pixel, until it ends (as detected by back-tracking) or returns to the starting point in the case of closed contours. This is done to find a beginning point. From here, the contour is retraced in the opposite direction, and the position of each pixel in the array, along with the direction of the pixel jump (indicating the orientation of the tangent at that point) are stored in arrays indexed in units of pixels of the arc-length. Thus, a record of orientation versus arc-length is formed. As the contour is retraced, the bits are cleared in the zero-crossings array. By the nature of the formation of the contours they will never form a T junction, but occasionally touch each other tangentially, so the removal of one contour only rarely affects others.

The direction of the jump from one pixel to the next is a crude indication of the tangent angle. First, it can take on only eight discrete values, counting diagonal jumps. A perfectly straight line might produce a repeated pattern of five horizontal jumps and one diagonal jump, and this should not be interpreted as a staircase. The orientation graphs are therefore smoothed with a Gaussian curve having a standard deviation of about 3 pixels. Second, in order for a contour to produce a similar orientation graph under any global rotation, the fact that diagonal jumps are longer than horizontal or vertical jumps must be taken into account. A circle, for instance, should be a smooth ramp since the tangent angle is varying smoothly with arc-length. It will look like a wavy staircase with four steps when plotted with one jump per unit length on the arc-length axis, since diagonal jumps move further around the circle. I have found a simple but effective method for reducing this imbalance. An arc-length counter is kept as the orientation array is being filled. For every horizontal or vertical jump, it is increased by one. For every diagonal jump, it is increased by $\sqrt{2}$. Whenever the counter is greater than one, the current pixel data is added to the end of the array, and the counter is decremented by one. In this way, horizontal and vertical jumps are recorded exactly once, while diagonal jumps may be stored twice to make up for the longer jump. After smoothing, this method produces ramps of constant slope for circles in the image.

3.2.2 Advantages and Disadvantages of the Orientation Graph

The orientation graph provides a representation for the contours in an image which is independent of its position within the image, and mostly independent of its orientation (it causes only a shift of the entire graph). This is very important, since the position and orientation of modeled objects in the image are not known ahead of time, so cannot be required in order to interpret the image data. (The positions along the image contours are kept in order to make accurate hypotheses at a later stage.) At the same time, the graph does capture the shape of the contour, and represents it in one dimension (arc-length). This step does not discard any information, since the contour was originally only one-dimensional, but was in a two-dimensional space. Bringing it into a one-dimensional array simply increases its accessibility. In addition, the orientation graph is derived from local data, so that shapes in one part of the contour are unaffected by any other part of the contour. This makes the shape of the graph repeatable despite partial occlusions, a vital requirement for the system.

While the orientation graph captures the relative angular shape of the contours, such as bumps and corners, it does obscure the spatial shape of the object. The orientations can theoretically be integrated to reconstruct the spatial shape, but smoothing and any other small integrated errors quickly accumulate. The positions along the contour are saved separately for this reason, and they are currently used only in the more global recognition stages. Therefore, during the local matching stage, the spatial features of two objects may be different, but if the angular characteristics are similar, their orientation graphs will not reflect their differences. Furthermore, the orientation graph records only the shape of the object boundary. Nowhere does it make explicit the shape of the object's area, or even which side of the contour the object is on. Thus contours connected by the body of an object but not connected by the boundary (due to occlusion or edge detector gaps) are not associated with each other.

Another potential drawback of working with contours in this form is the difficulty of performing the correct processing of closed contours. Since the starting point on a closed contour is arbitrary, distinctive shapes which cross the starting point should not be discriminated against. The information to do so has not been lost, but the representation does not lend itself to circular access. This may seem like a detail, but it is nonetheless a

problem caused by the non-circularity of the one-dimensional array.

3.2.3 Orientation Graph Examples

In the following examples, the image contours are shown above the orientation graphs of the same contours. The horizontal axis of the graphs indicates arc-length along a contour, and the vertical axis indicates the angle of the tangent to the contour at that point. It takes a little practice to see exactly which part of the graph corresponds to which part of the image. Also shown are the features explained later (in Section 3.3) which may help to show the correspondence. Figure 3.7 contains isolated objects of various complexity. Figure 3.8 shows one of these objects occluded by several others. There are several points to notice in each example.

- The circle in the image produces a ramp in the orientation graph, as mentioned above. Note that the ramp is smooth due to the adjustment for diagonal jumps. A square object produces a staircase graph, with smoothed steps. Each side of the square has a constant orientation as it is traversed, so the graph is level for each side.
- The hole in the widget is a separate contour, so it has a separate graph. There is no implication that a contour will all be caused by one object, or that each object will all be on one contour. The piano has a single continuous closed physical boundary, but the edge-finder broke the boundary into two contours (due to a shadow).
- Each graph begins at the top of the contour in the image. Thus the starting point of a graph can be almost anywhere on the object, depending on its orientation. The initial value of the orientation is likely to be close to horizontal, however, since it always begins at the top of the object.
- For each of the objects that forms a closed contour, the starting point is the same as the last point on the object. Unfortunately, their orientation values will not be identical, but will differ by 2π . As a closed contour is traversed counter-clockwise, the tangent orientation must increase by exactly 2π if the contour does not cross itself (it can not), and it joins itself smoothly. If, for some reason, the contour is traced in the opposite direction, the orientation would decrease. This cannot happen with closed curves, but broken contours are equally likely to be traversed in either direction.

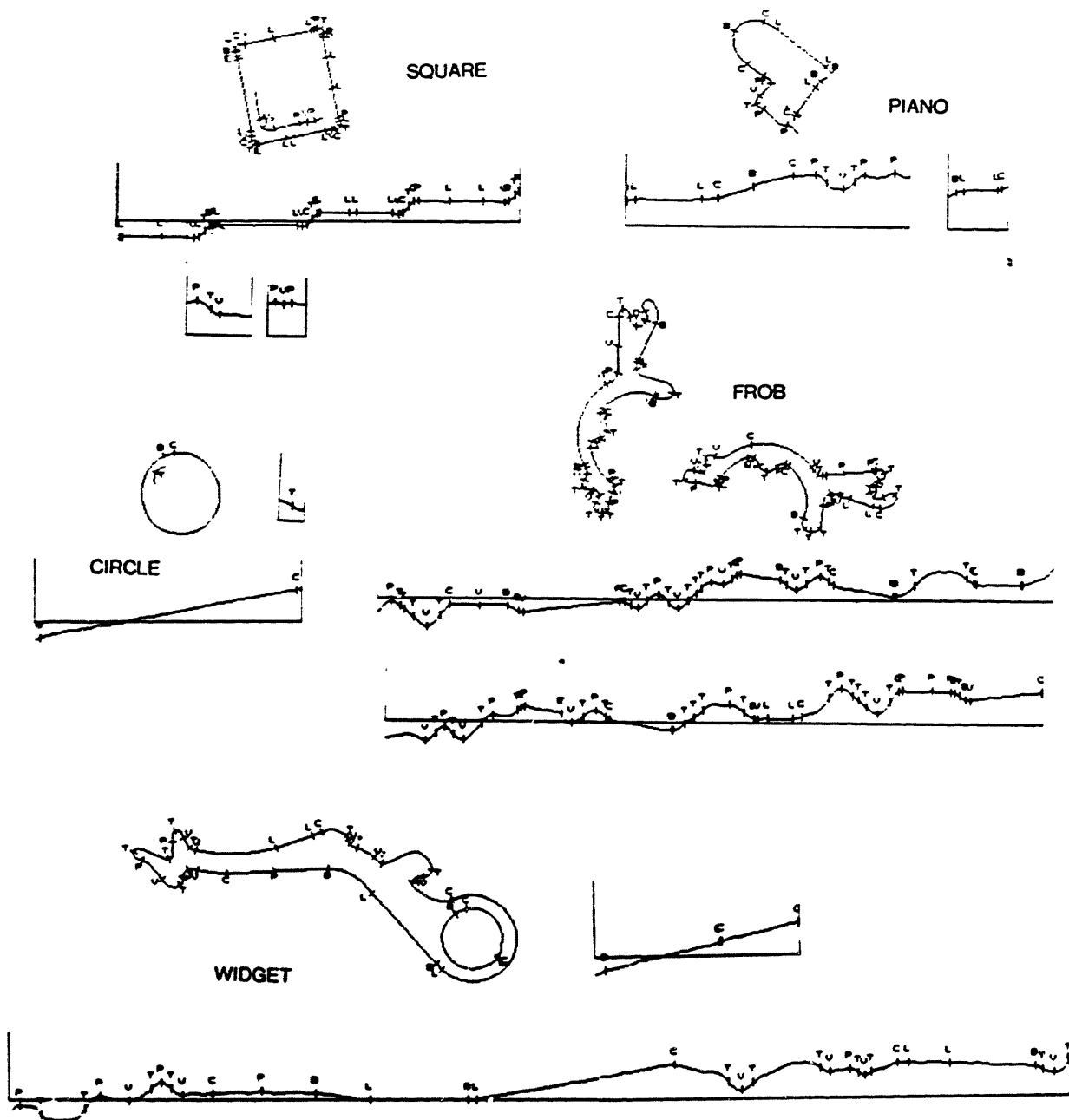


Figure 3.7: Some isolated shapes and their orientation graphs.

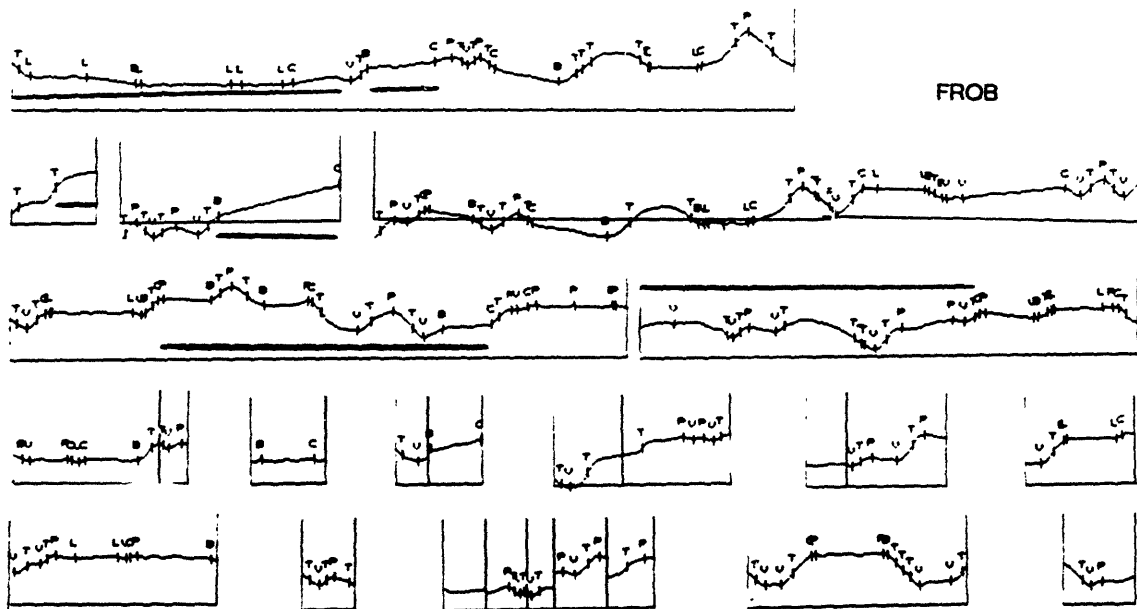
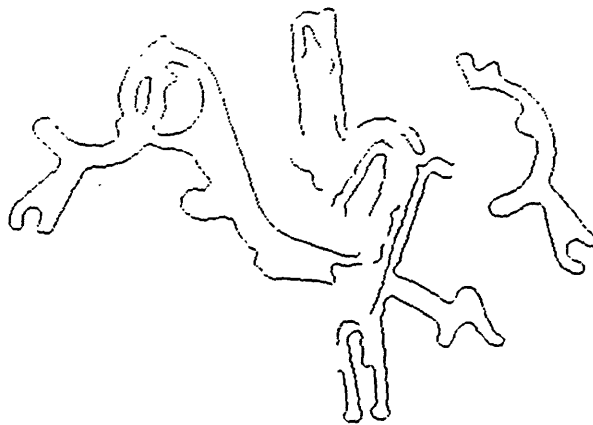


Figure 3.8: Partially occluded objects and their orientation graphs.
 Sections of the WIDGET are underlined.

- Note the effect of rotating the frob. In addition to changing the starting point, the entire graph is shifted vertically as well. However, the shapes of the graphs remain almost identical, and almost all of the same features are found on both.
- In the occluded image, contours from several objects run together. The contour of the widget from the isolated example has been spliced in with the other contours, as indicated by the dark bars. Portions of the original contour are missing completely. The separation between spliced sections can be any amount, and does not have anything to do with the separation in the original graph.
- Parts of the original contour have been traced backwards in the occluded image. Because of this, inside/outside consistency cannot be enforced using the direction of the graph tracing.
- Despite splicing and reversal, the orientation graph is repeatable in those portions of the object which remain visible. The local nature of the graph formation and the diagonal pixel jump adjustment provide this important quality.

3.3 Feature Extraction

The orientation graph reduces the contour information to one continuous dimension. While it might be possible to perform recognition on this representation directly, using auto-correlation for instance, the splicing effect of occlusion and the large amount of calculation required discourage this. Instead, the data is further abstracted to a (discrete) symbolic form which makes explicit the interesting features of the contour's shape. This abstraction is important for several reasons. Not only does it reduce the size of the data, but it focuses the attention of the recognition process, and therefore reduces the amount of computation required to process the data. At later stages, these features are compared with the model features in search of consistent partial matches. By extracting features, the search is limited to a finite number of comparisons, though the number is large. However, the recognition algorithm relies heavily on the accuracy of the features in order to reduce the amount of work needed to discard incorrect matches and keep correct ones. To be useful, any abstraction must correctly summarize its input without missing or distorting information. Thus, there are heavy demands governing the choice of a feature set. I present a set of criteria for

features, then describe the set of features used in these algorithms, with a justification in terms of the criteria.

3.3.1 Criteria for a Feature

The job of a feature is to focus the attention of the recognition algorithm on a distinctive part of the object, and to represent its local neighborhood in an informative way so that accurate matching can be performed. Therefore, a feature must be:

Local. In order for a feature to be robust to occlusion, it cannot be based on a large portion of the object. If it were, then any covering of a part of the contour would render the visible part useless. Features must also be local in the sense that they are not dependent on any global reference, especially one such as the “center of mass” which is significantly perturbed by occlusion, or any origin which requires that the pose of the object is known (since it is not).

Repeatable. One of the most important requirements of a feature is that it can be reliably found in the same location on the object over and over again, despite varying orientations and nearby occlusions. This also means that extra features and feature omissions should be kept to a minimum.

Distinctive. Features must be different enough from each other that they are never switched, and so that inter-feature matches need never be considered. But they must also be distinctive in the sense that they capture what is distinctive about the object, so that they are useful for distinguishing an object from others.

Informative. If the feature can describe its local neighborhood sufficiently well, hypothetical matches may be initially filtered before further investigation. Any early pruning of the search space is very valuable since it will significantly reduce overall recognition times. Also, the features must contain enough information that an image/model feature match can be used to form a narrow hypothesis about the global pose of the modeled object in the image.

Simple. If a feature requires too much processing to extract, then perhaps it should be broken into simpler parts to reduce the computation time. Any complicated processing of the image data should be performed at a higher level of the recognition process

where the model information is available, and where the algorithm is designed to avoid unnecessary work. This is typically a design trade-off.

General. The utility of a feature set will also depend on its ability to describe a wide range of objects. It is very tempting to work with a set of features which are simple, but only successful for a limited class of objects, such as polyhedra.

Other criteria have been proposed, based on slightly different definitions of a feature (see Marr, Nishihara [78]). There is obviously a certain amount of overlap among these criteria, and conflicting pressures are what make the choice of features a problem without a clear solution.

