Recognizing 3-D Objects Using 2-D Images

by

David W. Jacobs

Revised version of a thesis submitted to the Department of Electrical Engineering And Computer Science in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy at the Massachusetts Institute of Technology in September of 1992.

Recognizing 3-D Objects Using 2-D Images

by

David W. Jacobs
(dwj@ai.mit.edu)

Abstract

To visually recognize objects, we adopt the strategy of forming groups of image features with a bottom-up process, and then using these groups to index into a data base to find all of the matching groups of model features. This approach reduces the computation needed for recognition, since we only consider groups of model features that can account for these relatively large chunks of the image.

To perform indexing, we represent a group of 3-D model features in terms of the 2-D images it can produce. Specifically, we show that the simplest and most space-efficient way of doing this for models consisting of general groups of 3-D point features is to represent the set of images each model group produces with two lines (1-D subspaces), one in each of two orthogonal, high-dimensional spaces. These spaces represent all possible image groups so that a single image group corresponds to one point in each space. We determine the effects of bounded sensing error on a set of image points, so that we may build a robust and efficient indexing system.

We also present an optimal indexing method for more complicated features, and we present bounds on the space required for indexing in a variety of situations. We use the representations of a model's images that we develop to analyze other approaches to matching. We show that there are no invariants of general 3-D models, and demonstrate limitations in the use of non-accidental properties, and in other approaches to reconstructing a 3-D scene from a single 2-D image.

Convex groups of edges have been used as a middle level input to a number of vision systems. However, most past methods of finding them have been ad-hoc and local, making these methods sensitive to slight perturbations in the surrounding edges. We present a global method of finding salient convex groups of edges that is robust, and show theoretically and empirically that it is efficient.

Finally, we combine these modules into a complete recognition system, and tests its performance on many real images.

Acknowledgments

Many people in the AI lab have significantly contributed to this work with discussions of vision in general and my thesis in particular, including Tao Alter, Ronen Basri, Yael Moses, Whitman Richards, Amnon Shashua, Sandy Wells, and also, everyone else who was formally or informally part of WELG group. Todd Cass particularly provided me with many valuable suggestions, and much important feedback over the years.

I'd like to thank the many people who contributed to the software and hardware environment that I made use of at MIT. I particularly made use of software provided by David Clemens.

I am grateful to Tomas Lozano-Pérez and Shimon Ullman for their very helpful service on my thesis committee.

I want to thank Eric Grimson for providing many useful ideas, much invaluable advice, and constant support over many years. This thesis would not have taken the shape that it did without his influence.

As will be evident from the thesis itself, I owe a special debt to David Clemens, who collaborated with me on work that turned into the intial formulation of this thesis. In addition to specific contributions which are acknowledged in the thesis, my many conversations with David were an invaluable source of assistance to me.

I want to thank my parents for all their support over the years. I would also like to thank my sister.

And most of all, I would like to thank Liz and Nick, for everything.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology, and was funded in part by an Office of Naval Research University Research Inititative grant under contract N00014-86-K-0685, and in part by the Advanced Research Projects Agency of the Department of Defense under contract N00014-91-J-4038, Army contract number DACA76-85-C-0010, and under Office of Naval Research contract N00014-85-K-0124.

Contents

[1] This section is a modified version of material appearing in Clemens and Jacobs[32], and should be considered joint work between the author and David Clemens.

Chapter 1

Introduction

The human ability to recognize objects visually is far more powerful and flexible than existing techniques of machine vision. People know about tens of thousands of different objects, yet they can easily decide which object is before them. People can recognize objects with movable parts, such as a pair of scissors, or objects that are not rigid, such as a cat. People can balance the information provided by different kinds of visual input, and can recognize similarities between objects. Machines can not do these things at present.

There are a variety of difficulties in modeling this human performance. There are hard mathematical problems in understanding the relationship between geometric shapes and their projections into images. Problems of computational complexity arise because we must match an image to one of a huge number of possible objects, in any of an infinite number of possible positions. At a deeper level, difficulties arise because we do not understand the recognition problem. We do not know how to characterize the output that should follow each possible input. For example, people look at a few camels, and on the basis of this experience they extract some understanding of what is a camel that allows them to call some new creature a camel with confidence. We do not know what this understanding is.

Because recognition is such a difficult and poorly understood problem, most work on object recognition has begun with some formalization of the problem that greatly simplifies it. Also, many vision researchers are more interested in constructing useful machines than in modeling human performance, and many valuable applications require recognition abilities that are much weaker than those that people possess. For those interested in human recognition, however, there is the danger that we may solve simple recognition problems in a way that does not contribute to the solution of more ambitious problems.

In this work, we have tried to make progress on concrete, simplified recognition problems in a way that still speaks to the larger difficulties of human vision. Computational complexity presents tremendous challenges in any current version of the recognition problem, and it seems that complexity is a fundamental part of the problem that the human visual system solves. Therefore we have adopted a strategy for recognition that might be extended to handle problems of arbitrary complexity. A second deep puzzle of human vision is how we describe an object and an image that we wish to compare when the object is 3-D and the image is only 2-D. We also address this problem in this thesis.

We begin this introduction by describing a well-defined version of the recognition problem that still contains the difficult problems of computational complexity and the need to compare objects at different dimensions. We then describe a strategy for handling the complexity of this problem using *grouping* and *indexing*. We also show how the indexing problem forces us to confront the difficult issue of comparing a 2-D image to a 3-D model, and describe possible solutions to this problem. But at the same time this work has been led by our intuitions about humans. After describing our approach to recognition, we will explain how it fits these intuitions.

We assume a problem statement that is commonly used in *model-based* object recognition. A model of an object consists of precisely known local geometric features. For example, we might use points to model the corners of an object. Line segments or curved contours can model sharp edges in an object that often form occluding contours. 2-D surfaces or 3-D volumes can model larger chunks of the object. Most of the work in this thesis will use points and line segments. These allow us to fully describe the edges produced by polyhedral objects, and also to capture much of the shape of some non-polyhedral objects that contain corners and sharp edges. This assumption of a precise geometric model is limiting; it is not clear that people possess such models for the objects they recognize, or how one could apply a method based on geometric models to recognize members of a class of objects, such as camels, whose individuals vary considerably. Within this problem statement, however, we still have all the difficulties of recognizing a large number of complex and realistic objects.

Given a set of such models as background knowledge, the recognition system operates on a still photograph containing a known object. Using standard techniques it locates 2-D features that are analogs of our model's 3-D geometric features. The use of standard low-level vision modules to find features ensures that we are using data that can be derived bottom-up from the image. This means, however, that perhaps due to limitations in existing low-level vision, we will detect features imperfectly. We will miss some features in the image and detect spurious features, and we will encounter significant error in localizing features in the image. So recognition becomes the problem of matching model features to image features in a way that is consistent with geometry and with these potential errors.

Figure 1.1 shows an example of such a recognition task. We use line segments to indicate the location of edges of the phone that frequently produce edges in an