3.3.2 Choice of Features and Their Detection

Based on the above criteria, I have chosen four types of features for the contours which can be extracted from the orientation graphs, as shown in Figure 3.9. They are:

- **Tips:** points of maximum curvature in the image, corresponding to points in the orientation graph where the first derivative of orientation with respect to arc-length is maximum. This is detected when the first derivative is above a threshold and the second derivative crosses zero. The tip feature should appear in exactly the same location regardless of the direction in which the contour is traversed. The value and sign of the first derivative at the tip are kept for further discrimination among tip matches. When an image tip is matched to a model tip, the sign of the slope can be used to eliminate the ambiguity of relative traversal direction. The position of the tip feature along the contour is usually fairly stable, with the worst possible situation being a long constant slope (see the circle feature below).
- **Peaks and Valleys (Points of Inflection):** Local maxima and minima in the orientation graph are found when the second derivative is above a threshold and the first derivative crosses zero. In the image, these are the places where the curvature changes sign, as in the middle of an 'S' or the middle of a smoothed '2'. The peak and valley features are also detected in either direction of traversal, but unlike the tip, there is no sign associated with the feature to disambiguate the traversal direction along one of the contours relative to the other. Also, a peak will never look like a valley (unless the

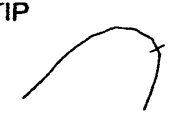
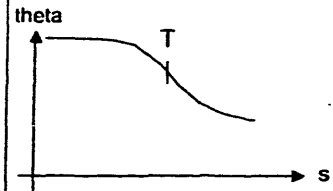

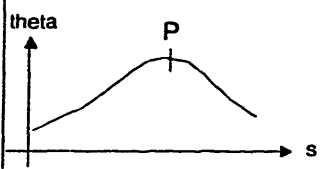
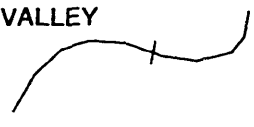
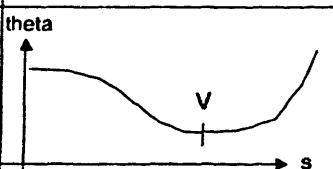
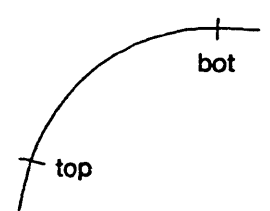
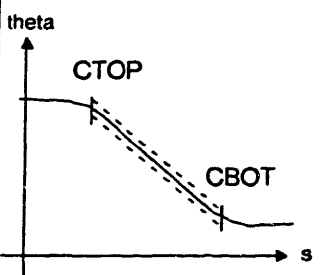
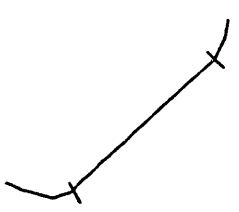
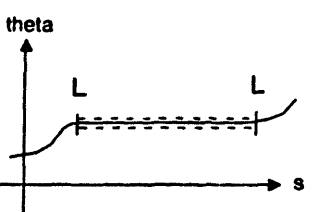
In Image	In Orientation Graph	Test
<p>TIP</p> 		$\frac{d^2}{ds^2}(\text{theta}) = 0$ $\left \frac{d}{ds}(\text{theta}) \right > k_1$
<p>PEAK</p> 		$\frac{d}{ds}(\text{theta}) = 0$ $\frac{d^2}{ds^2}(\text{theta}) < k_2$
<p>VALLEY</p> 		$\frac{d}{ds}(\text{theta}) = 0$ $\frac{d^2}{ds^2}(\text{theta}) > k_2$
<p>CIRCULAR-ARC</p> 		<p>No values outside bounds</p> $ \text{slope} > k_3$
<p>LINE</p> 		<p>No values outside bounds</p> $ \text{slope} < k_3$

Figure 3.9: The feature set, and extraction rules.

flat objects are allowed to be flipped over). This can also be seen in the image. When trying to match an image 'S' to a model 'S', one can be rotated 180° and still line up with the other. No matter how an 'S' is rotated (or in which direction it is traversed), it will never look like a '2'. The value of the second derivative can still be used as a parameter of the feature. Peaks and valleys can be located fairly accurately, too, unless the extremum is leveled (see the line feature below).

- **Circles:** Circular arcs in the image appear as ramps of constant slope in the orientation graph. It is more difficult to detect a ramp efficiently, repeatably, and symmetrically because it does not occur at a single point. The set of possible contour shapes which could be interpreted as a ramp feature is much larger than that for a tip, peak, or valley, so the detector is more complicated. It is also harder to use the feature since it may extend over a large enough section of the contour that it is likely to be partially occluded. Yet, if the feature is detected and matched, it is very strong evidence for the presence of the object, but with a degree of freedom as to the exact position of the object in the image. At each point along the contour, the circle detector tries to grow a straight line out in both directions. If the line can be grown with some slope such that every point along the line is within a max-error threshold of the contour, and if the total length of this line is above a min-length threshold, and the magnitude of the slope of the line is above a min-slope threshold, the detector indicates the length and slope of the feature. Actually, since the line length must be greater than the min-length threshold, an initial check is made to see if the initial (middle) point lies within twice max-error of the line between the two points half min-length away. If it does, the slope is set to best fit the three points, and the interior of the interval is checked at each pixel. If these points are within max-error of the line, then the interval is grown first in one direction, then the other. Because the line is grown in both directions, the process is symmetrical (with respect to traversal direction) up to this stage. However, the circular arc detector described so far would indicate a circle at every point along a ramp, and would find shorter overlapping circles at both ends of it. Therefore, whenever two circle features overlap, the shorter one is discarded. Unfortunately, the only simple implementation of this incompletely defined rule turns out to be asymmetric. I have not noticed significant detection instability due to the

asymmetry, but it is possible, and the detection algorithm is considered to be an incomplete solution.

When a circular arc is finally kept, each end of the circle is taken as a separate feature. This is a simple but imperfect way of reducing its susceptibility to occlusion, and ignoring the extra degree of freedom introduced when two circular features of different lengths are matched. It assumes that one end or the other may be occluded, but not both ends while leaving the middle visible. The method also relies heavily on the stability of the location of the endpoints of the circle, which are not as reliable as the overall detection of the feature and its slope. A more general method of matching circles would probably be useful.

Since the top of a ramp will remain the top when the contour is traversed in the opposite direction, the circle-top and circle-bottom features are distinct types of features. The slope (the reciprocal of the radius) is kept as a strong discriminating parameter, and its sign disambiguates the relative direction of traversal in a match. Theoretically, the circle feature becomes unstable only along curves of slowly changing curvature, when segmentation becomes a problem: where should a circle of one slope (radius) end and another one start? Practically, the endpoints are fairly unstable even along good circles due to a relatively high sensitivity to noise.

- **Lines:** Straight line segments in the image appear as level sections of the orientation graph. A detector similar to the circle detector is used, with the slope known to be zero. As with the circle detector, an asymmetry is introduced when the shorter of overlapping line segments is discarded during the traversal. Also, the line is used to create two individual endpoint features, as with the circle, but of the same type since, unlike circles, there is no way to distinguish between one end of a line and the other. Therefore, for one image line segment and one model line segment, there are four ways to match the two pairs of line features. For each of the matches, there is no relative traversal direction ambiguity, since each endpoint indicates which side the line is on. The line length is kept to help discriminate matches; image lines more than some threshold longer than the model are not used, but image lines shorter than the model are kept since they could be caused by occlusion. The line feature becomes unstable on slight curves (very slow ramps in the orientation graph) when long enough lines



Figure 3.10: "Inverted" model features.

can be formed, but the endpoints move. Like the circle feature, the endpoints are fairly susceptible to noise.

3.3.3 Evaluation of Chosen Features

The orientation graph of the widget in Figure 3.7 shows the features extracted from the object, which can be used as a model. Figure 3.8 is a typical image, with the features shown. Portions of the image orientation graph caused by the modeled object are indicated, and the repeatability of the features can be examined in these regions. The locality of the Tip, Peak, and Valley features is guaranteed from the local independence of the orientation graph and the local tests used to measure the first and second derivatives. Circle and Line features are also local in that they depend only on the part of the contour they describe, and are not referenced to any global origin. In the case of longer, less local, circular or linear features, the use of both endpoints reduces the feature's susceptibility to occlusion. As can be seen, the repeatability of the Line and Circle features is not as good along the middle of the line or circular arc as it is at the ends of the feature.

In Figure 3.10, the features are approximately inverted to demonstrate that they capture most of the object's shape. The relative lack of overlap reflects the orthogonality of the feature set; the only common redundancy is the presence of tips in the middle of small circular arcs. The ability of the inverted features to look like the original object demonstrates the extent of the information contained in the features. Also, from any match of an image

feature to a model feature, a complete estimate of the hypothetical position and orientation of the model in the image can be formed. (If lines and circles were more thoroughly utilized, instead of using endpoint features, a limited degree of freedom would remain in the hypothetical pose). The descriptive power of the feature set can be estimated from the lack of undescribed boundary. If almost every part of the contour is represented, then it is at least possible that the features will enable accurate discrimination among different objects.

The general applicability of the feature set to objects in the two-dimensional domain can only be evaluated through testing a wide range of shapes (see Section 3.10). The features are relatively simple, which reduces computational costs and is a reassurance that they are elementary in some sense.

3.4 Pose Grouping with an Indexing Array (Hough Transform)

The modules described up to this point are the first stages common to all three algorithms presented in this thesis. They begin with a two-dimensional array of intensity values, and produce a representation of the contours in the image, based on the features described above. These modules are used both to create a model of an object, and as the first stages of the recognition process. The rest of the recognition algorithm attempts to match parts of the image representation with the model. During matching, various hypotheses about the position and orientation of the model in the image will be developed, either explicitly or implicitly. This section describes a method for organizing and combining the hypotheses into compatible groups. It serves as the main recognition module for the first algorithm, which is driven from the image data. The second algorithm must also perform this task, but it can use a simpler method since the data has been reduced by a different main recognition module.

The pose indexing method was popularized by Hough [62] as a transformation, but it can be thought of in several other ways. It provides a means of grouping consistent hypotheses by placing pointers to them in an array of appropriate dimension, such that similar hypotheses are close to one another. Each hypothesis is taken as a vote for a particular pose, and the votes are tallied in the array. A significantly extended implementation by Ballard [81] relies on a high degree of accuracy in the hypothetical poses in order that consistent

poses fall in the same array element. If, instead, the array is viewed as a histogram, and more generally as an approximation to a probabilistic distribution in hypothesis space, two modifications are suggested which make the method more flexible and successful. These are: to place votes in as many buckets as are consistent with the feature match considering the conditional error distribution, and second, to weight each distribution according to the *a priori* probability that the match is valid. With these modifications, the method works relatively well, although it still suffers from the need for a large amount of memory and a trade-off in bucket size concerning noise reduction, resolution and grouping power.

3.4.1 A Simple Hough Example

The concept behind the Hough transform is quite straightforward, but its implementation can be somewhat confusing. Figure 3.11 illustrates an example application of the transform to recognition. Consider a triangular model, defined with respect to the origin as shown (the origin is arbitrary, but must be consistent). The objective is to find places in the image where parts of the model appear, and to collect all the found parts into consistent groups. This is done by making all possible matches of the model with the image, and for each match, making a vote for the position and orientation of the model origin which aligns the match. The votes are kept in the pose array, in which there is an array bucket for every pose. Each bucket keeps track of the number of votes for that pose. At the end, all the buckets are checked for local maxima or for values above some threshold. These buckets indicate poses of the model in the image such that many matches are aligned. The poses can therefore be interpreted as locations of the model in the image.

The application of the transform depends on the definition of a match. For this example, with only straight lines, a match will be defined to be the alignment of a model line segment centered on an image segment. Given the image shown, there are five image lines and three model lines, so there are fifteen possible pairings. Unfortunately, each line pairing can be matched two ways: with the object origin on one side, or 180 degrees around with the origin on the other side. So there are thirty matches, and there will be thirty votes. Five of the matches with arrows to their votes are shown. The match is made, and the position and rotation of the model origin is noted. The cell (box) in the pose array corresponding to that x and y position and that orientation, is incremented. Finally, after all matches and

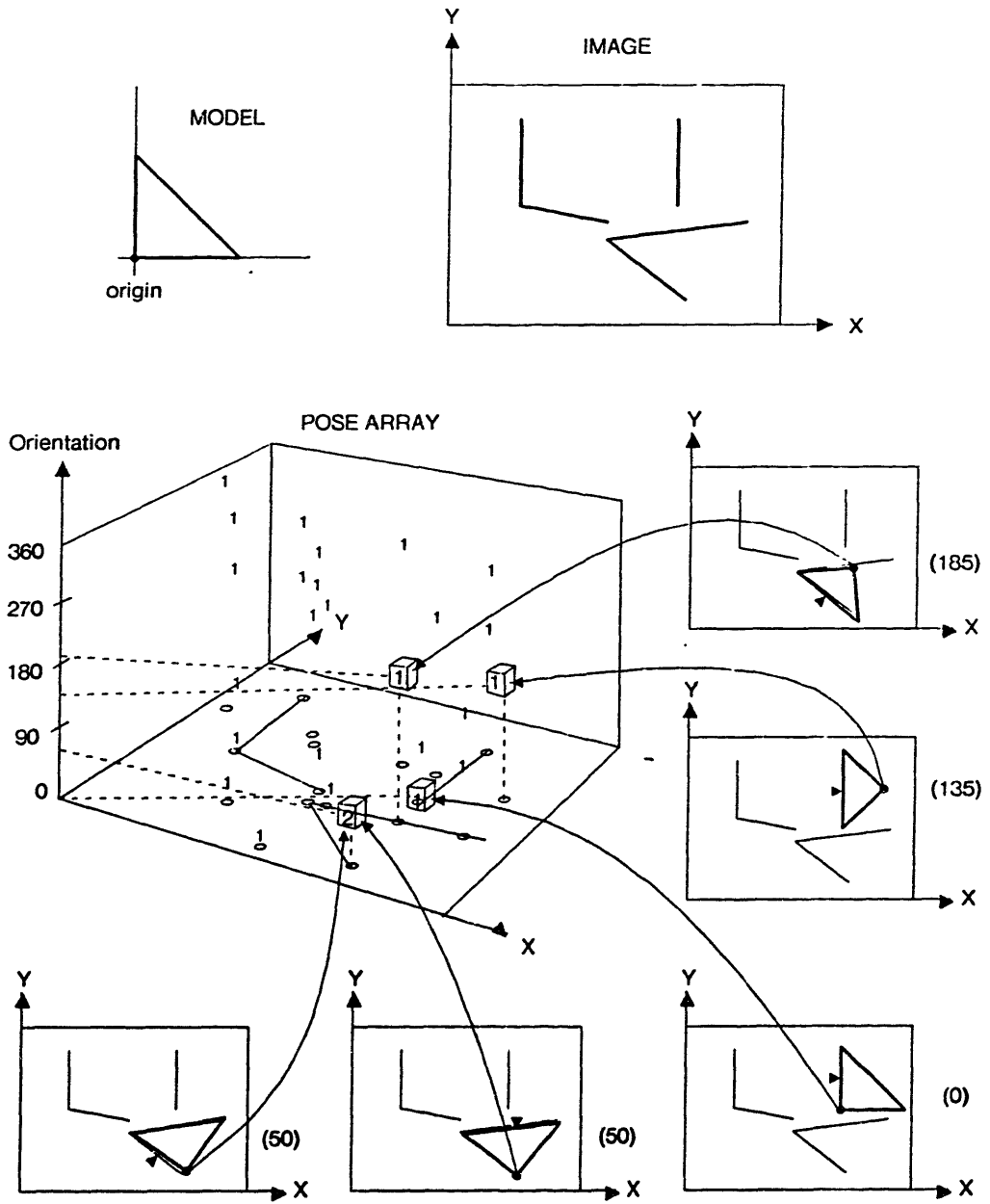


Figure 3.11: An example application of the Hough transform.

votes, the array contains the values shown (some not boxed). The matches in any bucket can come from lines anywhere in the image— anywhere that there was evidence supporting the pose. Note the local maximum bucket containing two votes. The bucket corresponds to the a pose which aligns two line segments. There is another potential location of the triangle in the image, but the two lines are not at exactly the proper angles, so the votes were not placed in the same bucket.

Several weaknesses are apparent in this simple application. First, the method is not very tolerant of noise— if a the lines of the model are slightly distorted, the votes will not go in the same bucket even though there is good evidence for a triangle at that location. Second, the size of the array cells has a lot to do with whether close votes are grouped, and it is possible for very close votes to be split between cells. Third, the total number of votes, based on the amount of data and the method of matching, will affect the height of the peaks relative to other cells in the image. The signal to noise ratio is not very good. Fourth, the particular type of matching used in this example would do poorly if occlusion were introduced, since the midpoints of the lines would vary. Perhaps the line segments could slide along each other, making multiple votes along the way. Consider curved surfaces; how many votes would be needed? Filling the buckets can be computationally expensive.

3.4.2 Interpreting the Pose Array

The Hough Transform was introduced in 1962 (US patent 3,069,654) as a means for detecting parameterized curves in binary images. Rather than trying to identify the position and orientation of a modeled object in an image, it was originally a method for estimating the radius and position of a circle, for example, from portions of the circle. The parameter space had as many axes as there were parameters in the shape, and took each piece of edge data as a vote for the values it supported. In this application, it can be seen as a transform from image space to parameter space. Many improvements and generalizations have been made, particularly by Ballard [81].

The strength of the approach comes from the fact that the information implied by features scattered over the image is organized by the values implied by the features, rather than being organized spatially. In this sense, it is a hash table, hashing the hypotheses on their parameter values. Since the goal of the table is to find all the hypotheses that are

consistent with a particular pose value, it is necessary to push a pointer to a hypothesis into every bin representing a pose with which it is consistent. That is, the hashing of a hypothesis formed from a match with limited reliability is one-to-many, not one-to-one.

The same conclusion can be reached via another useful interpretation, which comes from a probabilistic view (of Bayesian flavor). In this view, the desired output of a recognition system is $\Pr(H|I)$, a probabilistic distribution over the possible hypotheses about the model in the image, given the image. Peaks in this distribution signal likely hypothetical poses, and it provides the perfect measure of “goodness”: the probability of the hypothesis. Since a continuous distribution cannot be represented on a finite machine (except analytically—possibly worth trying but very complex), it is approximated by a histogram. The possible values of H are broken into discrete “buckets”, and a probability is associated with (“fills”) each bucket. Thus, from this perspective, the pose-array is a histogram. This allows access to probabilistic approaches in determining which buckets to fill. This perspective has proven to be quite useful, and is significantly expanded in Section 3.5.

3.4.3 The Histogram Segmentation Issue

The discretization of the set of all poses by its representation as an array must be done carefully. In particular, the size of the buckets and the way they are interpreted can make the difference between success and failure of the method. There are several trade-offs involved:

Resolution— If the histogram buckets are too large, the location of the peak buckets will not be informative enough to be useful, since each bucket stands for a wide range of values.

Noise— Large buckets will also tend to drown out sharp peaks in the distribution, since every bucket will have many entries. The value of a bucket represents the average density of the distribution within the entire bucket, so important small high density regions will not be represented since the surrounding area will be averaged in.

Population— If not enough data is used to fill the histogram, or if there are too many buckets, each bucket will either contain an entry or be empty. The resolution of the histogram value has been sacrificed.

Computation— The number of buckets is proportional to n^3 , in the case of a three-dimensional array, where n is the number of divisions along each axis. Any algorithm which must operate on each bucket sequentially or proportional to the density of the buckets may suffer from very long execution times as n is increased. Naturally, size requirements are also explosive.

One way to reduce the conflict of these considerations is to use a relatively dense array, but to consider the buckets in groups when searching for the peaks in the distribution. Thus, resolution in both pose and probability value can be preserved, at the cost of extra computation. If necessary, the buckets can be inspected in a course-to-fine hierarchy, using the larger groups to restrict the application of the finer inspection. Another alternative is to fill more than one bucket for each piece of data. This is equivalent to the first method if the same configuration of buckets is filled as was used to group the buckets in the first method. If so, then the number of buckets versus the number of data points will determine which is more efficient. However, it will be seen that it is very useful to fill other configurations of buckets, depending on the data, and that this can significantly improve the success of the histogram method.

3.5 A Probabilistic Interpretation

As introduced in Section 3.4.2, using an array to organize hypothetical poses can be interpreted as approximating a probability distribution by a histogram. Each cell of the array is a bucket of the histogram, and should contain the probability that the hypothesis associated with the bucket is true, given the image. One way to fill the buckets would be to look at each one sequentially, finding an approximation to the probability of each possible hypothesis, one at a time. By considering this process carefully, it can be seen that a more efficient method for filling the histogram is available, in which each feature match is used to fill whichever buckets it is relevant to. This method closely resembles the extended Hough transform, but is derived from a very different perspective.

3.5.1 Definition of Terms

The problem is to determine an efficient way of finding $\text{Pr}(H|I)$. I is the set of features which have been extracted from the image, $\{f_i\}$. I can take on many values, but its

particular value is found at the beginning of each run of the recognition algorithm. H is a hypothesis about the relation of the model to the image, composed of two related parts. The first part, T , indicates the transformation necessary to place the model in the image. For two-dimensional vision, T is a translation in x and y and a rotation in the x - y plane. The space of all T is therefore three-dimensional, with axes x , y , and θ . T can take on a continuum of values, but the histogram divides the space into discrete chunks of volume. The second part, M , indicates which matches led to the hypothetical transform. Each M is a complete assignment of every model feature to an image features, or to a null feature to indicate that the model feature has been occluded. Each M is made up of a set of matches, m , where each m is a pairing of one model feature to one image feature, (f_m, f_i) . The set of all M is discrete by nature, however, it is also very large. Very. For example, if there are twenty model features and fifty image features, then there are $(50 + 1)^{20}$ distinct M . It would take quite a while to consider every M , and even more time to consider every combination of T and M . Fortunately, for any particular T , only a few M are likely enough to be worth examining individually. For this reason, I like to envision the space of all H as a small set of M associated with each point in the T space. In my opinion, M is a more fundamental part of recognition than T , although T is the focus of most recognition systems. M says, "I recognize this part of image as being the same as that part of the model", whereas T says, "If I move my model exactly like this, it will look like the image".

3.5.2 A Bayesian Flip

$\Pr(H|I)$ is difficult to calculate directly; it asks the "inverse", recognition question, "Where and what object caused this image?". The first step is to convert this to the "forward", projection question, "How might this object look in an image?". The following relation is just what is needed:

$$\Pr(H|I) = \frac{\Pr(H)}{\Pr(I)} \Pr(I|H).$$

In words:

The probability $\Pr(H|I)$, that a particular hypothesis H is true given the observed image I , can be found from the probability $\Pr(I|H)$, that the image could have happened given that the hypothesis really is true, multiplied by an

a priori estimate that the hypothesis could be true, $\Pr(H)$, and scaled by a normalizing constant, $1/\Pr(I)$.

The normalizing constant is not important in this application. The *a priori* distribution can be further broken down: $\Pr(H) = \Pr(T, M) = \Pr(T) \Pr(M)$, since $\Pr(T)$ and $\Pr(M)$ are independent without I available to link them. $\Pr(T)$ might be interesting to use in cases where certain orientations are known to be more likely, such as when a cup is to be found on a table. $\Pr(M)$ is the *a priori* probability that a particular match is true, before the image has been seen or any H has been proposed. It might seem that no information is available to distinguish one M from another. While this is mostly true, $\Pr(M)$ could be used to devalue mappings which are not one-to-one, or to indicate that a model feature is more likely to be matched with the null image feature if a nearby model feature is also occluded. Currently, both distributions are approximated to be uniform. Under this assumption, $\Pr(H|I)$ is directly related to $\Pr(I|H)$.

3.5.3 Getting to the Feature Level

To find $\Pr(I|H)$, it is broken down to the level of the individual matches, where heuristic estimations of the probabilities can be made based on the geometry of the matched features. Given H , the location, orientation, and type of each image feature caused by a model feature can be predicted exactly. Let this predicted image feature be designated f_i^* , a vector indicating type, position, orientation, and discriminating value (just like all features). The difference between the predicted and actual image feature values is due to error: $f_i = f_i^* + e_{f_i}$. A distribution for the error, $\Pr_e(e_{f_i})$ can be justified analytically, or from empirical evidence based on many observations of the way model features appear in images. Note that it is independent of H —the *difference* between the predicted image feature and the measured value is due only to error, which does not depend on where the feature is or which feature it is matched with. This independence allows all matches to be treated in the same way. More importantly, $\Pr_e(e_{f_i})$ for one e_{f_i} is independent of its value for a different e_{f_i} . (It may be useful to use different $\Pr(e_{f_i})$ based on the type of f_i , in which case matches may be treated differently, but they are still independent of each other). With H given, $\Pr(f_i|H)$ can be found from $\Pr_e(e_{f_i})$ using $\Pr(f_i|H) = \Pr_e(f_i - f_i^*)$, where H is needed to determine f_i^* . Because of the independence of $\Pr_e()$, knowing the error at any one image feature does

not help to predict the error at another image feature, so the conditional distributions are independent. Also, the image features *not* caused by the model do not influence $\Pr(I|H)$. Under these conditions,

$$\Pr(I|H) = \prod_{\text{all } f_i \text{ in } M} \Pr(f_i|H).$$

Note that the product is over all f_i in M , including terms for the probability of occluded model features (matched to the null image feature), but not including any reference to other image features. Also, the distribution $\Pr(f_i|H)$ does not actually depend on all of H . H is T and M , where M is a set of matches m , each of which is a model/image feature pairing, (f_m, f_i) . The error probability of a particular match is independent of all other matches, so all m other than the one including f_i can be removed from the condition.

All together,

$$\Pr(\overbrace{T, M}^H | I) = \frac{\Pr(T) \Pr(M)}{\Pr(I)} \prod_{\text{all } (f_m, f_i) \text{ in } M} \Pr(f_i | T, \overbrace{(f_m, f_i)}^m).$$

This equation provides a means of determining the likelihood of any hypothesis from the distributions of each match. However, as mentioned above, it is not feasible to check every possible combination of T and M . The trick is to determine which mappings are likely for each T . For any given T , the model can be placed in the image, and the most likely image feature matches to each model feature can be found by their positional and rotational proximity, and by the similarity of their feature discrimination values. If no sufficiently likely image matches exist, then the model feature is matched to a null feature, indicating that it has been occluded. Returning to the earlier example of fifty image features and twenty model features, this is where the '+1' comes from in the expression for the number of mappings, $(50 + 1)^{20}$. In practice, this is not just a slight increase in the number of possible matches for each model feature. It is rare that more than one image feature is even a reasonable match candidate. If occlusion were not allowed, evaluation of each H would go very quickly. The only M considered would be made from the best match for each model feature, no matter how bad it was. When occlusion is accounted for, it is quite possible that any sufficiently poor match is incorrect. Thus, there will often be two significant matches for each model feature, especially since almost all of the hypothetical poses considered are not actually correct. In the worst case, if every match were dubious,

there would be 2^{20} possible M for each H , instead of one. In response to this problem, the actual implementation effectively sets a proximity threshold, beyond which any match is discarded, and occlusion is assumed. This approximation is not necessarily the best alternative. It might be better to try both possible matches in certain cases. The fact that this approximation is made would not otherwise be obvious, and is one of the important insights of the probabilistic perspective.

3.5.4 A Helical Distribution

Even with a vastly reduced set of M to consider, it would take a long time to step through every T in the histogram. If each axis of the T space is divided into n pieces, the number of T to consider will be proportional to n^3 . For each T , the important m must be found. Instead of imagining the process as occurring one bucket at a time, one can think of each of the $\Pr(f_i|T, m)$ for each m as existing throughout the space of all T simultaneously. Instead of stepping through T , then, each of the (for example) $20 \times 50 = 1000$ different matches needs to be considered. For each m , the value of the distribution could be determined at every T . Actually, $\Pr(f_i|T, m)$ is only significant at those places in T where an M which uses m is likely. In fact, at all other places it can be ignored, since the term will not appear in any of the $\Pr(I|T, M)$ products calculated at those values of T .

Therefore, what is of interest is, for each m , the locus of points in the space of T where $\Pr(f_i|T, m)$ is non-zero. For features with a roughly Gaussian error distribution in each of x , y , and θ , $\Pr_e(e_{f_i}|T, m)$ looks like a fuzzy ellipsoid centered at the origin of the space of e_{f_i} . The most likely value for e_{f_i} is zero— of all possible places the image feature might be, the predicted location is most likely. In the space of all f_i , $\Pr(f_i|T, m)$ looks like a fuzzy ellipsoid centered on f_i^* . However, the locus in T where $\Pr(f_i|T, m)$ takes on significant values will be a helical arc. At every T , f_i^* can be found, and given m , $e_{f_i} = f_i - f_i^*$ gives $\Pr_e(e_{f_i})$. The fuzzy ellipsoid can be thought of as just a look-up table, and each T implies a pointer into it. The helical arc, then, is the set of T values which would point near the center of the ellipsoid; they predict image feature locations close to where the image feature actually is.

Figure 3.12 graphically illustrates why this should be a helical arc. If the model is placed in the image such that the model feature and the image feature correspond exactly

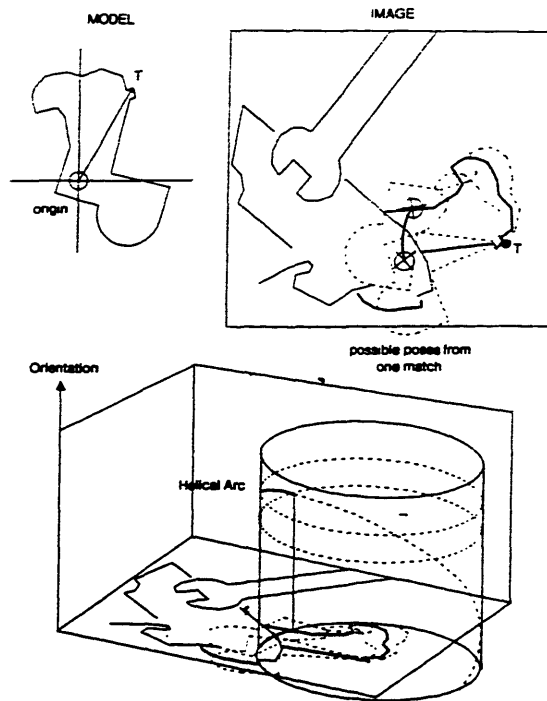


Figure 3.12: The helical distribution for T given a feature match.

in position and orientation (T_0), then $e_{f_i} = 0$. The value of T can be inferred from the location and orientation of the origin of the model in the image. If the position is shifted slightly (T_1), then T shifts slightly in the x and y directions, but remains at the same θ value, and $\Pr(f_i|T, m)$ goes down before many different transforms have been covered. However, if the modeled object is rotated slightly about the feature match (T_2), T shifts proportionally in the θ direction, but it also moves along a large arc in the position directions, traversing a good portion of a helix before $\Pr(f_i|T, m)$ drops off, covering many likely T s. This is because the transform variable θ rotates the model around its origin, not around any particular feature, so the position makes up for it. The radius of the arc is the distance of the model feature from the model origin.

The fuzzy helical arc is just the locus of significant values of $\Pr(f_i|T, m)$. It is actually defined over the entire space of T s. If $\Pr_e()$ is approximated by a rectangular solid instead of a fuzzy ellipsoid, then the helical arc (with a small rectangular cross-section) represents $\Pr(f_i|T, m)$ completely. There are also two other axes in the space of e_{f_i} which have not been discussed. These correspond to the errors in the other feature parameters: type and discriminating value. $\Pr_e(e_{f_i}) = 0$ if the error along the type-axis is non-zero. There is no

chance that a model feature will cause an image feature of a different type. The probability of an error in discriminating value is a useful distribution to consider. It could also be modeled as a Gaussian, or more crudely as a box distribution, just as the other axes are approximated. In the latter case, the presence or absence of the helical arc would be affected by the difference in discriminating value.

3.5.5 Error Distributions for Lines and Circles

The above section assumed a $Pr_e()$ which was a fuzzy ellipsoid. This is a reasonable assumption for point features, such as the tip, peak, and valley features, and for the “end” features of lines and circles if they are not considered in pairs. However, if the entire line or circular arc is considered, a different distribution should be used. When two lines are matched, for instance, the orientation error should be very small if the lines are straight, but the position in the direction of the line is uncertain due to instability in the location of the endpoints along the contour. In addition to the instability, the lengths of the lines may be different due to occlusion, in which case there is a degree of freedom in the match. The two effects can be accounted for by using a $Pr_e()$ which looks like a fuzzy line in the x -error- y -error plane. The resulting locus of non-zero probability in T space is also a fuzzy line, since the coupling between position and orientation does not matter.

Circular arcs are even one step more complicated. In addition to the degree of freedom introduced by arcs of different lengths, there is a degree of ambiguity introduced when arcs of slightly different radii are matched. If the radii are too different, the match is discarded, so the difference is guaranteed to be small. A compromise is to position the arcs on top of one another such that as much of the tighter arc is on one side of the wider arc as it is on the other side of the wider arc. This is done as the shorter arc slides along the longer. The resulting distribution in T space is (coincidentally) a helical arc, but with its center on the center of the wider circle, rather than on a feature. These alternative distributions are used in addition to considering the end features.

The distributions discussed so far are derived analytically, not empirically. Trying to generalize to other kinds of contour feature, the conditional distribution of the image feature error should probably be estimated more carefully for each case. One way to do so would be to assume that the major cause of error comes from location instability along the contour.

In that case, the error distribution could be modeled by matching various points along the contour in the neighborhood of each feature, and recording the pose implied by each. In this way, the particular neighborhood of each feature would indicate the possible errors, rather than a generic analytical distribution. This has not been implemented, but seems promising.

3.6 Pose Array Implementation

Whichever interpretation is preferred, the histogram approach is implemented as follows. Every possible match between image and model features of the same type is considered. From the probabilistic perspective, it was seen that the desired quantity is $\Pr(f_i|T, m)$: the conditional distribution for the image feature, given that it is caused by a particular model feature, and found for all of the transforms which make it significant. This conditional distribution can be approximated by a helical arc in the space of T values, whose presence is conditional on the difference between the discrimination values associated with the features. If the values are close enough, a helical arc of buckets is filled with information pointing to the match. After all matches have been considered, the buckets are searched sequentially, and the ten with the most match pointers are put in an ordered list for later verification.