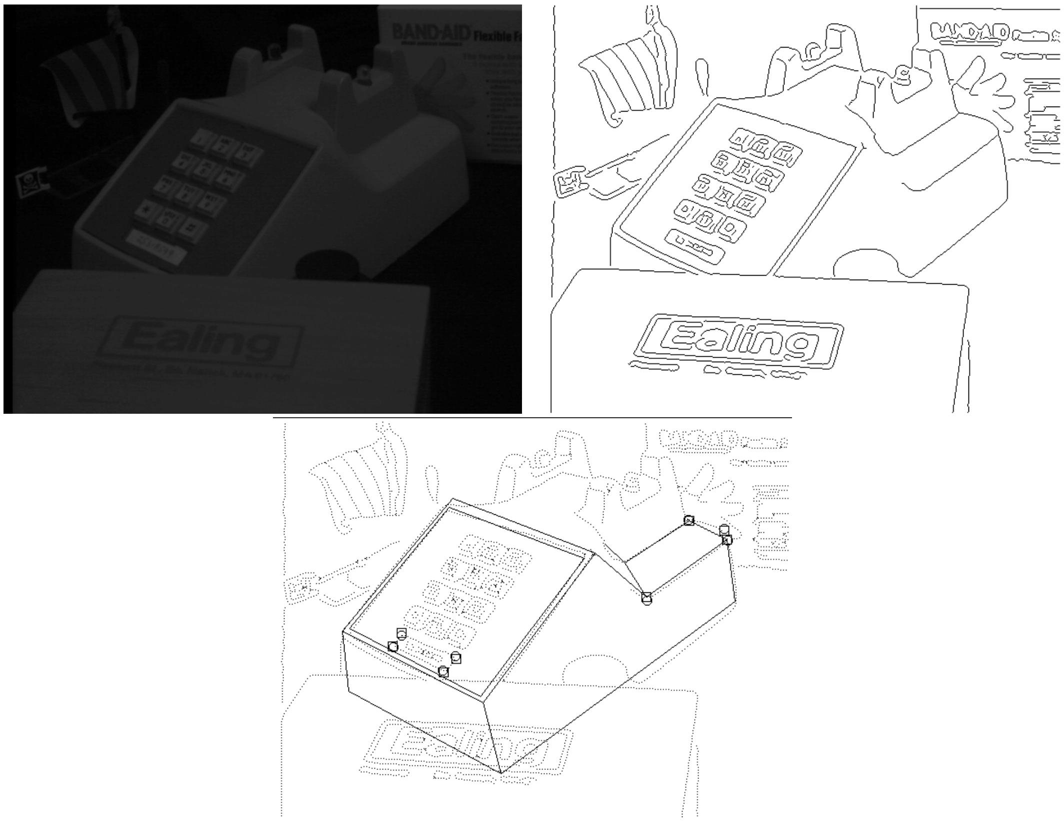


Figure 1.1: This shows an example of a recognition task in the domain that we consider. On the top left is a picture of a telephone with some objects in the background and some occluding objects. On the top right, some edges found in this image. On the bottom, some point features located in the image (circles) have been matched to some point features representing a model of the phone (squares). The image edges are shown as dotted lines, while line segments indicated the predicted location of some portions of the telephone's boundary.

image. Where two of these line segments indicate a stable vertex, we position a point feature. This representation ignores much of the volumetric structure of the phone in order to focus on simple 3-D features that usually show up as 2-D features in an image. Even these simple features, however, can capture much of the structure of a real object, such as a telephone.

Why should a version of recognition that is limited to geometry provide insight into human recognition? Under many circumstances additional information is not available to humans, and yet their recognition process proceeds, scarcely impaired. For example, when people are looking at a photograph of an object, or looking at a natural object that is far away, stereo and motion cues are not available. Frequently, objects do not have colors or textures that helps us recognize them. It seems that the visual system is able to take advantage of whatever cues are present, but that it can also proceed when many potential cues are absent. This thesis attempts to contribute to an understanding of how shape may be used to recognize objects. Because we seem able to function smoothly when only this cue is present, it seems plausible that we may be able to understand the use of shape in isolation, before we attempt to understand its interaction with other cues.

1.1 Coping with the Cost of Recognition

Once we have modeled an object using simple features we may approach recognition as a search among possible matches between image and model features. But we must somehow cope with an extremely large number of possible correspondences. Furthermore, the problem becomes harder to solve as we consider that we might have to discriminate between many tens or hundreds of thousands of different objects, and when we consider objects that have movable parts.

To illustrate the problem of computational complexity, let's consider how one of the most conceptually simple approaches to object recognition, *alignment*, requires more and more computation as our problem domain becomes more challenging. Suppose for simplicity that our geometric features consist just of points. Then, with alignment, a correspondence is hypothesized between some model and some image points. We then determine the pose of the object that would cause the model points in the correspondence to project near the image points to which we have matched them. To keep the number of poses considered at a minimum, the smallest possible number of features are matched that will still allow us to determine the pose of the object. We attempt to verify each pose by determining how well it aligns the features that are not part of the initial correspondence. Figure 1.2 illustrates this process. In practice, alignment systems may further refine a pose using additional evidence, but we will ignore this step for simplicity.

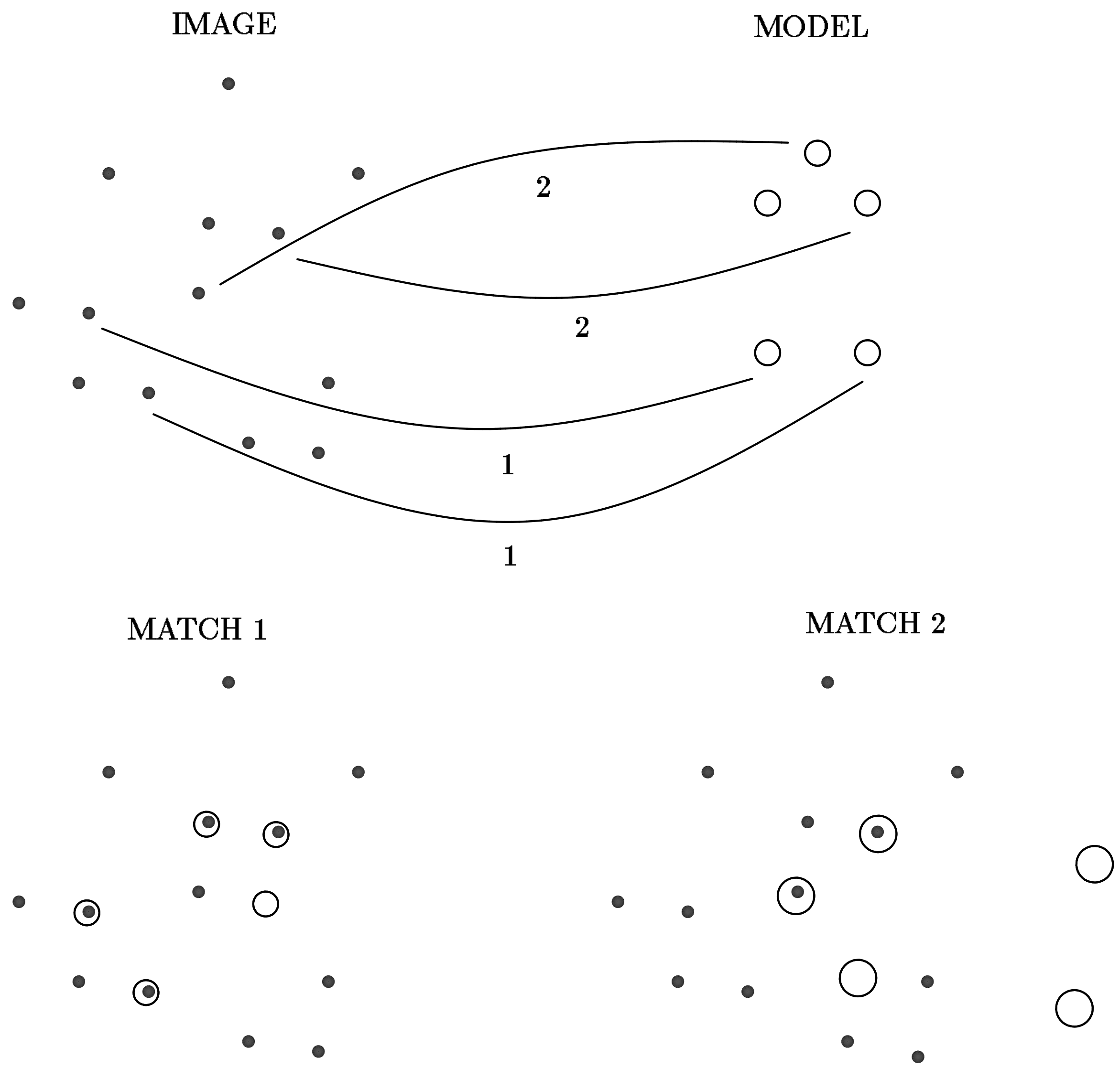


Figure 1.2: This figure illustrates the alignment approach to recognition. This example matches a 2-D model to a 2-D image with a transformation that allows for scaling, and rotation and translation in the plane. In the upper right, the open circles show some points used as an object model. In the upper left, closed circles show image points. Lines between the two show two hypothetical matches between pairs of points. The lower figures show how each of these matches can be used to transform the model into a hypothetical image location, where it may be compared with the image.