3.6.1 Using the Discrimination Values

For tips and circles, the curvatures of the image and model features are compared. For peaks and valleys, the second derivative of orientation is used, and for line features length is used. Currently, the magnitude of the difference in the absolute values is compared with the appropriate threshold, and matches which are not close enough are simply discarded. (Line features are a little different: image lines must be no more than a threshold longer than model lines, but they may be shorter). I have found that as many as sixty percent of the matches can be selectively discarded in this way without significantly affecting the presence of important matches at the histogram peak locations, and that in some cases the peaks are more pronounced due to the reduction of extraneous data. It should be worthwhile to weight the surviving matches proportional to the closeness of the discrimination values, but this has not been tried.

3.6.2 Filling the Helical Arc

For each match which survives, the buckets corresponding to significant values of $\Pr(f_i|T, m)$ must be determined, and filled in the pose array. For matches of features which are well defined to be at a point along the contour, the only ambiguity in the implied pose is due to inaccuracies in the feature extraction process. There are no degrees of freedom. Since the position of the image feature is known, and the position of the model feature relative to its (arbitrary) origin is known, the model is well located in the image, at least at the point where the features line up. From the orientation of the image and model contours at the feature, the orientation of the model in the image is also well defined. However, as seen in a probabilistic perspective, small errors in the exact position along the contour may cause angular ambiguity of five or more degrees, which, in turn, causes large errors in the estimate of the position of the model origin in the image. This coupling must be accounted for, or the histogram method will fail. The correct distribution for the pose given the match will reflect the coupling. It will have its highest concentrations of probability along a helical arc, as shown in Figure 3.12. If the rotational and positional parts of the pose were not coupled, the distribution would appear as a section of a cylinder, or if the positional error was not caused by rotation, a rectangular volume of high probability would suffice. But there is a particular angle associated with each position along a circular arc, so the path in three-dimensional pose space is a section of a helix.

If line and circle features were not reduced to their endpoints, a match would produce a different type of distribution, as discussed in Section 3.5.5. Since the features could have different lengths, there is a degree of freedom as, for example, the shorter line slides along the longer. With circles, an added match ambiguity is introduced if the circular arcs have different radii. In this case, the two degrees of freedom make the distribution fill an extended volume of high probability.

To use the conditional distribution in the pose array, it is further approximated as either being zero or constant at each location corresponding to an array element. For each match, at every bucket along a helical arc in the pose histogram, an indication of the match pair is pushed onto a list corresponding to the bucket. No weighting is currently used among the buckets that are filled, even though the distribution might indicate a higher probability at some than others. Also, all other buckets in the histogram are ignored, despite the fact

that the real distribution might indicate a small non-zero probability at some of them.

A more problematic approximation comes from the fact that the helical arc of buckets is actually found by stepping through the angle at a fixed step size, so buckets may be missed along arcs of large radius. Future implementations may attempt a linear approximation of the arc, filling all buckets along lines through the three dimensional space. Until then, the success of the algorithm depends on the fact that strong hypothetical values for T will have many matches, so the peak is likely to still be detectable.

3.6.3 Finding the Peaks in the Histogram

After all matches have been used to fill the pose array (T space) buckets, each bucket contains a list of matches which are likely for that hypothetical transform. $\Pr(T, M|I)$, must still be constructed from these matches. The probabilistic perspective shows that this is proportional to $\Pr(I|T, M)$, the conditional probability for the image given the transform and the likely matches, which can be found from the product of the $\Pr(f_i|T, m)$ over all m in M . While it would be more accurate to estimate values of $\Pr(f_i|T, m)$ for each match, they are currently all approximated by the same constant, one. This would make all the buckets have the same value if it were not for the more significant effect of occlusion. Since the product at each bucket should be over all m in M , there should be as many terms in the product as there are features in the model, typically roughly 20 terms. If only four matches are listed at a particular bucket, the other sixteen are assumed to be occluded. The terms for these matches are estimated to be much smaller: for example, $p_{occ} = 1/2$. Therefore, if there are n_m real matches in a bucket, the probability is equal to $p_{occ}^{20-n_m}$. If the buckets are just to be ordered, they may simply be ranked by n_m .

Because all the information needed to make a more accurate estimate $\Pr(I|H)$ is available at each match, the best poses could be more carefully ordered. Note that the standard Hough transform simply adds numbers in the buckets. The buckets which are local maxima do not contain information about the matches, despite the utility of this information in later stages of recognition. It is also difficult to justify summation at each bucket, though it could perhaps be as useful as the product if the logarithms of probabilities are considered, or if only ranking is to be performed and the probabilities are crudely approximated, as above.

To find the peaks in the array, the buckets are searched sequentially. Whenever a bucket with more than a fixed threshold of matches is found, a local climbing technique is used to find the bucket with local maximum number of matches. This bucket and each of its 26 immediate neighbors are collected, and those array buckets are emptied to prevent them from being found more than once. The sequential search then continues. By including all the neighbors around the peak, bucketing errors due to the approximation of the distribution for each match are partially repaired, at the cost of including more potentially incorrect matches as well. The result of the pose-indexing module is a list of the most likely hypotheses, with a list of the supporting feature matches for each.

3.7 Match Extension by Exploring Feature Neighborhoods

The features found along the contours contain information about their position and order relative to other features which has not been exploited by the modules described in the previous sections. In the first recognition scheme, the pose-indexing module collects matches from the whole image, based only on the pose each match implies, and produces a list of matches with each hypothetical pose. The neighborhood information can be used to recover matches which were not originally indexed to the same pose due to noise or other errors. If a match is correct, the whole sequence of features along the contour around a feature should match with the model, until the contour is occluded. Since this is much more likely for correct matches, neighborhood searching also provides a means for verifying hypotheses. In the second recognition scheme, the same module is used to “extend” all matches by searching their neighborhoods. The extended matches are much more discriminating than single feature matches, and they allow a much better estimate of the pose to be made so that accurate pose-grouping can be accomplished more easily. Because the contours provide an order to the features, the algorithm can be very efficient. Whether it is efficient and successful enough to be used as the first stage in the recognition process is addressed in section 3.10.

3.7.1 The Neck-and-Neck Neighborhood Search Strategy

The pose indexing array uses global spatial information to correlate matches. It takes every match and finds where the origin of the model should be placed in the image, then groups

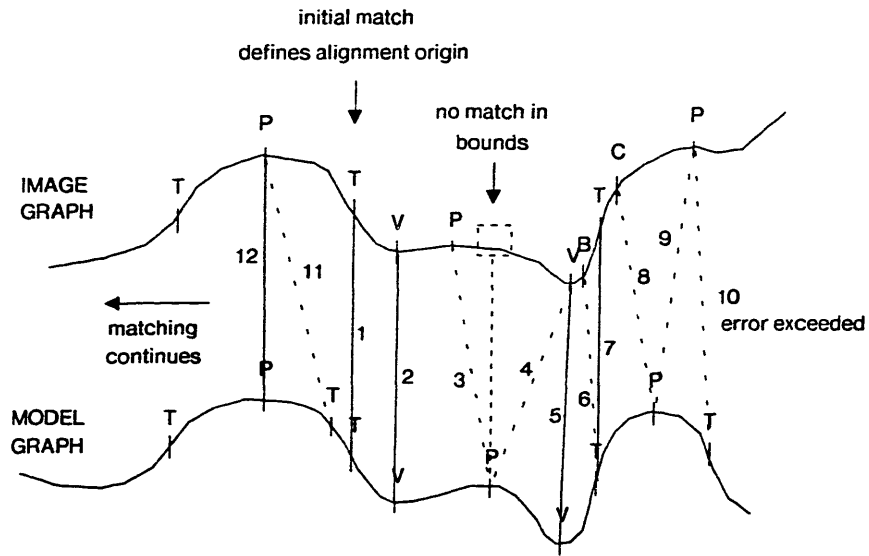


Figure 3.13: The Neck-and-Neck neighborhood matcher.

matches which put the model at the same place. The neighborhood matching module only considers the immediate neighboring features along the model and image contours, to check if they will match as well as the seed match. It is local, and only compares the orientation graphs, not the spatial image contours. To perform the same operation, a human might take two orientation graphs plotted on clear plastic, align the seed match features, and look to see if the bumps and other features in the graphs were also aligned. If the immediate neighbors are similar, the comparison continues away from the seed match in both directions until significant differences are detected. Every pair that is aligned is made part of the extended match.

Figure 3.13 illustrates the algorithm at work. The initial seed match is used to align the arc-lengths and orientations of the two contours, defining a local origin for the comparison. The relative traversal direction is dictated by the feature match, as described in section 3.3. One side of the match is checked first, then the other. Since the features are kept in order along the contour, the next feature and its distance (arc-length) from the seed match can be found along both contours. If their arc-lengths and orientations relative to the local origin are within bounds of each other, and the features are of the same type, the pair

is taken as a new match and the comparison continues. If the differences between these features is too great, an estimate of the error is made by taking the difference between the orientation values at the arc-length of the feature closer to the local origin. The error is accumulated as incorrect matches are made, but is reset to zero if a correct match occurs. After an incorrect match, the next feature along the contour which had the feature closest to the origin is used in the next match. Thus, the features always stay neck-and-neck in the race away from the origin. Matching in one direction ends when the accumulated error grows above a bound, or when the end of a contour is reached.

The neck-and-neck algorithm is very efficient relative to the pose-indexing module, since it does not have to confront the same explosion of possible matches. For a given starting match, the neighborhood matches to test are well ordered— not all combinations need be tried. Also, incorrect seed matches, which are much more numerous, require less time to process than correct seed matches. The overall execution time is kept down even if correct matches perform a significant amount of redundant search. When two similar contours are compared, in the best case, the entire stretch of similarity is found for every seed match in the stretch. This redundancy is not avoided, since different seed matches can find slightly different sets of extended matches due to slightly different initial alignments.

This module is an example of a model-driven search. Rather than considering every possible image feature, the hypothesis directs which feature is to be checked (based on the model). What is left of the search is very small. The difficulty is in efficiently finding initial hypotheses to begin the search. The first algorithm uses pose indexing to find a few good hypotheses. The extension module can then be used as a verification tool, since the number of unique matches it finds is a good indication of the quality of the hypothesis. Match extension does not explicitly interpret the cause of any occlusion, however, so it allows hypotheses which give strong evidence for an one part of an object even if nothing at all is detected in the image where the rest of the object should be. That is, any missing matches are assumed to be due to occlusion, even when no occlusion is possible. This is one of several reasons why a preliminary image parsing module would be very useful, as discussed later.

The second and third algorithms use all possible initial matches as seeds for the match extension module. In this case, the extended matches alone do not determine the best hypothesis. In particular, matches are never extended across separate contours, so evidence

from several contours is not correlated. Further steps are needed to group the extended matches, and they are presented in the next two sections.

3.8 Grouping Poses

The second algorithm takes the extended matches, and then accomplishes the same task as the pose index array accomplishes. It arranges the evidence into corroborating groups. In the pose indexing module, the matches used to load the array are made from weaker, single-feature matches. Noise can affect the hypothesis significantly, and there is little information to discriminate between good and bad matches. The extended features are much more informative. Bad matches are indicated by their relative shortness, and the relatively few surviving matches give fairly good estimates of a hypothetical pose. It is possible for a short match to be correct, but the cost of processing all short matches is prohibitive. Under these conditions, pose indexing is not practical. Instead, pose groups of extended matches are grown. Each extended match is considered sequentially, and the pose is estimated from the set of matches by taking the average difference in orientation at the features, and then aligning the centroid of the image features with the centroid of the model features at the estimated orientation. The pose is sequentially compared with a list of the current pose groups, and if it falls within a rectangular bound of any of the groups, the extended match is merged with the group. Merging consists of checking every feature match in the extended match with every feature match in the group, and discarding those new matches which contain either a model or image feature already in the group. The remaining new matches are added to the group, and the group pose is reevaluated including them. The efficiency of this grouping algorithm could easily be increased by ordering the groups by pose, and by ordering the features within the group by type or location, but neither of these improvements has been implemented.

This method of grouping has a minor drawback: the final groups depend to some degree on the order in which the extended matches are processed. This is because the group pose is used to determine whether or not an extended match should be merged, but the group pose changes during the growth of the group. This can cause bad matches to be included in, or good matches to be excluded from a group while it is small and the group pose estimate is inaccurate. Thus, an otherwise large group may form as two separate groups. The bounds

used for inclusion and the number of extended matches both influence the extent of this effect. Unfortunately, the algorithm is fairly sensitive to the size of the inclusion bounds, just as the size of the pose array buckets affects its grouping capability, but without the same bucket-splitting problems.

The success of the grouping scheme also depends on the accuracy of the pose estimate, both for the extended match and the group. The current estimate (described above) is fast and simple, but does not find the best orientation and position simultaneously. A better estimate can be obtained by solving a system of linear equations for all three variables to minimize the sum of the squares of the distances between matched features, as described by Ayeche and Faugeras [86]. Unfortunately, this method is slower, and requires a fourth variable, scale, in order to be linear. It is not acceptable to ignore the scale solution, since it significantly influences the position values. Despite this, grouping with this method was tried and was successful; only execution speed was prohibitive. It might be useful to use this more accurate estimate in a later stage, after the groups have been formed, to further combine groups. This would reduce the separation effect.

The groups form the final hypotheses of the second algorithm. I consider the grouping module to be data-driven, since it simply groups all matches and does not attempt to use any of the matches to control the search for other matches in the group. However, it is an example of a module which could almost be interpreted as hypothesis-driven. If the first member of each group is taken to be a seed for a hypothesis, then the accretion of the group could be considered to be the development of the hypothesis. In addition, due to the pose adjustment performed as each extended match is added, the grouping process does make some commitment to the first member of each group, and can theoretically make mistakes due to this commitment. But the order of the execution is not affected by the hypotheses; each group does not predict the existence of other extended matches and then search for them in the list of extended matches. Therefore, the module is not driven by the hypotheses.

3.8.1 Other Grouping Schemes

One way to eliminate the dependence on the order is to think of each hypothetical pose as a cube in the space of poses. The dimensions of the cube reflect the accuracy of the

hypothesis. Any cubes that overlap form a group, and the job of the grouping algorithm is to search for the linked chains of cubes. A reasonable way to implement this algorithm is to do group accretion as above, but without forming a single running group pose. Instead, the list of all the members is kept, along with an enveloping box. Each new pose is quickly checked for inclusion in the group envelope, and if it fits then it is sequentially checked against each of the cubes in the list. If the new cube intersects any of these, it is added to the group. Overlapping groups can be melded in a final stage, where the envelopes are again used for an efficient initial check. Note that unlike the first method, where a group pose is constantly being adjusted, the box-growing approach can easily extend groups over large portions of the pose-space. If the space is dense enough, depending on the number of hypothetical poses and the estimated error of each, there may be extraneous links between groups, or possibly even a flooding of the entire space. This method was attempted, but found to extend groups over too large a portion of the pose-space, and so it was not used. It does have potential value, however, in applications where the population and error bounds are more appropriate.

In a slightly different application of grouping, a maximal clique algorithm was developed to combine the extended matches along contours, without regard to pose. (This method would clearly have needed a further stage to relate hypotheses across contours, had it worked). Extended matches which contained any of the same feature matches were combined to form a new extended match. The approach suffered from severe flooding, since no global discrimination was made and poor combinations were allowed to propagate their error. If tighter local or global constraints had been used, the method might have been more successful.

3.9 Predicting Matches from Hypothetical Poses

The second algorithm collects the extended matches into groups by the pose that each match implies, using a data-driven driven approach. The third algorithm uses the same extended matches in a simple hypothesis-driven search for other matches in the image. Each extended match indicates a hypothetical location of the model in the image, and thus indicates where image features should hypothetically be found. A small search is performed around every potential image feature location, and if a reasonable match is found it is added

to the hypothetical mapping and used to reevaluate the pose. This is a typical verification process, demonstrating a commitment to a small set of good hypotheses which are used to focus attention on a few potential matches. It is also typical in that a significant amount of time is devoted to each seed hypothesis, so the number of seeds must be limited at the cost of performing an incomplete search. In the current implementation, extended matches are verified in order by the number of individual matches in the extended match, until ten unique poses have been considered.

Two reorganizations of the feature data are used to increase the efficiency of the verification module. First, the image features are placed in a fifty-by-fifty array corresponding to their position in the image. Each feature is placed in the array cell containing its position, and in the eight neighboring cells. This assures that each cell contains all features within one cell width of any position within the cell. In each array cell, a list of image features is kept for each feature type. For each type, there are usually only three to five features per cell even in crowded areas of the image. To find a predicted match to a model feature, the array is simply indexed and the first image feature of the correct type with position and orientation values within certain bounds of the predicted values is used as a match.

Second, for each model feature a list is included in the model representation of all the other model features ordered by their distance from the feature. The list from the model feature in the initial seed for the extended match is used to define the order of feature verification. Starting with model features close to the seed match, the estimated pose is used to predict the location of the other model features in the image. For each one, the best image feature is found using the method described above. If a good match is available, it is added to the hypothetical mapping of the model into the image. Including the new match, the pose is reestimated using the averaging technique described in Section 3.8. The ordering is vital to the success of the process. The risk of using this or any hypothesis-driven matching scheme is that a narrow commitment to an initial hypothesis will not allow the inclusion of the correct match. In this case, if the pose estimate is not exactly correct, the narrow image search for the matching feature is likely to fail. As discussed in Section 3.5, the error in the position of the matched features is usually very small, and the orientation error is also not particularly large, especially when the entire contour-extended group of matches is used to estimate the pose. However, because of the coupling between orientation and position, even the slightest error in orientation will cause an amplified error in position

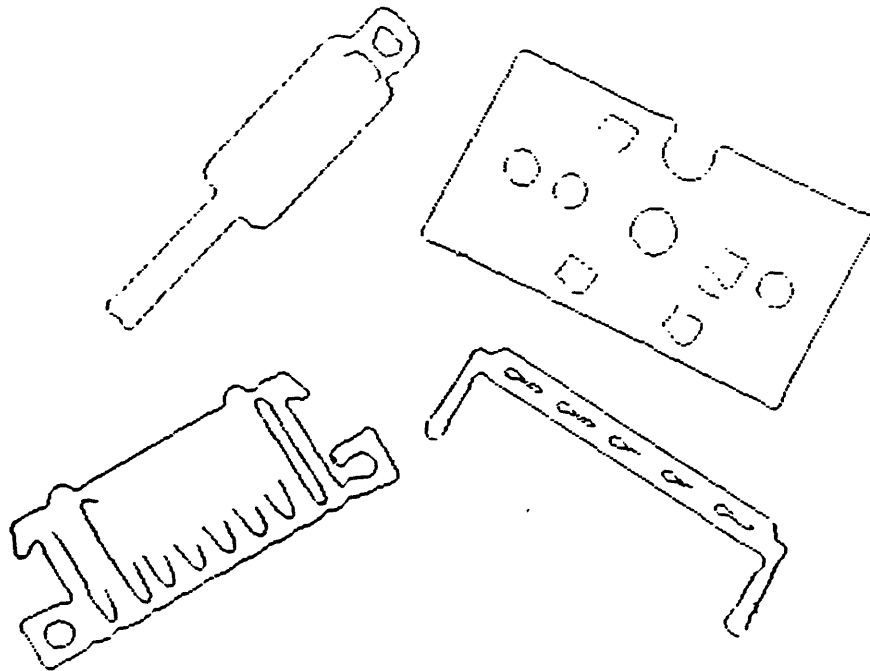


Figure 3.14: The four models used to test the algorithms.

proportional to the distance from the match. Therefore, the close features are found first, allowing greater accuracy in the pose estimation when the further features are considered.

Using this verification module, a mapping can be constructed using virtually no search, yet enforcing relatively tight geometric consistency. However, if the seed matches are not already partially correct, there is no hope that this module will find a correct interpretation of the image.

3.10 Results and Discussion

The three algorithms were developed on a Symbolics 3600 Lisp Machine. To evaluate their performance, the four model objects shown in Figure 3.14 were recognized in ten images. Each image contained all four modeled objects and several other objects, as shown in Figure 3.15. The images are essentially random arrangements of the modeled objects demonstrating varying amounts of occlusion. One of the not-modeled objects closely resembles a modeled object and acts as a decoy. All of the modeled objects have at least one axis of symmetry, and in many cases the rotated version of an object was found in addition to the correct interpretation. The algorithms are not designed to identify flipped-over objects, so single axis symmetries did not have an effect. Table 3.1 indicates the results

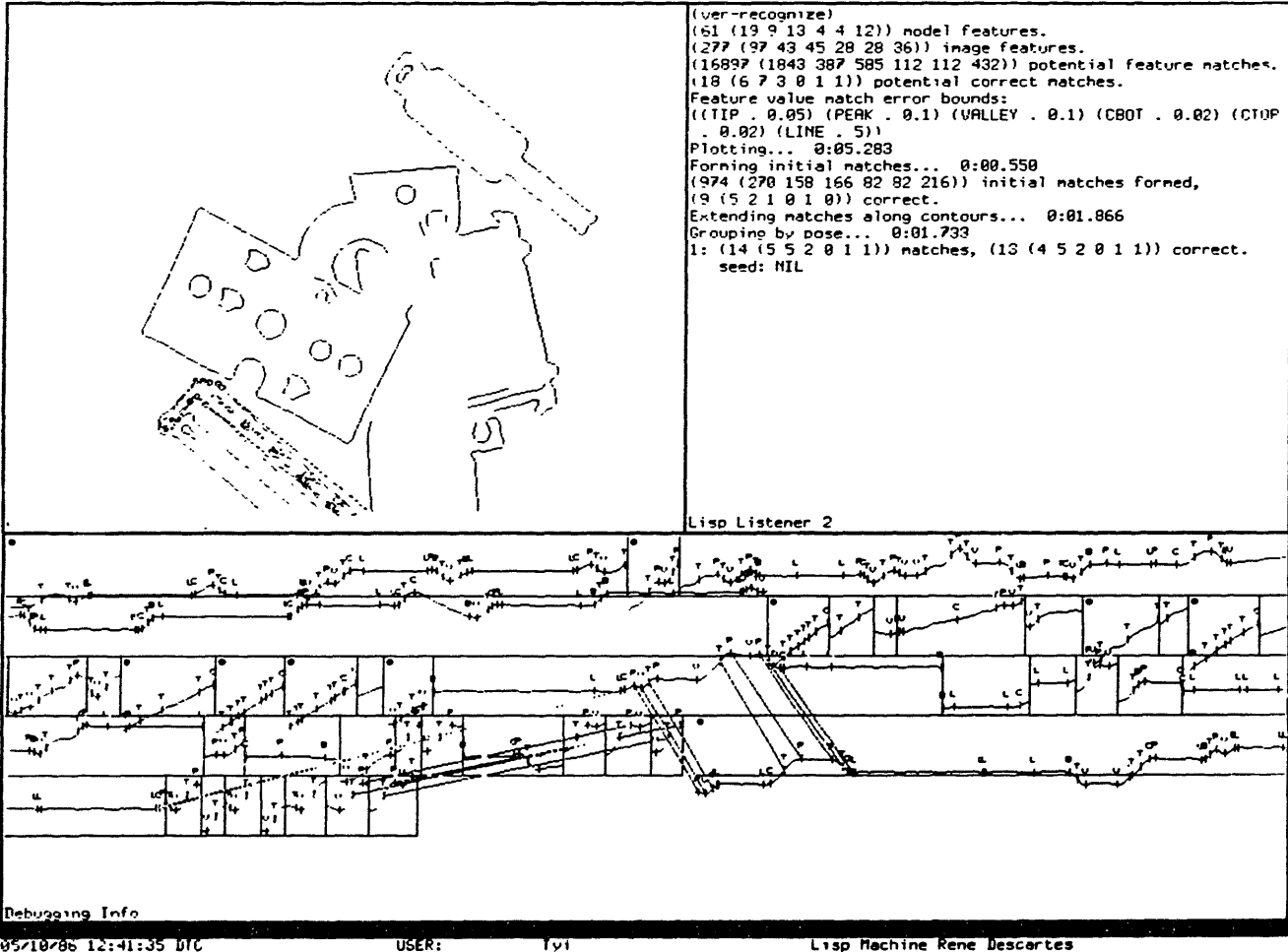


Figure 3.15: The test procedure in action. The fuzzy outline of the clip is the most highly ranked hypothesis of the second algorithm. Small circles indicate the matched features, and lines connect matched points on the orientation graphs.

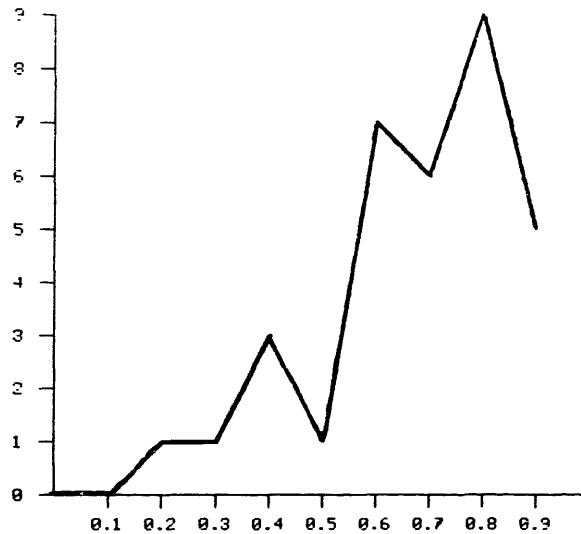


Figure 3.16: A histogram of the visibility of models in the 40 test images. The horizontal axis is the ratio of the number of visible model features to the total number of features in the model. The vertical axis indicates the number of model instances with that visibility.

of the trials, which are described below and discussed in the next section.

To determine the correctness of the matches, the location of each model in each image was defined by hand and matching features were found automatically (item 5 in Table 3.1). In addition, the portions of the model which were visible in each image were found by hand to determine the number of unoccluded model features (item 3, and see Figure 3.16). The fewer the number of visible features, the harder the recognition task. Also, the ratio of visible to matchable model features gives an indication of the repeatability of the features (item 8). In all cases, when the number of features or matches was recorded, it was broken down by feature type in order to track the relative utilities of the different types. Also kept were the total number of model features (item 2), and the total number of image features (item 1). The product of these gives the total possible number of initial matches (item 4). The sum of the products of each feature type gives a more realistic indication of the number of initial matches faced by any algorithm using this feature set (item 4, Total).

The initial matching module eliminates many of these matches using the discrimination values of the individual features. This initial screening serves both to reduce the amount of data (item 12) and to improve the population of correct matches relative to total matches (items 9, 10, and 11). The "Total" column does not equal the weighted sum of the feature type columns for ratios because it was averaged separately, and division is

Average Statistics for 40 Trials, by Feature Type							
	Total	Tip	Peak	Valley	Circle Top	Circle Bot	Line
1. Total Image Features	326	111	58	64	30	30	35
2. Total Model Features	74	28	9.8	11	6.3	6.3	13
3. Visible Matches	visibility varied from 20% to 100% (see histogram)						
4. Possible Matches (1. × 2.)	5149	3097	561	684	186	186	435
5. Correct Matches	23	11	3.5	4.2	1.4	1.5	1.7
6. Initial Matches	1563	597	244	283	116	116	207
7. Correct Initial Matches	13	6.3	1.3	1.8	1.3	1.4	1.0
8. Repeatability (5./3.)	.47	.64	.51	.50	.35	.41	.20
9. Correct Matches per 1000 Matches (5./4.)	1.0	3.6	6.4	6.0	8.9	9.1	3.9
10. Correct per 1000 After Screening (7./6.)	8.6	11.0	5.6	6.7	13.	13.	5.4
11. Improvement (10./9.)	7.6	2.9	1.5	1.3	1.9	1.8	3.5
12. Reduction (6./4.)	.064	.188	.435	.404	.682	.682	.477
13. Reduction of Correct Matches (7./5.)	.53	.52	.34	.36	.71	.75	.38
14. Use of Available Correct Matches							
Pose Indexing	.67	.60	.71	.68	.65	.60	.52
Ext. Mch. Grouping	.66	.53	.71	.68	.68	.65	.59
Mch. Verification	.70	.66	.76	.67	.66	.58	.52

Table 3.1: Values averaged over ten images, four models in each. The total column is averaged separately, not the total of the averages.

	Pose Indexing	Extended Match Grouping	Extended Match Verification
Correct First-Ranked Interpretations out of 40	34 (85%)	37 (93%)	33 (83%)
Percent of Available Correct Matches Found	67%	66%	70%
Failures: Percent Visible, D = Decoy, F = Flipped.	81, 62D, 52, 51F, 33, 12	62D, 51F, 12,	91, 89F, 62D, 51, 36, 24, 12
Minimum Percent Visible with Success	24	24	33
Confidence: Best score Relative to Second Best	1.8	2.0	2.6
Rank of Seed Leading to Best Interpretation	3.6	N/A	2.0
Lowest Rank of a Seed that Lead to Success	69	N/A	10

Table 3.2: A comparison of the three algorithms.

not left-distributive.

For each of the three recognition routines, the top-ranked hypothesis was interpreted as correct if it contained mostly correct matches. The number of correct top-ranked pose hypotheses out of the forty trials reflects the basic success of the algorithm, as shown in Table 3.2. Another measure of each algorithm is its ability to use the available matches, indicated by the ratio of correct matches in the final hypothesis to the total number of correct matches available. In addition, the number of features in the most highly-ranked wrong hypothesis is kept, and the ratio of the number of matches in the first interpretation to the number in the best wrong interpretation is an indication of the confidence of the identification of the best hypothesis. A value close to two means that the second best interpretation was only half as good as the best interpretation. For the few failures, the percentage of the model that was visible is indicated with characters indicating several special cases.

Average Execution Times (seconds)		
Pose Indexing	Extended Match Grouping	Extended Match Verification
Model Construction: 11		
Image Feature Extraction: 22		
Initial Matches: 0.6		
Pose Indexing: 71.0	Match Extension: 3.0	
Finding Peaks: 6.5	Pose Grouping: 4.7	Verification: 1.4
Match Extension: 6.2		

Table 3.3: Execution times for the different recognition modules.

To evaluate the effect of truncating searches by using a heuristic measure of the quality of a partial match, the rank of the seed that eventually produced the top-ranked hypothesis was recorded (for the first and third algorithms only, since they have a final hypothesis-driven stage). The number of matches in the seed and the number of correct matches in the seed were also recorded, indicating the quality of the seed.

Despite the fact that absolute execution times are heavily influenced by programming ability and quirks of particular machine architectures, and are generally not indicative of the utility of an approach to solving a problem, the average execution times for each module of the three algorithms have also been recorded (see Table 3.3). The edge-finding process is excluded from these times.

3.10.1 Discussion

There are several interesting points to notice in the data. One is the importance of the initial match screening. By using the discrimination values the total number of matches is reduced by an order of magnitude. The percentage of correct matches among all possible matches is quite small, so the overwhelming majority of the calculations performed by the first recognition module are on incorrect matches. The efficiency of the recognition system is greatly improved by extracting features with enough descriptive strength to allow effective screening at the level of initial matches. The accuracy of the recognition process may also be improved despite some reduction in the number of correct matches available, due to the

elimination of incorrect information that could, for example, lead to false peaks in the pose indexing histogram. Thus the initial matching module can be said to be performing some degree of recognition.

A significant amount of execution time is spent finding the features in the image, but I do not interpret this as an indication that the features are too complex. The feature extraction procedure is fairly simple, and its efficiency could probably be improved a great deal. As with other low-level vision tasks, a large amount of repetitive calculation lends itself to special-purpose hardware or parallel implementation.

On the other hand, the relatively large execution time of the pose indexing module is much more intrinsic to the method. While pose indexing provides a general means of correlating data from many sources, it must be used properly to be successful. Filling the proper distribution of buckets is a naturally time-consuming task. Also, due to its data-driven nature, it processes all of the input data uniformly, without drawing any conclusions that might be used to reduce the work.

The shorter execution times of the hypothesis-based modules are probably significant. They are mostly due to the much reduced search space inherent in hypothesis-driven methods. However, the execution speed is also affected by the relatively small number of input hypotheses. In particular, if all the expanded matches were verified instead of the ten longest, the module might take one or two orders of magnitude longer to execute. Thus, the cutoff number (ten) has a great influence on the module's speed. In a typical example of the accuracy versus speed trade-off, the cutoff also affects the ability of the recognition system to find the best interpretation of the image. The issue is whether the ranking of the extended matches, based on the number of matches in each, can guarantee that the best interpretation will be among the ten top-ranked hypotheses. In the terms introduced in Section 2.3.2, will the top ten extended matches always include all the hypotheses which meet the acceptance criterion? The answer is, no. The final, accurate acceptance criterion is based on all the matches in a hypothetical mapping. To expect a simpler measure such as the number of matches in one section along a contour to be as selective as the complete criterion is asking too much. The inclusion of the cutoff makes the search incomplete.