The basic idea of alignment, as I have described it, was introduced by Roberts[91], in the first system to recognize 3-D objects from 2-D images. Fischler and Bolles[43] stressed the value of using minimal matches between image and model features. Alignment has been further explored in a 2-D domain by Ayache and Faugeras[3] and Clemens[29], and in 3-D by Lowe[73], Ullman[104], and Huttenlocher and Ullman[57]. Chen and Kak[28] discuss an alignment approach for recognizing 3-D objects using 3-D scene data.

The computational complexity of alignment varies with the domain. When we assume that a 2-D image includes a 2-D object that has been rotated, translated or scaled in the plane (a similarity transform) a match between two pairs of points determines the object's pose. If there are m model points, and n image points, these give rise to about m^2n^2 possible hypotheses. This is a fairly small number, and helps explain why 2-D recognition systems have been able to overcome complexity problems, especially when the problem is further simplified by assuming a known scale. For a 3-D object viewed in a 2-D image with known camera geometry, a correspondence between two triples of points is required to determine a small number of poses. The number of hypotheses in this case grows to about n^3m^3. This number is large, and existing 3-D alignment systems use techniques that we will discuss later to avoid a raw search. If there are M different known models, the complexity grows to Mn^3m^3. And if an object has a part with a single rotational degree of freedom, such as a pencil sharpener has, then a correspondence must be established between four points to determine the pose of the object, increasing the number of hypotheses to Mn^4m^4. To place these figures in perspective, we are interested ultimately in solving problems in which the number of objects is perhaps in the tens of thousands, and the number of model and image features are in the hundreds or thousands. Even for rigid objects, basic alignment methods applied to the recognition of 3-D objects in 2-D images will give rise to perhaps 10^{19} hypotheses. Coping with such a large number of possibilities is far beyond the capabilities of existing or anticipated computer hardware, or current guesses at the capabilities of the human brain.

We have used alignment to provide a concrete illustration of the cost of recognition. This cost does not arise from some peculiarities of alignment, however. The cost of most approaches to recognition is large, and grows as we generalize the problem by generalizing the set of images compatible with a model. That is, a 3-D model viewed from an arbitrary position can produce more images than can a 2-D model viewed from directly overhead. More images are compatible with a library of objects than with one object, or with a non-rigid object than a rigid object, or with a class of objects than with a single object model. In the case of alignment, this generality adds to the complexity because the number of correspondences needed to determine object pose grows. As Grimson[47] shows, the complexity of constrained search methods grows considerably as the complexity of the task grows (see in particular Grimson's discussion of 2-D recognition with unknown scale) for essentially the same reason, larger matches are required before any geometric constraints come into play. Examples of constrained search approaches are found in Grimson and Lozano-Pérez[50], Bolles and Cain[13], Ayache and Faugeras[3], Goad[46], Baird[4], and Breuel[19]. For similar reasons, the complexity of other approaches such as methods that explore transformation space (Ballard[5], Baird[4], Clemens[29], and Cass[26],[27]) and template matching (Barrow et al.[6], Borgerfors[14], and Cox, Kruskal and Wallach[35]) will grow as the range of images that a model can produce grows. In fact, these approaches have usually been applied only to the simpler 2-D recognition problem.

1.2 Our Approach

In this thesis we show how to control this cost by doing as much work as possible on the image, independent of the model, doing as much work as possible on the model, independent of the image, and then combining the results of these two steps with a simple comparison. This approach originates in the work of Lowe[73]. It has been discussed in Jacobs[60], Clemens[30] and in Clemens and Jacobs[32], and in our discussion we will freely make use of points made in those papers. This approach reduces complexity in a number of ways. First, much of the complexity of recognition comes from the interaction between model and image, so we keep this interaction as simple as possible. Second, as much work as possible is done off-line by preprocessing the model. Third, processing the image without reference to the model can take the form of selecting portions of the image that are all likely to come from a single, unknown object. This allows us to remove most possible combinations of image features from consideration without having ever to compare them to model features. And because this process is independent of the model, it does not grow more complex as we consider large libraries of objects, non-rigid objects, or classes of objects.

The first step, *grouping* (or *perceptual organization*), is a process that organizes the image into parts, each likely to come from a single object. This is done bottom-up, using general clues about the nature of objects and images, and does not depend on the characteristics of any single object model. The idea that people use grouping may be prompted by the introspection that when we look at even a confusing image in which we cannot recognize specific objects, we see that image as a set of chunks of things, not as an unorganized collection of edges or of pixels of varying intensities. A variety of clues indicate the relative likelihood of chunks of the image originating from a single source. The gestalt psychologists suggested several clues, such as proximity, symmetry, collinearity, and smooth continuation between separated parts. For example, in an image of line segments, two nearby lines are more likely to be grouped together by people than are two distant ones, and gestalt psy-

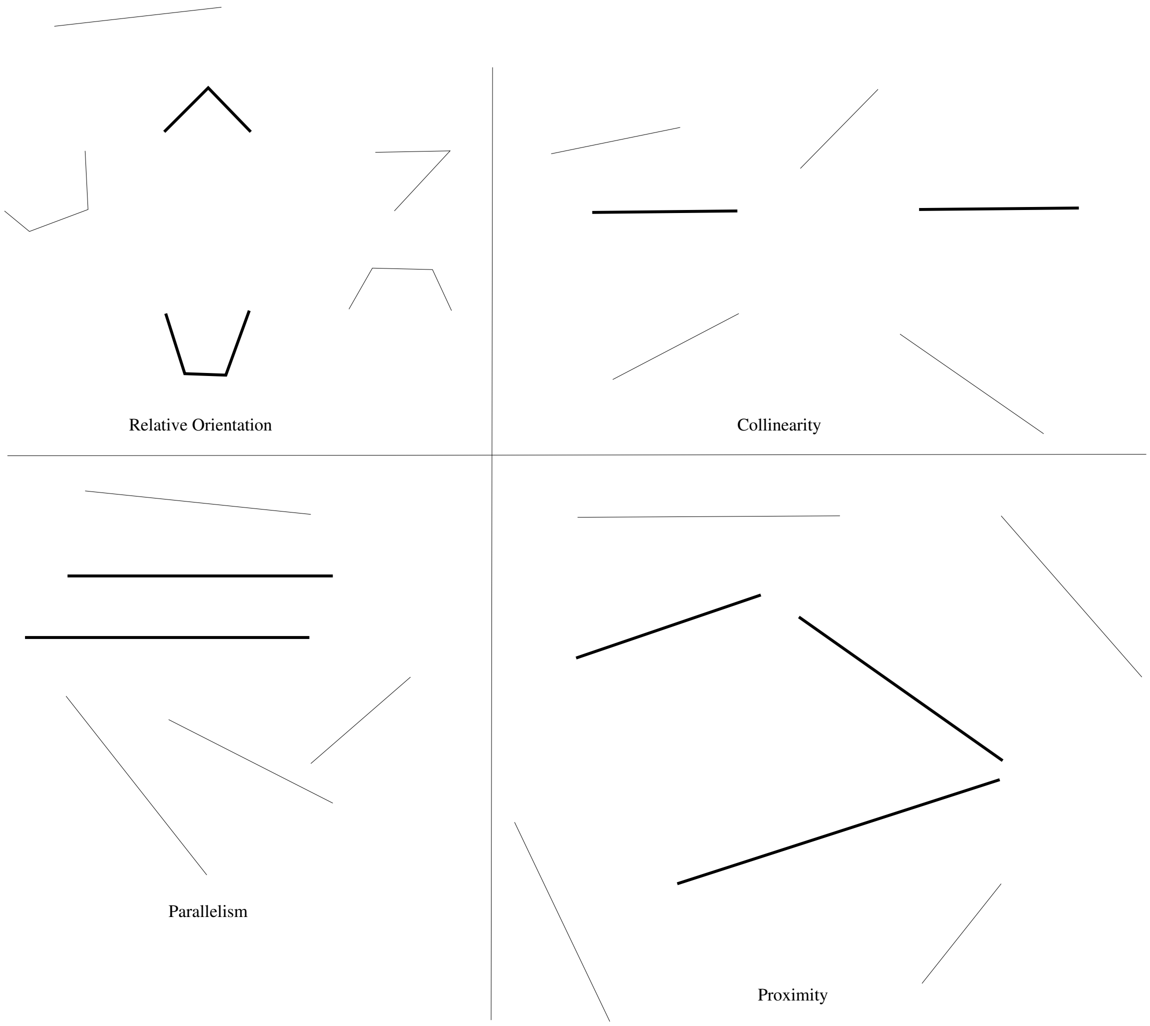


Figure 1.3: Some of the clues that can be used to group together edges in an image. In each example, some lines are shown in bold face that a grouping algorithm might collect into a single group based on one particular clue.