But how bad is the problem? The negative effect of the cutoff is to exclude a needed expanded match from consideration by the verification module. In the recognition trials, this would result in an incorrect best hypothesis and no correct interpretations in any of

the other hypotheses. This happened with the third algorithm in four of the forty trials. In at least one of the cases, the model contours were broken into several small parts, such that no single contour extended match would be very long. This is exactly the way in which the extended match count is an inaccurate assessment of the quality of the final interpretation it would spawn. It would seem that the cutoff is too tight and should include more matches. However, it is not clear how much is to be gained by a small increase. There will probably always be some cases in which the largest correct extended match is relatively small, and to include it would mean including a very large number of incorrect matches during every execution of the module. An indication of this phenomenon is that among the successful trials of the verification algorithm, the average ranking of the extended match that led to the correct interpretation was only 2.0. (The lowest rank that led to a successful interpretation was ten because of the cutoff— it does not reflect the lowest ranking that could have led to a success if given the chance). Despite this low average, in ten percent of the trials the potential correct seed was ranked worse than tenth. Thus, doubling the number of hypotheses included might only increase the success rate a small amount. This reflects the weakness of an incomplete search: its dependence on the accuracy of an early evaluation of partial hypotheses.

The same phenomenon can be seen in the first algorithm, where only buckets in the pose indexing array that contain more than four matches are used as seeds to the match extension module. In this case, the measure of quality is also the number of matches, but the matches have been correlated from the entire image and are closer to being a complete mapping. It is conceivable that a reasonable estimate of the acceptance criterion could be made from the matches, but since only the number of matches is used, some error is predictable. This method of applying a cutoff is also different from the first case in that it is less severe, and fixed. The number of matches of the best excluded seed in the first case could float depending on the quality of other seeds. In this second case, the average rank of the most successful seed was also low (3.6), but the lowest-ranked seed leading to a correct interpretation was ranked sixty-ninth. As before, this indicates a large number of very accurate rankings, with a few rare cases in which the ranking was far out of order.

When the algorithms were successful, the confidence level was usually very good. On average, the best interpretation had twice as many matches as the second-best for all of the algorithms. The percentage of available correct matches actually found was also high for

each approach, at close to two-thirds. Surprisingly, the percentage was roughly constant over feature type despite the lower repeatability of line and circle features. When the algorithms failed, the incorrect interpretations were sometimes still quite reasonable—the decoy was more visible than the model in one of the images, and in another image the only features which could disambiguate the rotation of the model were occluded. In one image only twelve percent of the model was visible. Discounting these errors, the three recognition algorithms missed only eight instances of the models in 112 trials, and the second algorithm missed none of 37 trials.

Two of the failures are notably bad. In one case, despite 81% visibility, the pose-indexing method ranked five incorrect interpretations higher than the correct interpretation. This was simply a case of an image with many features which coincidentally produced peaks by supporting poses that fell in the same pose buckets. It is surprising that this did not occur more often since the buckets are fairly large and the helical arc fills many buckets. In the second case, despite 91% visibility because the final hypothesis-driven module was not given the correct seed among the top ten seeds. While the model contours were somewhat disconnected, it seems more like a fluke demonstrating the method's sensitivity to a fixed cutoff.

The success of the second algorithm in particular is partly due to its multi-tiered structure. The algorithm consists of a data-driven initial feature matcher, followed by a hypothesis-driven match extension using connectivity, followed by a data-driven grouping scheme based on hypothetical pose. At each stage, the data were further refined but not over-refined. The only selective thresholding was in the initial match discrimination. I do not find it surprising that the most accurate algorithm is a compromise between the two approaches.

Chapter 4

A Review of Related Research

To place this thesis in context with other work and to further explore the concepts presented in the earlier chapters, this chapter presents condensed versions of four recognition approaches developed by other researchers. Each algorithm is described briefly, and discussed in comparison with other methods and this thesis. The discussion covers feature choice and extraction, object representation, computational efficiency, module complexity trade-offs, and particularly flow of control (data- versus hypothesis-driven). I attempt to find the strengths and weaknesses of each method, though actual experimentation is needed to carefully determine which kinds of images will cause an algorithm to miss a visible object, for example. Though most of the approaches were developed for a limited domain similar to mine, I also consider the ways in which the algorithms rely on the assumptions of their domains, and the aspects of the algorithms which might be successfully extended beyond their domains.

There is considerable research focused on other aspects or domains of recognition. In particular, algorithms for recognizing three-dimensional (3D) objects have been developed. Most of these systems begin with 3D data, however, rather than a 2D image. This avoids the significant problem of inverting the projection of the world from three to two dimensions. These 3D-to-3D systems bear a strong resemblance to the 2D-to-2D systems discussed below. It seems generally to be the case that this extension is feasible, with a few subtle but important differences. The extension of 2D-to-2D systems to 2D-to-3D is much more complicated, perhaps in a fundamental way. The few recognition schemes that work in this domain have achieved only limited success, and tend to take different approaches from

those presented below. While the work is quite interesting, it is not reviewed in this thesis.

4.1 Bolles, Cain: Local-Feature-Focus

One of the better-known recognition systems for the 2D domain was created by Bolles and Cain [82], dubbed the Local-Feature-Focus method. Their work was motivated by the exponential growth of the execution times of simple matching schemes in the number of features. This led them to organize model features into pre-computed clusters around focus features. The clusters are used to predetermine efficient search strategies for each focus feature, and also to limit the search to a local neighborhood. The recognition module then must only examine the matches within the cluster when forming initial hypotheses, and does not compare matches of model features from different clusters. This reduces the number of matches significantly, but also removes the ability to correlate information globally during hypothesis formation. Their implementation also emphasizes the detection and exploitation of various symmetries, and also of similarities between features of different models in a small model library. However, they deemphasize feature definition and extraction based on the assumption that those modules can be built to fit the task. The features they actually use (corners and holes) are very specific to a certain class of shapes, and it is not obvious that the recognition method can be successfully generalized to more complicated object domains, where their notion of a feature is less well defined. Hypotheses are then verified, taking advantage of the fact that the images are made under a special lighting conditions so that objects are always black and background is white. Despite the use of binary images and special-purpose features, the recognition approach is interesting and potentially modifiable to more general domains.

Perhaps the major strength of this system is its pre-computation of local match extension strategies for every distinct feature in a small library of object models. Models are developed from images of isolated objects, and the focus features with their local feature clusters are automatically chosen. First, all distinguishable feature instances are found by searching through the whole model library for similar features. All quarter-inch holes, for example, from within one model or between models, are grouped, and considered together to be a distinctive feature. It is not obvious that a typical library will have many duplications, but if it does then dealing with them explicitly will increase the speed and accuracy of

the recognition algorithm. Each distinctive feature is developed as a focus feature. The smallest cluster of nearby features which will accurately distinguish every instance of that feature in the model library is found, with some redundancy included. Thus, when a focus feature is hypothetically matched in the image, the features in the cluster can be used to tell exactly which model feature it is. This is essentially a pre-determined search strategy for every distinct model feature.

In addition, the system performs a great deal of work searching for symmetries of various kinds among the model library to determine even more efficient clusters. It seems to me that this work is excessive unless a highly symmetric library is expected. In fact, any particular feature, such as a corner of angle forty degrees, which is unique within estimated error bounds would not need a cluster at all, if it were not for two other concerns. First, in order to be more robust to occlusion, a redundancy level can be set, defining the minimum number of features included in the cluster which will uniquely identify the focus feature. If a redundancy level of three is specified, then any two of the cluster features can be occluded or noisy and the focus feature will still be identified correctly, no matter which model it belongs to. Second, the clusters are always made large enough that the model pose can be determined from the cluster matches. The focus features are then ranked by size, with the smaller clusters considered to be better since larger clusters take considerably more time to match during recognition. If most of the model features are unique, then there will be many clusters as small as the redundancy level.

Image features are first sifted by eliminating any which are not within the error bounds of any model features. For example, if there are no holes with approximately a half-inch radius in the model library, then a half-inch hole in the image will be ignored. If there are any similar model features, then there should only be one focus feature which is similar, since similar model features are pre-grouped. Image features matching the best focus features are considered first. The initial match defines the list of cluster features to look for in the local neighborhood. All possible local matches are found, in which the image feature is similar to a cluster feature, and is located at a similar distance from the focus feature.

The problem is then to decide which of these local matches are correct. The correct matches are defined to be the largest group of geometrically compatible matches which can be found. The largest group is found using a maximal-clique algorithm. First, each of the potential matches is made into a node in a graph. Pairwise constraints on the matches are

then applied for each possible pair of matches, that is, for each possible arc in the graph. A pair of matches is compatible if they are one-to-one (e.g. two image features cannot both be matched to the same model feature), belong to the same model object, have similar separation in the image as in the model, and if the orientation of the image features with respect to the line between the features is similar to the model. If all of these constraints are met, an arc is placed between the matches in the graph. (Constraints very similar to these are even more thoroughly exploited by Grimson and Lozano-Pérez [84,85], as shown in Section 4.2). The largest group of matches for which every match has an arc to every other match in the group is taken as the correct interpretation. From these matches, an initial hypothesis is made about the pose of a model in the image. In addition, mirror symmetry is explicitly included in the formation of initial hypotheses. Note that the constraints listed above will allow a mirror-symmetric match to be made. Any three of the cluster features are then used to determine which symmetry has been found, and this is considered when the hypothetical pose is calculated.

Each initial hypothesis is immediately verified so that if the hypothesis is good, the image features can be marked as “explained”, and no future processing will be done to them. While the “worse” focus features can be used if needed, the algorithm stops if the image has been exhausted. Therefore, not all focus features are tried in determining which are best. If a mediocre hypothesis is considered to be good enough (as it might, since occlusion is always possible), then those image features will not be available later to the correct hypothesis. This kind of consideration can significantly increase execution times, since it may be very difficult to reliably determine when enough hypotheses have been tried that the best has already been found.

Verification proceeds in two stages. First, matches to all model features at the hypothetical pose in the image are sought. The pose is refined based on these matches. Then, a polygonal approximation to the model boundary is placed in the image, and the image is checked for the proper light-to-dark transitions along the boundary segments. This form of verification relies very strongly on the fact that special backlighting is used in the image formation, so that the image is black where objects are and white elsewhere. Determining this information with grey-level images is a difficult task which deserves research. Places along the contour which are black on both sides are assumed to be caused by occlusion. All white areas and transitions in the wrong direction are taken as negative evidence, however.

This kind of application of polygonal contour approximation is valid, since its error is only the distance between the actual contour and the polygon. Many other systems use line segments as features, thus relying on a repeatable assignment of segments to the contour. As is discussed below, it can be very difficult to achieve such a canonical segmentation.

Bolles and Cain state that their recognition process is designed based on the assumption that verification is expensive, and so initial hypothesis formation is worth spending time on in order to sufficiently reduce the number of initial hypotheses. It might have been useful to concentrate on finding a more efficient verification module. However, from my perspective, it seems that cluster growing is already a form of model-driven search, even if it does not explicitly verify a hypothetical pose. The initial match of an image feature to a focus feature begins a model-driven search in the local neighborhood of the image feature for those features which are similar to the cluster features. Once they are found, the process returns to a data-driven mode as the maximal-clique is determined. The search for cluster matches is, admittedly, not guided by a hypothetical pose, but only by a hypothetical positional match. It is curious that no use is made of a hypothetical orientation at this stage, which could be deduced from the orientations of the image and model features. Holes have no orientation, but matched corner features could be used to predict exactly where the cluster features should be found, and not just how far away they are. In any case, the cluster match search is both guided and limited by a hypothetical initial match and model knowledge, and so is hypothesis-driven.

4.2 Grimson, Lozano-Pérez: Tree Pruning with Constraints

Grimson and Lozano-Pérez [84,85] attack the search space problem directly. They represent the set of all model-to-image matches in an “interpretation tree”, and then use geometric constraints on pairs of matches to prune large branches of the tree. The tree-pruning process is a good example of a data-driven stage, which is followed by a hypothesis-driven stage. Their algorithm is somewhat general, in that it is designed to work for several domains of input data: two-dimensional grey-scale images, three-dimensional range images, and tactile data. In each case, the input consists of sparse samplings of the object’s surface which indicate both the position and the direction of the surface normal. Models consist of polyhedral approximations to the surface and pre-calculated ranges for the values of the constraints

for every different pair of surfaces. In the implementation for the two-dimensional domain, surface samples are in the form of connected line-segments approximating the image contours, and the model is also polygonal. (Special methods for recognizing a library of models are not considered). One advantage of data-driven recognition approaches is that they can guarantee that every possible mapping of the model features into the image has been considered, since they never rely on the accuracy of early hypotheses. However, the set of all mappings is very large, so data-driven methods are often slower. The success of their approach hinges on being able to find a set of constraints which can be applied quickly, but is also powerful enough to eliminate large portions of the tree. The constraints work well, but to include the possibility of occlusion, several heuristic modules are needed to further reduce the search. These modules invalidate the guarantee of a complete search of all consistent mappings. The gain in speed is large and the cost in error may be acceptably small. This kind of trade-off is one of the main issues in this thesis, and is discussed below and in Section 4.5.

Edges are found in the image using the DOG edge detector [Marr, Hildreth 80], which is described in Section 3.1. The continuous contours are then approximated by connected straight line segments. These are the features, which are matched by each branch of the interpretation tree. While a polygon may be used to closely approximate a curve, and to reduce the amount of data needed to represent it, there are several significant drawbacks to using line segments as features. First, line segments do not make anything about the contour more explicit—they do not perform any abstraction of the data. They encode the contour, reducing the amount of data needed to represent it (with some loss), but without interpreting the contour in any way. This means that the search is not focused by further abstraction to particular parts of the contour. However, simply abstracting the contours from the image may provide enough of a focus. Second, line segment features are not repeatable except on polygonal objects. The particular place where one segment ends and another begins is not defined in such a way that the same contour will have the same features using a different starting point. The segmentation is also very sensitive to noise and occlusion. Hence, a one-to-one matching of line segments between the model and the image cannot be expected by any means. In order to use line segments as features, therefore, the matching algorithm must anticipate faulty segmentation. In this algorithm, segmentation errors are accounted for by using them to determine the acceptable ranges

of values for the set of geometric measurements between matches. The ranges are used to constrain the set of compatible match pairs.

It is possible to place constraints on individual matches if the features are distinctive enough, but simple line segments do not have much character. Each line segment feature has four degrees of freedom, which can be interpreted as length, orientation, and the x and y positions of the midpoint. Three of the degrees are needed to define the transform of the model into the image. Only the fourth, length, is available for discriminating whether the features are similar. Thus, the only constraint on individual matches is that an image segment cannot be matched to a model segment which is more than some error bound shorter than it. Shorter image segment matches are allowed, since occlusion is possible. This constraint is employed under the dubious assumption that the segments can be matched one-to-one, and that a short line segment in the model cannot cause a longer one in the image.

More can be said about pairs of matches, since two independent segments have eight degrees of freedom. After three are used for the alignment and the two lengths are used to constrain the individual matches, there are three degrees left with which to discriminate. One is the difference between the segment orientations. If this value is outside the range of possible model values for the two matched features, the matches are inconsistent. The other two degrees of freedom can be thought of as the projections of the distance between the segment midpoints onto each of the segments. These are used to restrict the range of possible positions of the image midpoints along each of the model segments. If these ranges disappear, then the two matches are inconsistent. How heavily the determination of the ranges is influenced by the irreproducibility of line segments is not clear. It seems that segmentation errors are approximated as constant bounds on the error of the distance between segment midpoints, and on angular error.

The constraints are used to eliminate image/model mappings which include any inconsistent pairs of matches, in the following way. The top node of a tree is associated with an image feature. Each branch from the node represents a match of the feature with a different model feature. The next level represents the ways to match a second image feature with all different model features. At this level, the constraints can be applied to each node to determine if that combination of the two matches is consistent. If not, then the node and all of the branches below it can be pruned out of the tree. At the third level, two more

consistency checks can be made for each node, one check between the first and third level, and one more between the second and third level. At level n , there will be $n - 1$ sets of constraints to apply, since there will be $n - 1$ new pairs of matches in the interpretation. It might seem more natural to organize the tree in the opposite way, finding image matches for each model feature at each level. But this would make a broader tree, and since more constraints are applied as the search goes deeper, the tree would not be pruned as effectively. Using this method, the tree is pruned quite efficiently, and the correct interpretation can be found quickly. However, only images of the isolated model can be matched this way.

Consider the broader tree mentioned above, where each level finds a match for a model feature instead of an image feature. Occlusion can be taken into account by adding a special branch from each node which indicates that the model feature does not match with any image feature. As mentioned earlier, the addition of such a relatively likely branch at every node is not just an incremental increase in the size of the task. A tree which without occlusion might only have one consistent path will now have 2^{20} paths (for the example of twenty model features), since the null branch is consistent with every match. In the actual deep tree, the effect of occlusion is less obvious. It will cause branches corresponding to some model features to never be taken, at any level. But since not all image features are due to the modeled object, it must also be possible to indicate that an image feature is not matched to the model. Again, this is done by adding a null branch at every node, and the addition of a branch which is consistent with every match dramatically decreases the pruning power of the constraints.

Accounting for occlusion increases the search time enough that other methods must be introduced. A common way to improve search times is to try to perform a "best-first" search, and then to ignore the rest of the search after a good enough match has been found. Ignoring the rest of the search space relies on having an accurate way of ordering the search. But, as discussed in Section 2.3.2, if this could be done perfectly, no search would be needed. The whole point of the search is to find the best interpretation. Nonetheless, if a very good interpretation is found, there may be no need to continue the search, and it may be possible to arrange for the more likely matches to be checked first. To this end, several additions are made. First, the longer image features are put at the top of the tree, since they contain more information about the image. Also, the tree is traversed depth first, so full interpretations are made as soon as is possible. Second, a measure of

the quality of a partial interpretation is defined as being the total length of the model segments matched to real image segments. Interpretations which can never be better than the current best interpretation are discarded. Thus, the best interpretation according to the quality measure is still guaranteed, but, as the authors point out, there is no assurance that the quality measure accurately indicates the most correct interpretation. Third, the search makes a further commitment to long interpretations by assuming that any image features which have been matched to a model feature in a long interpretation cannot be excluded from other interpretations by making a null match with that image feature. This short cut seems very difficult to justify. It is difficult to predict how much these measures will improve the speed of the algorithm, and how much they will reduce the accuracy. If the complete search were to be made, accounting for occlusion, it is difficult to imagine that any reasonable interpretations would be missed. Since some visible instances of the model are in fact missed, it is probably due to the extra pruning. Also, multiple instances of the model may not be found, since any interpretation worse than the best can be discarded.

Even with these measures, the size of the tree grows so quickly with the number of model and image features that a further step must be taken. To reduce the number of features in the tree, a low-resolution Hough transform is implemented which groups the matches by hypothetical pose. Pose resolution is not important because the Hough buckets are only being used to group the matches; careful pose estimation is made later. Thus, the pose-space can be divided into large buckets to reduce the possibility of incorrect grouping due to the considerable inaccuracy of the pose implied by a single match. The buckets are inspected in overlapping groups, and the matches in the most full bucket groups are used to form small interpretation trees. The Hough grouping serves both to reduce the number of features in each tree and to provide a means of performing a best-first search. By using the hypotheses with the most support first, it is likely that the best interpretations can be found sooner.

Each sufficiently long interpretation is verified by finding the pose which best fits the matches, and then rechecking each match to make sure that it is within an error margin of being correct. Interpretations which were pairwise consistent are not necessarily globally consistent, and these are discarded.

This recognition system was initially designed to perform a complete search over the space of all possible matches for consistent interpretations. In its purest form, there is no

ordering of the search, and it is very slow. With the inclusion of several speed improvements, the tree search is “best”-first to some extent, and is no longer complete by the same consistency criteria since it makes some commitment to “good” interpretations. It is possible to add one further significant short-cut. That is to define a quality threshold, and to end the search whenever an interpretation is found which surpasses it. This technique has been used in many other algorithms without acknowledgment of its potential error. To determine the effect of truncating the search, trading speed for accuracy, Grimson and Lozano-Pérez compared the number of tree nodes traversed without a quality threshold to the number traversed with a quality threshold of ten matches (in an image of twenty features). The results indicated a speed reduction from sixty to ninety-five percent, but in twenty to forty percent of the cases, the early commitment was incorrect. Results varied quite a bit with the accuracy of the data, which was simulated, and with the shape of the object. But the comparison demonstrated a clear trade-off between accuracy and speed.

4.3 Ayeche, Faugeras: HYPER

Ayeche and Faugeras at INRIA, in France, developed a recognition system called HYPER [Ayeche, Faugeras 86] for the two-dimensional domain including scaling. The system begins with a two-dimensional image of a two-dimensional world, in which special lighting is not required and objects may occlude one another. An additional goal is to allow modeled objects to appear at any scale in the image, thus adding a degree of freedom to the transformation between model and image. The pose space is then four-dimensional, with two axes for position, one for rotation, and one for scale. They, too, use connected line segments as features and to represent the model and image, and suffer from some of the related problems presented above. The general form of the system is to make a limited number of initial hypotheses from “privileged” segments, and then to develop, or verify, the hypotheses, including adjustments as supporting evidence is found. The adjustment process is implemented as a Kalman filter, which updates the predicted pose of the model to minimize the sum of the squares of the match errors. Performance seems fairly good, and the method has several strengths, as well as some questionable aspects.

Edges are found in the image using the difference-of-Gaussians method, with a flexible threshold designed to preserve connectivity along weak portions of otherwise strong edges.

A binary image (formed under special lighting conditions) can alternatively be used, and is in fact required for certain complicated objects, and for objects with reflective surfaces which may cause highlights. The contours are fit to polynomial curves, and then approximated by connected straight line segments. Problems with this kind of segmentation are mentioned above, in section 4.2. In particular, to match an image feature, the matcher must effectively decode the segment, and return to the level of the contour to find if the model feature could have been derived from the contour. If this is not done, and the segmentation is assumed to be canonical, several problems will occur: initial hypotheses made from a single match will contain significant error, supporting matches will require large error bounds, and the mapping of features will not be one-to-one. HYPER does suffer from these problems during the formation of initial hypotheses, but he uses a different matching scheme during verification which successfully reduces the effect of segmentation being non-canonical.

The formation of initial hypotheses is an important step. Correct hypotheses must not be discarded, but a large number of hypotheses will take a long time to verify. The approach taken by Ayeche and Faugeras is to define "privileged" features, the ten longest segments in the model, and to only consider these features when making initial hypotheses. This seems like a risky reduction. It essentially relies on the system being so reliable that it will still be successful even if fifty to ninety percent of the model is initially ignored just to speed the search. The claim is made that it is very rare for all ten of the longest segments to be simultaneously occluded. This may be the case, but longer segments are more likely to be partially occluded, and it is not clear that the rest of the recognition process is reliable enough to consistently be successful starting with only one or two correct matches. In addition, there are many shapes which would have to be approximated by many short segments, causing the longest to be almost arbitrary, and possibly clustered in one portion of the model. Under these circumstances, the number of visible privileged segments could be severely reduced. It is quite attractive to use best-first search methods, but accurately finding the best starting point is important and not always easy, and the method will not degrade gracefully unless the search is continued after the best matches fail. Execution times may be forced to increase significantly in order to gain this robustness, but it is hoped that the extra work is only required when a difficult image is encountered. HYPER may truncate the best-first search too early, considering the somewhat arbitrary choice of "best" features, and the need for some redundancy.

From an initial pairing of a model feature with an image feature, a crude estimate can be made of all four degrees of freedom of the pose. Initial matches are further screened by eliminating those matches which predict a scale factor outside of the *a priori* bounds, if any are given, and also by a special first stage of verification. In this first stage, the angle of a segment with one of its two immediate neighbors is used, as an auxiliary value associated with the feature, to rule out many initial matches. This seems to be a much better way to reduce the number of hypotheses as long as the accuracy of the segment angles is good enough. However, the accuracy will be reduced by the contour segmentation problem.

Hypotheses are developed by attempting to continue segment matches along the connected polygons in the model and image. The extension grows out from the initial match, alternating between the two sides of the segment on the polygon. For each model segment, a match is tried with the corresponding next image segment, and no skipping is considered. Thus, inconsistent segmentation problems will propagate without correction. The segmentation error is reduced by the method of matching: rather than trying to line up the midpoints of the segments, only the distance from the model midpoint to anywhere along an infinite line containing the image segment is used. Thus, segments can be misplaced along the contour to some degree as long as the orientation of the contour is not changing quickly.

The quality of each potential new match is evaluated based on the difference between the image segment and the hypothetical position of the model segment using the pose developed so far. An arbitrary measure is used, involving a weighted sum of limited normalized differences of angle and length, and the distance between the midpoints. The orientation error is emphasized, perhaps because the position and scale estimates are less accurate. If the measure is below a relatively high threshold, the segments are matched. If matched, the new match is used to adjust the hypothetical pose, using a Kalman filter, to minimize the sum of the squares of the distances between model midpoints and image segment lines (as mentioned above). This update is particularly good in that it solves for the transform (pose) values simultaneously, and with a reduced though still significant amount of computation. After each unsuccessful match, the total amount of model perimeter that has been eliminated is compared to the smallest total amount missed so far by previous hypotheses. If it is greater, the current hypothesis is rejected, since it can never be better than the one already found. This eliminates unnecessary work on bad hypotheses, but also eliminates

the discovery of the second best hypothesis, which may be needed if there is more than one modeled object in the image, or if the best hypothesis turns out to be incorrect through later verification. A final verification stage is applied to the best hypothesis by repeating the matching procedure, but starting with the final estimate of the transform. This can eliminate bad matches or introduce good ones which were incorrectly classified the first time, when only a few matches had been made and the pose was inaccurate.

The development of the hypothesis has several strengths. It begins near the initial match so that errors in the initial estimate of the pose have the least consequence, and evaluation of the error of matches far from the initial match, where orientation errors cause amplified errors in position, has the benefit of many matches. The hypotheses are effectively updated as they are verified, which significantly improves the likelihood that a single correct match will lead to a correct complete hypothesis. Also, the overall method switches early from a searching stage, which considers many initial matches and produces a (perhaps drastically) limited number of hypotheses, to an efficient verification stage, which involves no search since it relies (perhaps too strongly) on the connectedness of the segments to order the matching. However, there is one severe apparent restriction which has not yet been mentioned. That is, that information from contours which are disconnected, either by occlusion or in the model, are never combined. Since the match extending technique relies on the ordered connection of segments, occlusions which perturb the object boundary on both sides of the initial match will prevent the match from being extended to other parts of the object. If this is the case, it would significantly restrict the maximum amount of contour which could support a hypothesis, despite a large percentage of the object perimeter being visible.

The published results do not seem to suffer much from this problem, and are indeed quite good, both in recognition success despite occlusion, and in having very short execution times. Most of the images were taken under good enough lighting conditions to process them as binary images, but they were not completely in focus, and the objects were not quite parallel to the image plane, demonstrating some robustness to image imperfections.

4.4 Turney, Mudge, Voltz: Feature Saliency

A data-driven recognition system developed by Turney, Mudge, and Voltz emphasizes the ability to distinguish image objects among a library of models. They introduce a definition of feature “saliency” that indicates which features are most distinctive. Features that are more salient are weighted more heavily in the recognition process. Coincidentally, their system uses modules that are superficially similar to mine. Their images are binary, but the curved contours are represented in orientation graphs, and no line-segment approximation is made. However, no distinctive shapes on the contour are identified. Instead, local sections of the contour are used as features. As in one of my algorithms, all possible feature matches are grouped using Hough transform techniques. Because of the type of feature and the matching technique, this is a very computationally expensive approach. Furthermore, since each match is only used to fill one bucket, and no verification stage is used, the method is error-prone and sensitive to image noise. Its success depends on the clarity of binary images, and on a large number of feature matches which help to make up for the inaccuracy of the Hough vote from each one. Occlusion is allowed, but because of the strong emphasis on salient features, if those sections of the model are obscured, the object will probably not be recognized.

Because of the use of binary images, edge finding is simple and accurate. A third-order interpolating polynomial is fitted to the chain of edge pixels to produce the orientation graph. This seems to be a slightly more accurate method than the one described in section 3.2, and also more time consuming. Rather than performing the further abstraction of finding notable points along the contour that can be used to guide the comparison of two graphs, the graphs are used directly. While it contains the shape information isolated from the position and rotation of the object in the image, the orientation graph makes nothing about the shape explicit. As a feature, the graph as a whole is not local, and so would never match completely with an image if the object were occluded. Therefore, the graph is divided into overlapping segments of some fixed length, and each of the segments is taken as a separate feature. In their paper, the segments are twenty pixels long, and are centered at every other pixel. Thus, there are many unfocused features to process, rather than a few that capture the important parts of the orientation graph.

In order to determine saliency, before recognition begins, every model feature is com-

pared with every other feature in the model library. This is done by performing a Hough transform for each feature with each of the other models. For every feature match, a measure of the closeness of the match is symbolically added to an array bucket corresponding to the model pose needed to align the features. Any peaks in the pose-array indicate a similarity between the feature and some other part of the model library. A weighting factor is assigned to each feature, and is used to scale the contribution of the feature to the Hough buckets. By solving a very large equation, the values of the feature weights are found such that the sum of the squares of all the Hough bucket values is minimized. Features which contribute to peaks will have small weights, while relatively unique features will be heavily weighted. The weight is taken as a measure of the “saliency” of the feature. This method of determining the saliency tends to polarize the values—portions of the orientation graph receive maximum weighting, while the rest has a saliency value near zero. A smoother assignment would help a great deal with the system’s robustness to the occlusion of salient features, but this is not done.

The recognition process consists of taking every overlapping section of every orientation graph in the model library and aligning its midpoint sequentially with every point in the image orientation graph. For each comparison, the model feature is shifted so that its average orientation aligns with that of the image feature. The sum of the squares of the differences in orientation between the model and image graphs at every pixel is used to evaluate the match quality. As mentioned in section 3.2, the orientation graph obscures position, since it is a derivative of it. Thus, the differences between portions of the contour are not best found in the orientation graphs, but should be measured directly in the image. In any case, the bucket corresponding to the model pose such that the model and image features are aligned is incremented by an amount proportional to the product of the match quality and the saliency of the model feature. Only one bucket is affected, despite the fact that the alignment estimate may easily contain enough error to be incorrect by several buckets. This is especially true since the bucket size is as small as the pixel size.

It is no surprise that the process takes several CPU minutes. It is a complete search in the sense that every match is explicitly tested, and no hypotheses are formed or assumed until the entire process is done. However, the combination of matches relies on having the matches fall in the same buckets. If “falling in the same bucket” is taken as a criterion for acceptance, then the method is indeed complete. This criterion is not at all robust to

noise in the image, or to slight differences between the model and image objects. Since no verification stage is used, the final interpretation is determined from the largest intersection of the lowest level feature alignments in the Hough buckets.

The only apparent strength of the system is its ability to distinguish very similar objects, such as two keys that differ only in their pin patterns, if the salient features are visible. However, if the pin patterns are partly occluded, no matter how visible the rest of the key may be, it will not be recognized as a key of either type. Furthermore, I believe it is important to *ignore* fine distinctions until the general class of object is identified, in which case it is not difficult to do a small amount of work to refine the identification. This system does not generalize well to more realistic domains, due to its stringent rigidity and image clarity requirements, and its lack of abstraction.

4.5 Discussion

In the preceding sections, I have reviewed the Local-Feature-Focus (LFF) system, the Tree Pruning system, the HYPER system, and the Saliency system. The reviews contain various comments regarding domain definition, feature choice, search strategy, robustness to perturbations in the image, and extensibility beyond the intended domain. In this section, I present a more comparative discussion of the same issues, including comparisons with the three systems I have developed— the Pose Indexing method, the Extended Match Grouping method, and the Extended Match Verification method.

4.5.1 Domain Definition and Dependence

I chose these four recognition systems because they were designed for domains similar to the one for which my algorithms were designed. There are some differences, however, and in some cases domain restrictions have been exploited by the algorithms, and thus have had an influence on their design. All of the systems were designed to recognize modeled objects in a two-dimensional image of a two-dimensional world, in which the shape of an object remains constant. Also, they were designed to succeed despite partial occlusion of the modeled objects. Each system extracted edges from the image, and used only the shape of the edges for identification, making the systems geometrically biased. (Some systems used the binary value of the image to supplement the edges during verification).

It is unfortunate that the type of image (binary or grey) and the required lighting conditions have such a strong effect on the accuracy of edge determination. Both the LFF system and the Saliency system work exclusively from binary images, in which edges are easily found and the interior of objects can be distinguished from the background. The HYPER system sometimes requires images in which edges can be found by simple thresholding; that is, the objects are painted black and placed against a white background. Under these conditions, with the extra edge accuracy, strict rigid geometry constraints can be exploited that will fail under natural lighting conditions. In the LFF system, which simply assumes that useful features can be extracted from the image to suit the application, there are at least error statistics built in that define, for example, the range of distances to search for cluster features. If grey images were used instead, with the appropriate statistics the process would be slowed, but it would not fail unless enough features were omitted in the image that the cluster search was insufficient to identify the cluster. However, the powerful verification stage depends on being able to make object, background distinctions that are much more difficult to make in a grey image. The Saliency method is very sensitive to the kind of contour location errors that grey images introduce. The features do contain enough information to make relatively good pose hypotheses from a single match, but their use of the Hough transform is very unforgiving. It will not successfully group consistent correct matches that have any error. In the HYPER system, the contours are approximated as lines, and the connectivity is used to drive the match expansion. With grey images, the segmentation and connectivity could be significantly reduced, but it is difficult to tell how much this would affect the algorithm's success. The Tree Pruning system and all of my systems are designed to use grey-scale images, but can suffer badly from highlights or parts with intricate detail that obscure the object boundary. Most of the tests were performed with solid-colored objects which had some contrast with respect to the background. While human vision can find a very low contrast line by integrating the evidence along its length, current edge-finding algorithms are primarily local and error-prone.