chologists suggested that this is because they are more likely to come from a single object (see Kohler[67] and Wertheimer[113] for an overview of this work). Witkin and Tenenbaum[115] and Lowe[73] have applied this view to computer vision. Other recently explored grouping clues include the relative orientation of chunks of edges (Jacobs[60], Huttenlocher and Wayner[58]), the smoothness and continuity of edges (Zucker[116], Shashua and Ullman[96], Cox, Rehg, and Hingorani[36], Dolan and Riseman[41]) including collinearity or cocircularity (Boldt, Weiss and Riseman[24], Saund[93]), the presence of salient regions in the image (Clemens[30]), and and the color of regions in the image (Syeda-Mahmood[99]). Figure 1.3 illustrates the use of some of these grouping clues. Our view of the grouping process is that it combines a wide variety of such clues to identify clumps of image features. These clumps need not be disjoint, but their number will be much smaller than the exponential number of all possible subsets of image features.

Grouping can also provide structure to these features. For example, if we group together a convex set of lines, we have not only distinguished a subset of image lines, convexity also orders these lines for us.

The second step in our approach to recognition is *indexing*. We use this as a general term for any simple, efficient step that tells us which groups of 3-D model features are compatible with a particular group of 2-D image features. Grouping will provide us with sets of image features that contain enough information so that only a few of the groups of model features will be compatible with them. To capitalize on this information, indexing must be able to use that information to quickly narrow its search. Ideally, the complexity of indexing will depend only on the number of correct matches, avoiding any search among incorrect matches.

The word indexing evokes a particular approach to this problem through table lookup. In this approach, we store descriptions of groups of model features in a hash table, at compile time. Then, at run time, we compute a description of an image group, and use this description to access our table, finding exactly the set of model groups compatible with our image group. There are important problems raised by such an approach in determining how to represent our model and image groups to make such a comparison. But indexing satisfies our need to use the rich information provided by grouping to quickly find only the feasible matches. It also forces us to confront one of the key problems in human image understanding: How can we describe a 3-D object and a 2-D image so that we may compare the descriptions in spite of the difference in dimensionality?

To make this discussion more concrete, I will describe the recognition system that is presented in this thesis, which is just one possible way of implementing this general approach. This system proceeds in the following steps:

- At compile time, the model is processed to find groups of line segments that appear as salient convex groups in images. We then determine every 2-D image that a pair of these groups could produce from any viewpoint, and store a description of each of these images in a hash table. One of the main contributions of this thesis is the development of an efficient method of representing all these images.

- The image is processed to find similarly salient convex groups of lines. We choose as our salient groups those in which the length of the lines is large relative to the distance between the lines. Both analytically and empirically, we can show that we can find these groups efficiently, and that these groups are salient in the sense that they are likely to each come from a single object.

- At run time, we compute a description of a pair of convex image groups, and perform matching by looking in the hash table.

- After grouping and indexing, we have a set of consistent matches between image and model features which we use to generate hypotheses about the location of the model.

- We then look for additional model lines in the image to evaluate these hypotheses.

Processing the models is done at compile time, and does not directly affect the run time of the system. Processing the image is efficient because it does not depend at all on the complexity or number of models, and because our desire to find salient groups of image features provides us with strong constraints that limit the sets of image groups we need to consider. The combinatorics of matching models and images is reduced because the time spent looking in the hash table depends primarily on the number of valid matches between model and image features, not on the number of invalid matches that must be ruled out. So with this approach the complexity of recognition depends entirely on the capability of our grouping system. First of all, it depends on the cost of the grouping system itself. Second, it depends on the number of groups produced; the more groups we must consider, the longer recognition will take. Third, it depends on the size of these groups. The more information a group provides about the object, the fewer the number of model groups that will match it by accident. Of course, it is difficult to quickly produce a small set of large groups, some of which come from the objects we are trying to recognize. But we may at least see a path to extending this approach to handle arbitrary problems of computational complexity. The better our grouping system is, the more difficult the recognition task we can handle with it.

Stepping back from this particular instantiation, the general approach that we advocate is to use grouping to form information-rich subsets of the model and image.

Grouping has not been extensively explored, but it is clear that there are many clues available in an image that indicate which collections of image features might come from the same object. If we can group together enough image features, we have a good deal of information about the object that produced these features. Grouping is necessary so that we do not have to search through all combinations of image features. If we also know what collections of model features tend to produce image features that we will group together, we can also group the model features at compile time, limiting the number of model groups we must compare with the image groups. But to take advantage of the information an image group gives us about an object, we need a flexible indexing system that can quickly match these image groups to geometrically consistent model groups. So there are two central problems to implementing our approach. We must determine how to form large groups, and we must develop indexing methods appropriate for large groups of features.

1.3 Strategies for Indexing

This thesis approaches indexing by matching a 2-D image of a scene to the 2-D images that known objects may produce. This differs substantially from most existing approaches to visual recognition, which attempt a more direct comparison between the 2-D image and the 3-D model. Direct comparisons to the model can be made if we first infer 3-D properties of the scene from a 2-D image, or if we extract special features of the image that can be directly compared to a model. Figure 1.4 illustrates these strategies. There are two main advantages to our approach. First, the image and the model can be easily compared at the 2-D level. Second, the problem of organizing the image data becomes much easier when we are not constrained by the goal of reconstructing 3-D properties of the scene that produced our 2-D data.

We make these points clearer by comparing this approach to some past work. The main thing we wish to show about previous recognition systems is that they have been limited by a desire to directly compare a 3-D model to a 2-D image. In order to accomplish this, they have focused on descriptions of the model that are view-invariant. That is, a model is described with a property that is really a function of an image, but which we may associate with the model because the property is true of all the model's images. Since a model is described with an image property, we may directly compare the model and image to see if they have the same property. As an example of a view-invariant property, if two lines are connected in a 3-D model, they will be connected in any image of that model (disregarding the effects of error and occlusion). We will see how various systems have adopted techniques for comparing images to the invariant properties of models. At the same time, we will see that when these systems perform grouping, this grouping has been centered around view-

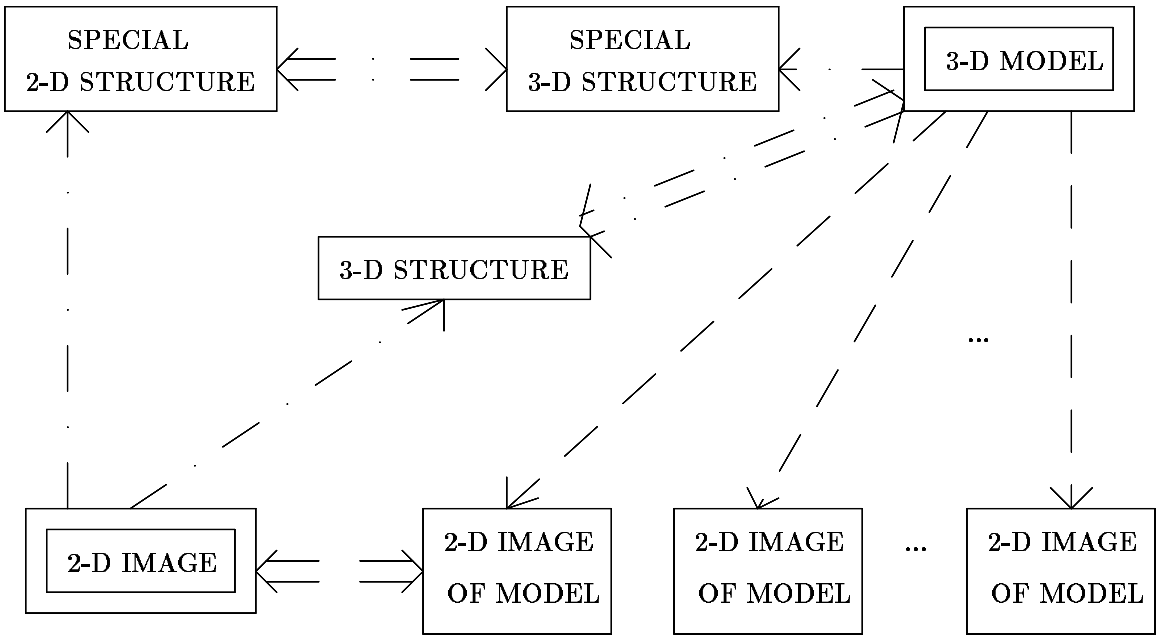


Figure 1.4: There are two main approaches to indexing 3-D models (double box in upper right) from 2-D images (double box, lower left), as shown by the two types of dashed lines. We may characterize the images a model can produce, and compare in 2-D. Or we may attempt a direct comparison to the 3-D model. We can do this by deriving 3-D structure from the 2-D image, or by in some other way creating structures from the image and model that are directly comparable. Inferences are shown by single arrows, direct comparisons are shown by double arrows.

invariant features. If these are the basis of comparisons, then it makes sense for a grouping system to organize the image into chunks based on which chunks contain view-invariant features. Grouping and indexing become entangled as grouping strives primarily to produce chunks of the image that contain the properties suitable for indexing.

There has been, however, a lack of general methods for describing a model and an image in directly corresponding ways. There has not been, for example, a comprehensive set of view-invariant features that capture all or even most of the information in an object model. To overcome this, systems sometimes make restrictive assumptions about the models. And the use of only a few view-invariant properties has meant that the clues used for grouping have been limited, since grouping has been based on these properties. Overall, because the systems suffer from a lack of general methods of inferring 3-D structure from 2-D images they ignore the 2-D information that they cannot use, both in the grouping phase and in the indexing phase. Our approach to indexing compares a 2-D image with the images a 3-D model can produce, instead of directly comparing to some property of the 3-D model. We justify this approach by describing how past systems have failed to find comprehensive methods of making direct comparisons, how this has led to indexing that uses only impoverished information from the image, and how at the same time grouping has also been caught by these limitations, and made use of impoverished information as well.

As we look at these systems, it will also be a convenient time to notice the importance that grouping processes have played in controlling the complexity of recognition systems. While grouping has made use of limited clues, its performance has been crucial in controlling the amount of computation needed by recognition systems. Even systems that do not focus on grouping have depended on it to make their algorithms tractable.

Roberts' work[91] provides a clear demonstration of these points. This early system recognized blocks world objects that produced very clear straight lines in images. Connected lines were grouped together into polygons, and connected polygons were combined into larger groups. The system started with the largest groups that it could find. For example, it would group together three polygons that shared a vertex. It would then use the number of lines in each polygon as a viewpoint-independent description of the group, and match the group only with similar model groups. Each match was used to determine a hypothetical pose of the object, and the pose was used to determine the location of additional model features, whose presence in the image could confirm the hypothesis. The system began with larger image groups, which tend to match fewer model groups, but it would continue by trying smaller and smaller groups until it found the desired object. As a last resort, the system would consider groups based on a vertex formed by three lines, and including the three vertices on the other ends of each line. Any image group of four such points could match any such model group (given the system's model of projection).

This system is relevant to our discussion in a number of ways. First, Roberts clearly recognized the importance of grouping in reducing the cost of recognition, and used groups essentially to index into a data base of objects based on viewpoint-invariant properties. Roberts also realized that larger groups could provide more discriminatory power for indexing. Only simple objects, however, could be handled by this approach to grouping and indexing. Also Roberts did not get a lot of discriminatory power out of some big groups of features. Only the number of sides in each polygon were used for indexing. As we will see, this is only a tiny part of the information that is available with such groups. This limitation is a direct consequence of using only viewpoint-invariant descriptions for indexing.

The ACRONYM system of Brooks[21], which is based on an earlier proposal by Binford[10], represents 3-D models using a limited form of *generalized cones*. For ACRONYM's version of generalized cones there is a simple mapping between properties of a cone and of its image. These "cones" are volumes which have a straight central spine. The cross-section of the cone orthogonal to this spine is bounded by straight lines or circles. As the spine is traversed, this cross-section may grow or shrink linearly. These parts have the property that their projection onto a 2-D image always has a simple form, either a quadrilateral (or *ribbon*) when seen from above, or elliptical when a circular cross-section is seen end-on. Therefore, a bottom-up grouping process may be used to find ellipses and ribbons in the image, which are matched to the 3-D generalized cones. Both the model and the image are organized into information-rich primitives. Much less computation is needed to find an object using this grouping than when simpler primitives, such as points, lines, or individual edge pixels are used, because only a few primitives must be matched to identify an object. ACRONYM seems to only need to match two of these image primitives to comparable model primitives in order to recognize a model. Also, computation is reduced because an image contains fewer of these primitives than of simpler ones.

Computationally, ACRONYM benefits from organizing both the model and the image into higher level units before comparing the two. But only 3-D primitives that project to directly analogous 2-D primitives are chosen. This greatly limits the type of objects that can be modeled. The choice of primitives to use to model objects has been made not according to the requirements that arise from the need to represent a variety of objects, but according to strong constraints on what 3-D primitives can be readily compared to 2-D parts. The grouping system is then based on the fact that a class of simple model parts project to a class of simple image parts that we can look for with a bottom-up process. Again, comparison between 3-D and 2-D is made with simple but not very general properties, and grouping is based on these properties.

Marr and Nishihara[79] also discuss a recognition strategy based on generalized cones. They do not present an implemented system applied to real images, and so some of the details of their approach are vague. A key component of their approach, however, is the assumption that they can group an image into components produced by separate generalized cones, and then detect the central axis of each cone from a single 2-D image. This also implies limiting the models and images handled so that the projections of cones have a stable central axis. They suggest that the relationships between a set of these cones will be used to index into a data base of objects. A variety of properties of the cones and their relationships are suggested for use in indexing, but it is not clear exactly how this indexing will work. To some extent, Marr and Nishihara expect to make use of 3-D information derived from stereo, or other sources, to determine the 3-D relationships between cones. But 2-D clues, such as the relative thickness of cones, or symmetry in the image, are also suggested.

Marr and Nishihara's proposal, and Marr's[78] work in general is quite important to our discussion because of their early stress on the importance of bottom-up processing for recognition. In Marr's view, a great deal of computation should be done on the image before we attempt to compare it to a model. This computation should depend on general properties of the image formation process, not on specific properties of the objects that we expect to see. In Marr and Nishihara's view, an image is broken up into component parts, and these parts and their relationships are described in detail before we attempt recognition. This is a prescription for a quite ambitious amount of grouping in recognition.

We also see in their approach some of the pitfalls of using view-invariance for grouping and indexing. Generalized cones are suggested as representations primarily because it is felt that their axes will be stable over a range of viewpoints, not because of their representational adequacy. And computing viewpoint invariant properties to use in indexing requires a great deal from the bottom-up processing. Complete, segmented generalized cones must be found, and 3-D scene information is required to compute the relationships between these cones.