All of the systems require very simple images, and only HYPER demonstrates the successful recognition of any moderately intricate objects. It is easy to underestimate the difficulties which would arise in more complicated images that include textured regions, highlights, shadows, and objects with many visible details. The restriction to simple images seems to be due to the delicacies of edge finders, the emphasis on shape for recognition,

and the dramatic increase in the size of the search space with the number of features in the image. In order to allow realistically complicated images, all of the systems described in this thesis would need to confront the issue directly, rather than through any simple modification.

All but one of the systems requires that the image objects be in the same scale as the models. It is not necessarily easy to relax this restriction. Purely data-driven methods can sometimes make straightforward modifications to include another degree of freedom in the pose hypothesis, at least theoretically. For instance, Hough transform techniques can simply add another dimension to the pose-array. First, memory limitations may prevent this. In addition, this would require being able to make scale estimations from a single feature match, which in turn would require more complicated and accurate features. For the Tree Pruning approach, initial tests have already been performed in which the constraints are modified to divide out scale, forcing one less constraint and greater complexity in the remaining ones. The effect on recognition speed could be very large. The Saliency method, using sections of the orientation graph, would require so many more comparisons that it would be totally impractical. For hypothesis-driven methods, the problem is equally difficult. If the initial hypotheses consist of a complete pose estimate (including scale), they will require the same kind of extra work as data-driven methods require. Without a scale estimate, it would be very difficult to use the model to find further matches. If hypothesis development includes pose refinement, or does not begin with an accurate estimate of scale, then it will need to follow a more conservative development approach to avoid committing to an incorrect partial hypothesis. HYPER relies on either an *a priori* estimate of the scale, or on the length of the image segment, which is very error-prone, especially across different scales. Unless the line-segmentation module adjusts for scale (which is not known at the time of segmentation), it will segment curves using roughly constant size segments, which do not reflect on the scale of the object at all. Only polygons might work over a reasonable range of scales. This weakness is helped by the use of the connectivity of the segments to drive the hypothesis development, until the least-squares estimate of scale has enough information to be accurate.

The explicit inclusion of small model libraries is made by the Saliency method and the LFF system. In the Saliency method, making distinctions among the library models is the entire motivation for the use of saliency. It involves a special preparatory stage in

which the models in the library are compared, and the results of the comparison are used during recognition to influence the choice of model match. Unfortunately, the emphasis on model distinction is to the exclusion of model recognition in many cases. Despite using model comparisons to weigh feature interpretation, they are not used to speed up the search. Each model is still accessed independently and sequentially. The LFF system also performs considerable precomputation on the model library, using the similarities of features to combine them into cluster groups, and using the differences to optimize the search strategy for each cluster. This kind of preparation can improve execution times a great deal. Execution time need not be proportional to the number of models in the library. Since an efficient method for discriminating among models is determined ahead of time, unnecessary comparisons are never made. Other systems, such as HYPER, Tree Pruning, and my own, could theoretically be converted for use with libraries. The models could be checked sequentially, but it would then be necessary to include the ability to find no match for a model. As mentioned in section 2.3.2, this requires a more careful verification of good hypotheses, and probably a threshold on a heuristic quality measure. Recognition, like test taking, is more difficult when "no match" is a possible outcome. To be sure that no match can be made, many of the poor initial matches must be tried. It is hard to prove that an initial match is so poor that it could not possibly be part of a correct interpretation. Thus, even the fastest positive recognizers can become very slow if forced to make a confident negative choice. In fact, this can increase recognition times by several orders of magnitude. The LFF system avoids considering all the weak hypotheses by "using up" the image. If it can find good hypotheses which account for every image feature, then it can make the negative claim that the other hypotheses are not possible. This approach can be much quicker in this domain, but as images get more complicated or when non-modeled objects appear in the image, it can fail without any grace. Any sequential approach, and most known approaches to model libraries become impractical when the number of models becomes large enough to be useful in everyday situations. The problem of large libraries has still not received much attention, but a hierarchical classification of objects seems necessary.

4.5.2 Extension to 3D Domains

Several of the systems have been extended to the domain of three-dimensional data and models. The Tree Pruning system was developed with this generalization in mind, and requires only sparse surface position and orientation information. The constraints for this domain are very similar, and only the global consistency test is significantly more complicated. An extension of the LFF system, called the 3DPO system, was proposed using clusters of 3D features such as cylinders and coplanar surface discontinuities. A description of a working system has very recently been published, and the general concepts seem to extend well. My algorithms depend heavily on features in the orientation graph. The three-dimensional analog may not be as useful: 3D surface features may be difficult to find repeatably, and may not summarize the surface as completely as they do in two dimensions. It is still important to focus the search, and the improvements to the Hough transform are still applicable. The concept of hypothesis-driven search is quite generalizable. The use of connectivity to direct early hypothesis growth would probably still be useful, even though surfaces do not connect linearly.

None of the systems attempted to perform recognition under the kind of object transformations that occur in a projection of the three-dimensional world on to a two-dimensional image. The design of recognition systems in this domain is typically overwhelmingly influenced by projection. None of the methods reviewed would be reasonable choices to extend in a similar form to the projection domain, because of their strong rigidity requirements.

4.5.3 Feature Choice

The LFF system requires human definition of the best features for the application. The assumption that a good set of features can always be found may not be sound. Their use of corners and holes restricts the class of recognizable objects considerably. I believe that the clustering method will still work with a more general set of features, but the determination of such a set is not straightforward. Is it the case that there exist objects for which no reasonable feature set exists? Is there one set that will work for many objects? In my systems, I attempt to use a set which covers a large number of shapes. But some simple objects, such as a circle, have only one feature, and will probably be missed, since the number of feature matches is currently used to judge (crudely) the quality of the hypothesis.

The line-segment approximation at least evenly distributes the features over curves, but its lack of repeatability is a significant drawback, as discussed above. The Saliency system features are evenly distributed, but the lack of ability to segment a contour repeatably and the inability of the feature to focus the attention of the recognition process is compensated by brute force, at the expense of increased computation and decreased flexibility.

A detailed presentation of feature criteria appears in section 3.3.

4.5.4 Search Strategy

All of the systems in this thesis avoid performing a sequential search through all possible mappings of the model into the image. However, all do work from the set of all possible single matches of model features to image features. All three of my algorithms and the Saliency algorithm explicitly form all possible single feature matches. The Saliency method uses every match to fill the Hough transform array. In my approaches, the initial matches are limited to features of the same type, and are immediately screened using the discrimination values associated with each type. The HYPER system only matches privileged (longer) segments, but does try all combinations. If none of these work, there is no provision for continuing with other initial matches. This is like a less accurate initial screening with a preset rule that depends only on the individual features, not on a comparison of both features. The LFF method ranks all features by the ease with which they can be processed, and begins by trying to develop the best hypotheses first. If a good enough hypothesis is developed, using a relatively accurate verification stage, the other initial hypotheses are never used. If no good hypothesis is formed, the weaker focus features are still available. The Tree Pruning system begins with all model matches to a single long but otherwise arbitrary image feature. The second stage continues with another set of matches, pruning the inconsistent branches. Towards the top of the tree, most model matches are tried by several interpretations. As long as at least one interpretation continues down through all of the levels, all individual feature matches will be tried.

This initial hypothesis generation stage is data-driven in all of the algorithms, since hypothesis-driven recognition requires a set of hypothesis seeds to begin, and the set of all hypothesis seeds is too large. The LFF system, HYPER, and my match extension module (used by my second and third algorithms) immediately reverse the control of the search and

use each initial match to determine the next match to try. In the LFF system, this is the purpose of clusters. In HYPER and in the match extension module, connectivity is used to guide the development of the match. In both cases, the length of the match extension is used to judge the hypothesis, and only better hypotheses are kept, reducing the number of hypotheses for further verification. Hypothesis-driven search can be very fast, as is implied by the execution times shown in section 3.10. As discussed in section 2.3.3, the search space is quickly reduced, but at the risk of missing the best interpretation. The data-driven methods also reduce the search space, but by other means. Tree Pruning drastically reduces the number of complete mappings which are considered. Before pruning, the leaves of the tree represent all possible (on the order of 10^{76}) mappings. After pruning, Grimson reports that from one to fifteen interpretations survive, having visited typically 4000 nodes in the entire tree. The Saliency system and my Pose Indexing system reduce the number of complete mappings through the Hough transform. Individual matches are grouped by hypothetical pose, so very few of the possible combinations of individual matches are tried. Ignoring the less filled buckets further reduces the search, effectively using a quality measure based on the popularity of the pose.

The LFF system and the Extended Match Grouping system both require an additional data-driven stage before final verification. The LFF cluster-guided matches are searched for consistency using a maximal-clique algorithm. This is needed to resolve the ambiguity in pose and model choice remaining after the focus feature match. (The process is remarkably similar to Tree Pruning). The Match Grouping system groups the larger matches by pose using group accretion. This is needed because of the local restriction of the search—connectivity cannot be relied upon to globally combine consistent hypotheses. A model may contain several separate contours, and these are further disconnected by occlusion and edge-finding errors. HYPER should also require such a stage, but none is mentioned.

Finally, all but the Saliency system and my Match Grouping system make a last evaluation of the hypotheses in a verification stage. The LFF system uses the each cluster hypothesis to find feature matches throughout the image. In addition, the object is placed in the image and checked all along its contours to make sure that an object/background boundary exists. (As mentioned, grey images would make this much more difficult). The Tree Pruning system has already considered all matches, but has only applied pairwise consistency checks. A pose estimate is made, and each match is checked for consistency with

the pose. The HYPER system uses the hypothetical pose developed using connectedness, and redevelops the segment matches using the more accurate initial pose estimate. This is needed since one-to-one line-segment matching cannot be relied upon. The Match Verification system checks for image features predicted from the pose implied by the extended match, in a manner similar to HYPER's segment matcher. Matches close to the seed match are checked first, and the pose estimate is adjusted as matches are added to the hypothesis. In my method, however, the search is not limited by connectivity. The Pose Indexing method verifies the bucket hypotheses by expanding each match using contour connectivity, and then using the total number of matches as a measure of the match quality.

Chapter 5

Conclusion

In this thesis I have presented general issues in model-based vision, and focused on specific topics within the two-dimensional domain. It is hoped that the identification of the trade-offs they involve will make the task of designing new recognition systems clearer. In particular, the distinction between hypothesis- and data-driven recognition and the investigation of their strengths and weaknesses should prove useful in the evaluation and development of future systems.

Chapter 1 included a definition of model-based recognition and of the two-dimensional domain. The chapter identified the major axes along which recognition algorithms may differ, and the importance of these aspects in vision. The aspects discussed were: the dimension of the world and data, the class of identifiable objects, the ability to recognize objects despite occlusion, the type of sensory data and special viewing restrictions, and the inclusion of a library of models.

Chapter 2 continued with a discussion of the issues related to the aspects of model-based vision, then introduced three fundamental trade-offs pertinent to the two-dimensional domain. These were: feature complexity versus matching algorithm complexity, accuracy versus speed (completeness), and hypothesis- versus data-driven recognition.

The implementation of three algorithms to explore these trade-offs was described in Chapter 3. A collection of feature criteria were derived and presented along with a feature set that satisfies the criteria and demonstrates one example of the feature complexity trade-off. Three recognition algorithms were also presented that demonstrated workable compromises involving search completeness and the exploitation of early hypotheses. All

three begin with the same initial feature matches, which are well pruned using descriptive information contained in the features. The first algorithm uses a data-driven pose indexing module that benefits from the development of a probabilistic analysis of Hough transform techniques. Unfortunately, the robust approach requires extra computation. The second and third algorithms use feature connectivity information to expand the feature matches in an efficient hypothesis-driven module. The second returns to a data-driven approach that combines the extended features by pose. Experimentally, this algorithm was found to be most successful at finding models in images. The third continues with a hypothesis-driven approach, using the extended matches to predict the locations of other features and to quickly find supporting evidence for the hypothetical pose. The third was found to be faster, but occasionally missed objects due to the dependence on an accurate ranking of the extended matches, which are only partial mappings. An evaluation of the performance of the algorithms showed many of the predicted effects, including the value of a multi-tiered system that combines hypothesis- and data-driven modules.

Four recognition systems designed by other researchers were presented in Chapter 4. The presentation and comparison served to expand the concepts of earlier chapters by providing further examples of domains, feature definitions, and matching schemes. In each case, an attempt was made to understand the advantages and disadvantages of the approaches.

5.1 Future Work

Improvements to the current algorithms have been indicated throughout the text. The circle and line features require a more stable detection routine and a means for making more complete use of the information they capture about the contour. In the pose indexing module, a more accurate way of filling the appropriate distribution of buckets would allow a more confident detection of the correct peak by increasing the ratio of the number of matches in the correct bucket to the background peaks. In the match extension module, it would be useful to be able to extend matches along contours until an occlusion prevented further matching. Currently, the extension is often limited by slight differences between the contours.

It would be informative to perform extensive testing of the effect of cutoff values on the

accuracy and speed of the algorithms. The algorithms contain many unjustified threshold values, which, for the most part, I have left "untuned" in order to avoid the development of algorithms that rely on careful tuning. However, if an analytic justification for certain optimal values could be made, it might evoke better performance from the algorithms.

On a larger scale, I believe that two important shifts in emphasis are needed to make headway against the current limitations. First, there is a need for more image interpretation before models are considered. For example, a hypothetical association of features caused by the same object would be quite advantageous. Not only would search combinatorics be reduced, but explicit reasoning about occlusion would be possible. This need becomes stronger as larger model libraries are supported, since it becomes more important to predetermine as much about the image as is possible before referring to the entire data base. In three-dimensional worlds, an analysis of shadows, highlights, shading, and physical support would all be very useful, though they are not necessarily easy to extract. A depth-map is probably not as useful as some image parsing and the identification of likely geometric relations among objects in the image.

Second, I believe that a rigid dependence on geometric shape will lead to algorithms which will not generalize well to realistic environments. Not only must flexible objects be included, but other cues should be used in addition (not independently), such as color, texture, or eventually higher-level feedback such as context. It is well known that other cues are needed, and I believe that it is important to incorporate them and expand to flexible objects before the dependence on rigidity leads to the development of techniques that are not realistically useful.

Bibliography

- Asada, H., and Brady, M. 1984. The Curvature Primal Sketch. *MIT Artificial Intelligence Laboratory Memo No. 758*.
- Ayache, N., and Faugeras, O. D. 1986 (Jan). HYPER: A New Approach for the Recognition and Positioning of Two-Dimensional Objects. *IEEE Trans. Pattern Anal. Machine Intell.* PAMI-8(1):44-54.
- Ballard, D. H. 1981. Generalizing the Hough Transform to Detect Arbitrary Shapes. *Pattern Recognition* 12(2):111-122.
- Berzins, V. 1984. Accuracy of Laplacian Edge Detectors. *Comp. Vision, Graphics and Image Proc.* 27:195-210.
- Bolles, R. C., and Cain, R. A. 1982. Recognizing and Locating Partially Visible Objects: The Local-Feature-Focus Method. *Int. J. Robotics Res.* 1(3):57-82.
- Canny, J. F. 1983. Finding Edges and Lines in Images. *MIT Artificial Intelligence Laboratory Technical Report No. 720*.
- Grimson, E., and Lozano-Pérez, T. 1984. Model-Based Recognition and Localization from Sparse Range or Tactile Data. *Int. J. Robotics Res.*, 3(3):3-35
- Grimson, E., and Lozano-Pérez, T. 1985. Recognition and Localization of Overlapping Parts from Sparse Data. *MIT Artificial Intelligence Laboratory Memo No. 841*.
- Hough, P.V.C. 1962. Method and Means for Recognizing Complex Pattern. *U.S. Patent 3,069,634*.
- Marr, D., and Hildreth, E. C. 1980. Theory of Edge Detection. *Proc. R. Soc. Lond. B* 207:187-217.
- Marr, D., and Nishihara, H. K. 1978. Representation and Recognition of the Spatial Organization of Three-Dimensional Shapes. *Proc. R. Soc. Lond. B* 200:269-294.
- Marr, D. 1976. Artificial Intelligence— A Personal View. *MIT Artificial Intelligence Laboratory Memo No. 355*.
- Turney, J. L., Mudge, T. N., and Volz, R. A. 1985 (July). Recognizing Partially Occluded Parts. *IEEE Trans. Pattern Anal. Machine Intell.* PAMI-7(4):410-421.
- Ullman, S. 1983 (June). Visual Routines. *MIT Artificial Intelligence Laboratory Memo No. 729*.