Lowe's work[73] first made many of the points that we have discussed so far, while his work in turn was influenced by Witkin and Tenenbaum[115]. Lowe stressed the importance of grouping in reducing the complexity of recognition. He also for the first time stressed the value of using viewpoint invariance in grouping and indexing. He developed probabilistic arguments to support the idea that groups of features with viewpoint invariant properties are particularly likely all to come from a single object, and showed how these properties could be used for indexing as well. The primary example Lowe gives is that of grouping together parallelograms. Parallelism is a viewpoint-invariant property if we assume orthographic projection, that is, a projection model with no perspective distortion. In that case, 3-D parallel lines will always project to 2-D parallel lines, while 3-D lines that are not parallel can only appear parallel from a tiny range of viewpoints. Connectedness is also a viewpoint-invariant property. By combining these, we have a strategy for grouping by forming parallel-

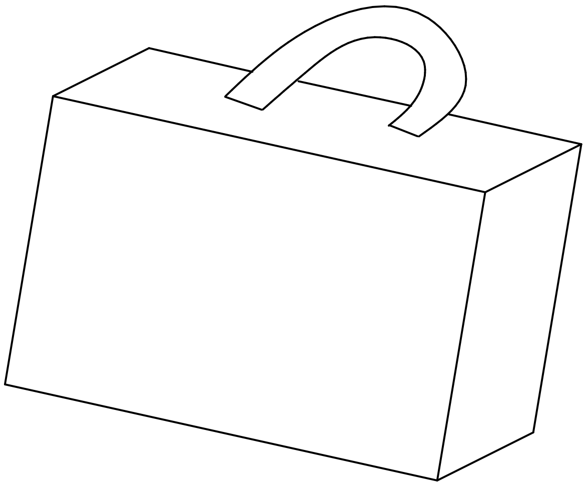


Figure 1.5: A geon-based description of this suitcase is given in the text.

ograms in the image, and indexing by matching these only to parallelograms in the model. Such matches form a good starting point in the search for an object. Lowe's recognition system uses this approach to efficiently locate some common objects, such as a stapler.

Lowe's work provided the first analysis of grouping and how it could be used to solve recognition problems, and it has had an important influence on the work described in this thesis. His emphasis on the need for grouping in recognition is the starting point for this work. In chapter 5 we will have more to say about the viewpoint invariant features Lowe used. Here, we simply point out that Lowe's decision to match 3-D models directly to 2-D images using view-invariant features greatly limited the extent of grouping possible. Only a few kinds of features were used by his system to form small image groups, because these were the only features known that could be easily compared across dimensions. The small groups that are formed with this approach provide only limited amounts of information about the model, and so they produce only moderate reductions in the search for models. This approach also depends on models that contain significant numbers of viewpoint-invariant features, such as parallelograms.

Biederman[9] built on Lowe's work to produce a more comprehensive model of human object recognition. Bergevin and Levine[8] have implemented this theory for line drawings of some simple objects. Biederman suggests that we recognize images of objects by dividing the image into a few parts, called *geons*. Each geon is described by the presence or absence of a few view-invariant features such as whether the part is symmetric, and whether its axis is straight. Connections between parts are also described with a few view-invariant features. Together, these provide a set of features which Biederman believes will identify classes of different objects, such as suitcases and coffee mugs. For example, consider the suitcase shown if figure 1.5. The case is one geon, described by four view-invariant features: its edges are straight, it is symmetric, its axis is straight, and its cross-section along this axis is of constant size. The handle is a geon with a different description because its edges and axis are curved. An example of a view-invariant feature of the connection between the two geons is that the two ends of the handle are connected to a single side of the case. We can see that computing a description of a geon requires a fair amount of bottom-up processing. Geons must be segmented in the image, and their axes must be found.

Biederman explicitly makes the claim that view-invariant features can provide enough information to allow us to recognize objects. Although Biederman does not completely rule out the possibility of using descriptions that vary with viewpoint, he does not describe how to use such information, and downplays its importance. Throughout this section, we have been arguing that on the contrary, reliance on view-invariance has led to impoverished groups that lack discriminatory power. In chapter 5 we show some limitations to what view-invariant features can capture about the 3-D structure of an object. Here we point out two possible objections to Biederman's claim that a few parts and a few view-invariant properties can serve to distinguish objects. These are not logical fallacies of Biederman's theory, but simply empirical questions that we feel have not been adequately explored.

First, although we have emphasized the importance of grouping, Biederman's approach places especially strong requirements on grouping and other early processing modules. Because view-invariant features capture only a fraction of the information available to distinguish objects, it is natural that a system that relies entirely on these features will require more effective grouping than might otherwise be necessary. One must reliably segment an image into parts, and sufficiently detect these parts so that one can determine the viewpoint-invariant properties they possess. It is problematic whether this can be done in real images with existing early vision techniques, and Biederman's approach has only been tested on line drawings. However, since Biederman proposes a psychological theory, it is hard to know what to expect of human early vision and grouping, and so it may be plausible to assume that people can locate and describe object parts in images. This is an open question. It is not clear whether people can consistently segment an object into canonical parts whose boundaries do not vary with viewpoint, and whether we can derive robust descriptions of these parts if they are partially occluded.

Second, Biederman has not demonstrated that these view-invariant properties are sufficient to distinguish among a large collection of real objects. His theory claims that metric information along with any other information that varies with viewpoint plays no significant role in helping us to identify an object. This claim is still unresolved.

As a final example, we consider Huttenlocher and Ullman's[57] application of the alignment method to the recognition of 3-D objects in 2-D scenes. This work does not focus at all upon grouping; rather it analyzes many of the problems that arise in implementing alignment for 3-D to 2-D matching. However, to get the system to work in practice, a simple grouping method was used. Pairs of vertices that were connected by a straight line were combined together. This type of connection is another viewpoint invariant feature. This example points out the omnipresence of grouping and indexing based on simple viewpoint invariant features in practical recognition systems.

Each of these systems uses a different set of view-invariant features to match between a 2-D image and a 3-D model. In each case we can see that only a small fraction of an object's characteristics can be used for indexing. Most of an object's appearance is not view-invariant. One part of an object may be much larger than another in some images, but this difference in relative size may change over views. Shapes in an object may appear sometimes convex and sometimes non-convex, sometimes sharply curved and sometimes moderately curved. The approaches discussed above must ignore this kind of information when doing indexing. As a result, the types of objects that can be modeled are quite limited. And the groups that are formed in the image tend to be small because there are few clues available for grouping. Witkin and Tenenbaum and Lowe have argued that when a set of features produce a view-invariant property this is a clue that these features come from a single object. But even if this is true, view-invariance is not the only such clue. However, it is the only clue that tends to be used for grouping when indexing relies on view-invariance. Forming small groups using these properties has been a useful step beyond raw search. But these groups do not contain enough information to discriminate among a large number of possible model groups. So these systems are left with a good deal of search still to perform.

One could take these comments as a spur to further research in view-invariant features. With more such features available to us, we could provide coverage for a wider range of objects, and we could expect to find image groups containing many of these features. In chapter 5 we are able to fully consider this question for the simple case of objects consisting entirely of point features. We characterize the set of inferences that we can make about the 3-D structure that produced a particular 2-D image. And we show that any one-to-one mapping from a 2-D feature to a 3-D feature will have serious limitations, and will require strong assumptions about the nature of our object library in order to be useful.

This thesis argues that we should transcend the limitations of view-invariant properties by comparing a 2-D image against the 2-D images that our 3-D models can produce. We show that these images can be simply characterized. This allows us explicitly to represent the entire set of image groups that a group of model features can produce, when viewed from all possible positions. Indexing then becomes the problem of matching 2-D images, which is relatively easy. As a result, we are able to build an indexing system that can use any group of image features to index into a model base that represents any group of model features. This means that our bottom-up grouping process is not constrained to produce groups with a special set of features. Grouping may make use of any clues that indicate which image features all belong to the same object.

1.4 Grouping and Indexing in this Thesis

In this introduction, we have focused on the combinatoric difficulties of recognizing objects. We have described how these problems can be overcome by a system that forms large groups of image features, and then uses these features to index into a library that describes groups of model features. This gives rise to two difficult subproblems: how do we form these groups? and how do we use them for indexing?

There are two criteria that we might use in forming groups of image features. First we can group together features that all seem likely to come from a single object. And second we can form groups that contain properties that are required by our particular indexing scheme. I have described how these two motivations have become entangled in many existing recognition systems, so that groups are formed only if they fulfill both criteria at once. When view-invariant properties are used for matching, grouping is limited to producing groups that have these properties. In this thesis we suggest that more extensive grouping can be done when our indexing system allows us to make use of any clues in the image that indicate that features originate with a common object.

In chapter 6 we describe a grouping system that makes use of one such clue, by locating salient convex groups of image lines. Convexity is a frequently used grouping clue because objects often have convex parts. We define a notion of *salient* convexity that allows us to focus on only the most relevant convex groups. We show both experimentally and analytically that finding these groups is efficient, and that such groups will be likely to come from a single object. This is not meant to suggest that salient convexity is the only, or even the most important clue that can be used in grouping. Rather, it is a sample of the kind of work that can be done on grouping. It is a thorough exploration of the value of one grouping clue out of many.

The second key problem to this approach to recognition is indexing. We have described two possible approaches to indexing. The first approach compares 2-D structures directly to 3-D structures. This is done using a one-to-one mapping from 2-D to 3-D properties. We have described in this introduction how existing systems have made use of only a limited set of such properties, and in chapter 5 we thoroughly examine such properties, and demonstrate some limitations to their use for indexing. The second approach to indexing is to compare the model and image at a two-dimensional level. The core problem to implementing such an approach is determining the most simple and efficient possible way of representing the 2-D images that a 3-D group of features can produce. We solve this problem in a variety of cases in chapter 2. These results have the additional benefit of providing a new and very simple way of looking at the matching problem in recognition. In fact, these results form the basis for the results described in chapters 3 and 5, in which we explore the limitations of some other approaches to vision.

We then put these pieces together into a working recognition system. In chapter 4 we show how to build a general indexing system for recognizing real objects. In that chapter we consider practical problems, such as how to account for image error. We then combine these pieces into a complete recognition system that first forms salient convex groups, and then use these groups to index into a model base of object groups. The matches produced by indexing generate hypotheses about the locations of objects in the image, which are then verified or rejected using additional information about the model. In chapter 7 we test this system and demonstrate that the combination of grouping and indexing can produce tremendous reductions in the search required to recognize objects.

1.5 Relation to Human Vision

In this introduction we have focused on a simplified version of the recognition problem in which we match local geometric features. Although we noted that it is not clear that human recognition can be fully described as using precise geometric models, this formulation of the problem is still quite general, and allows us to see clearly some of the complexity problems that arise in recognition, and how we may deal with them. However, I now want to more weakly claim that our approach to recognition is also a promising one for addressing more ill-defined and challenging recognition tasks. I claim that human vision may perform grouping and indexing, and that in addressing these problems in a simple domain we are taking steps along a path towards understanding human vision.

Let us consider an example. It is my intuition that one recognizes something like a camel by first organizing an image of a camel into parts and then noting features of these parts. The camel's torso might be described as a thick blob, with four long skinny parts (legs) coming from the bottom corners of this blob, and a hump (or two) on top of this blob. A long list of such features could flesh this description out in greater detail. This idea of recognizing an object using a rough description of its parts and their relations is not new. It seems to me to be our naive notion of how recognition would work, and appears as a more technical proposal in the work of Marr and Nishihara[79], Biederman[9], and Hoffman and Richards[52], for example.

As I have described above, however, the stumbling block for these proposals has been a felt need to describe an object's parts in terms of view-invariant properties. This has resulted in proposals that are simple and direct. A small number of such properties is proposed which provide a canonical description of the image, which can be matched directly against known objects. However, it is not clear whether such features could be rich enough to distinguish one object from all the others, and the paucity of features used usually implies that they must all be perfectly recovered for recognition to succeed. In fact, for reasons that are described in detail in this thesis, I think that the search for an adequate set of view-invariant features which can explain human recognition is not a promising one.

Instead, I propose that we use 2-D features that need not be view-invariant to describe each of the images that a model might produce. For example, a camel's legs are long and skinny, but "long and skinny" is not a view-invariant feature. From a range of viewpoints, such as many overhead views, the legs will not look long and skinny. In general, there are some objects that never look long and skinny, and other objects that look more or less long and skinny some or most of the time. Different object parts produce this feature, to different degrees, with varying likelihood. It is my intuition that such non viewpoint-invariant features as the relative size and general shape of an object's parts are crucial for recognizing it. If this is true, then we should describe objects that we know about in terms of the 2-D features that tend to appear in images of those objects. The first step in using such features is to understand the set of images that a model might produce. Once again, the central problem that we face is to find the simplest and most efficient representation of this set of images. This will provide us with a means of understanding the extent to which different sets of 2-D features can capture the shape of a 3-D object.

It is also clear that grouping will be a fundamental problem in this attempt to model human recognition. The intuitive strategy that I have described will also depend on some bottom-up process that at least roughly groups together parts of the image into objects and object parts. This will be a necessary first step towards determining a set of features in the image that all come from one object.

These intuitions form a secondary justification for addressing the problems that we have in this thesis. I believe that the problems of grouping and of describing the images a model produces are central ones for developing an understanding of how we may describe 3-D objects using a set of 2-D features.

Chapter 2

Minimal Representations of a Model's Images

2.1 Introduction

In this chapter we discuss optimal methods of representing the set of all images that a group of object features can produce when we view them in all possible positions. This is the central problem for our approach to indexing, illustrated in figure 2.1. As we describe in chapter 1, we can effectively perform indexing by comparing an image to the set of all images a model can produce. Since it is easy to compare 2-D images, we can build a general indexing system that uses all the available information in any groups of image features that we form. This releases us from the limitations of methods that derive 3-D scene information from a single 2-D image. However, to index efficiently we must understand how we can best represent a model's images. Understanding this question is also valuable for analyzing other approaches to recognition. When we know what images a model can produce, we can determine what it is that any particular description of a model captures about that model's images.

Throughout this chapter we will be focusing on the problem of matching an ordered group of model features to an ordered group of image features. Therefore when we talk about the model or the image we will not mean the entire model or image, but only a particular set of features that have been extracted from the image and grouped together by lower level processes. In chapter 6 we will describe these processes.

We take a geometric approach to the problem of representing a model's images. This approach is based on work of the author and David Clemens, as described in Clemens and Jacobs[32]. Related approaches can be found in Bennett, Hoffman and Prakash[7], and in Richards and Jepson[90]. We describe models using a *manifold* in *image space*. We will only be making use of some elementary properties of manifolds. So for our purposes, the reader may think of an n-dimensional manifold intuitively

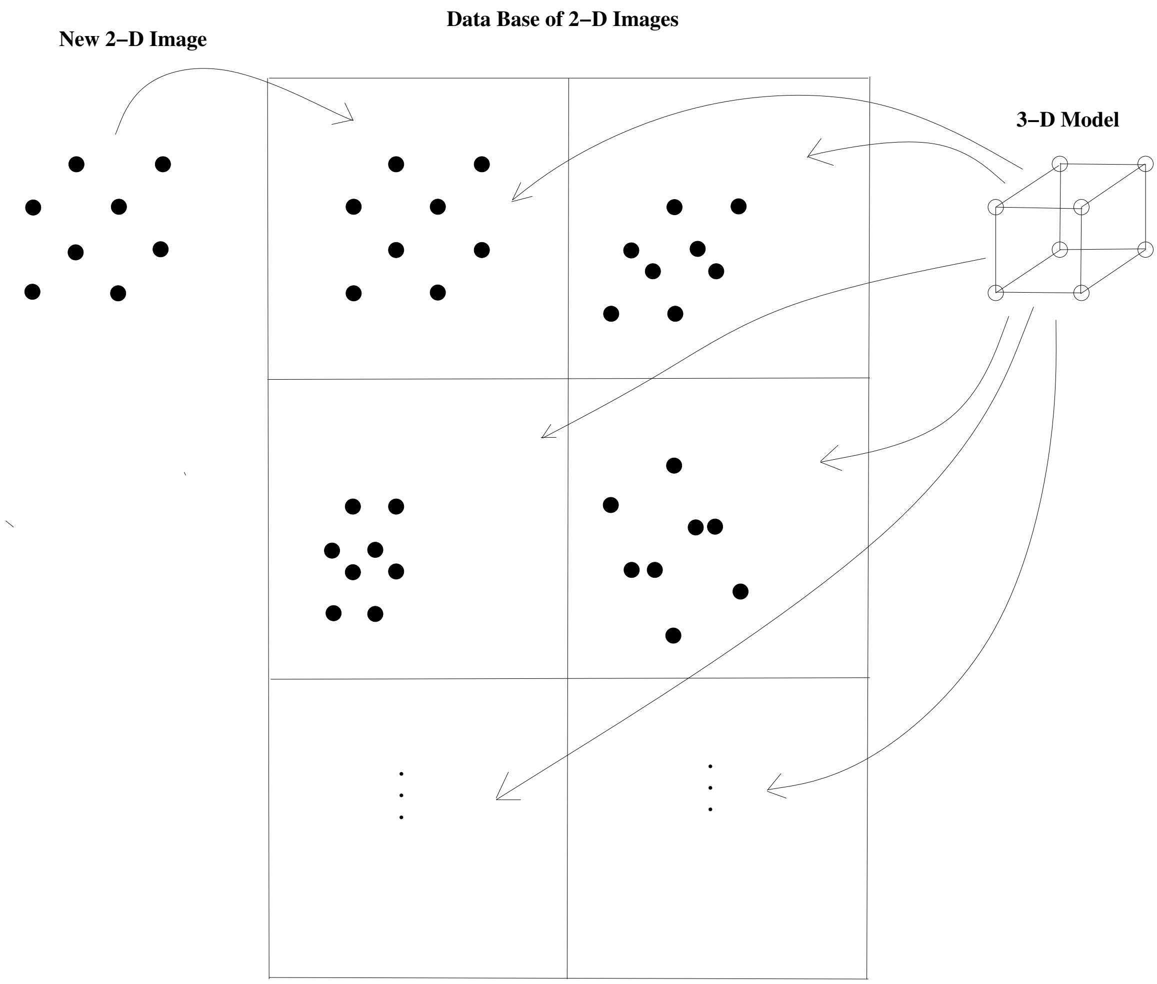


Figure 2.1: With indexing, all possible 2-D images of a 3-D model are stored in a lookup table. A hash key is then computed from a new image, and used to find any 3-D model that could produce that image.

as an n-dimensional surface in some space. An image space is just a particular way of representing an image. If we describe an image using some parameters then each parameter is a dimension of image space, and each set of values of these parameters is a point in image space corresponding to one or more images that are described by this set of parameters. For example, if our image consists of a set of n 2-D points, then we can describe each image by the cartesian coordinates of these points, $(x_1, y_1, x_2, y_2, ...x_n, y_n)$. These coordinates describe a $2n$-dimensional image space. There is in this case a one-to-one mapping from the set of possible images to each point in this image space. Suppose our models consist of sets of 3-D points. As we look at a model from all possible viewpoints it will produce a large set of images, and this set of images will map to a manifold in our $2n$-dimensional image space. We will therefore talk about a model group producing or corresponding to a manifold in image space[1]. This is illustrated in figure 2.2.

The most obvious motivation for thinking of these image sets as geometric surfaces is to discretely represent these surfaces in lookup tables. To do this, we discretize the image space, and place a pointer to a model in each cell of the discrete space that intersects the model's manifold. Then a new image will point us to a cell in image space where we will find all the models that could produce that image.

The main advantages of this geometric approach are less tangible, however. It allows us to visualize the matching problem in a concrete geometric form, particularly as we are able to describe a model's images with very simple manifolds. Beyond indexing, we may also use this approach to determine what can be inferred about the 3-D structure of a scene from one or more 2-D images of it. An image corresponds to a point in image space, so if we know what manifolds include the point or points corresponding to one or more images, we have a simple characterization of the set of scenes that could produce them.

Our geometric approach also provides a straightforward generalization of the notion of an invariant. An invariant is some property of an object that does not change as the object undergoes transformations. Invariants have played a large role in mathematics, and in the late nineteenth century geometry began to be viewed as the study of invariant properties of geometric objects. Invariants have played a significant role in perceptual psychology and in machine vision, as we will describe later. For some interesting objects and transformations, however, invariants do not exist. We will show, for example, that for sets of 3-D features projected into a 2-D image, there are no invariants. There has been no natural generalization of the notion of invariants

[1] Actually, it is not necessary that a model's images be represented by an n-dimensional manifold in image space. For example, one can devise representations of an image in which the dimensionality of a small neighborhood of images can vary from neighborhood to neighborhood. However, such representations seem somewhat far-fetched, and so the assumption that a model produces a manifold in image space does not seem too restrictive.

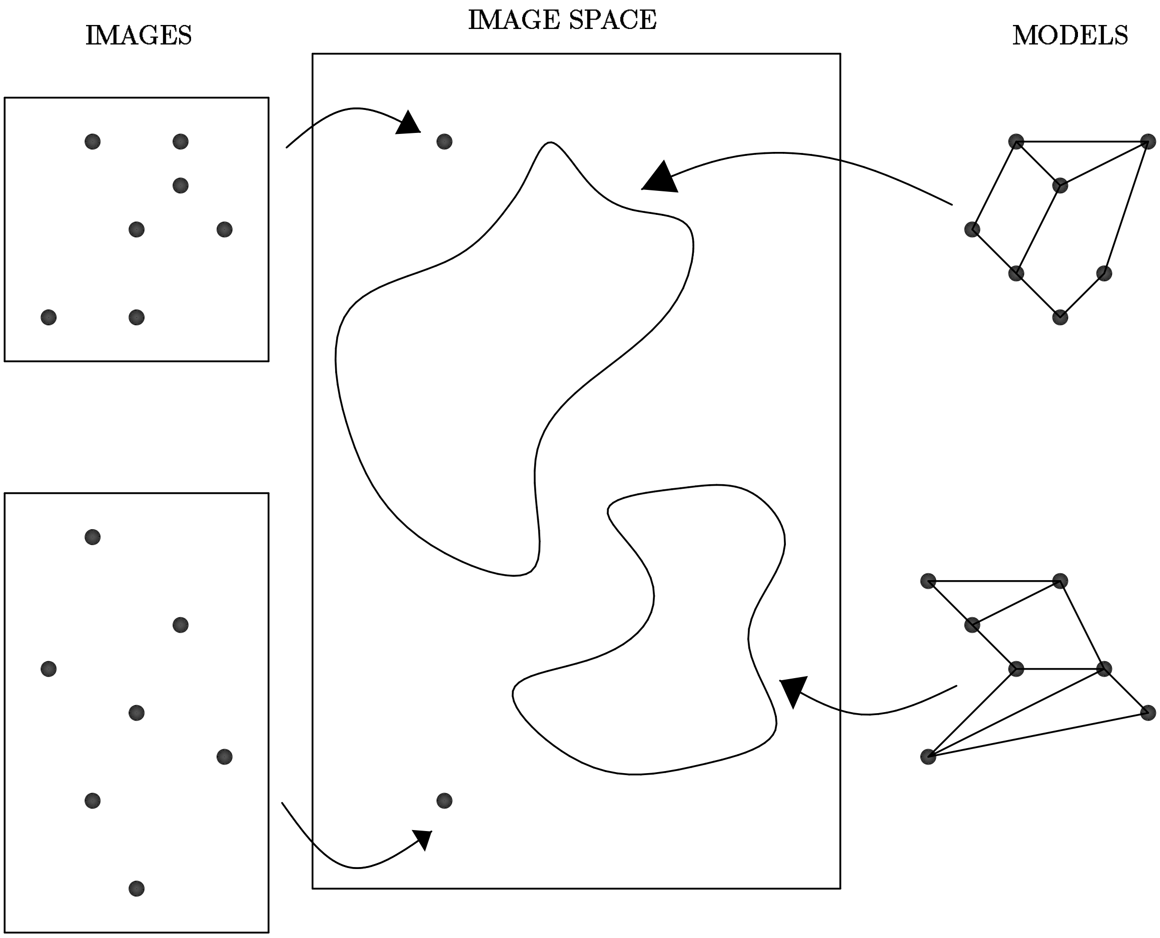


Figure 2.2: Each image of 2-D points (on the left) corresponds to a single point in image space. A model of 3-D points (on the right, lines are included with the model for reference) corresponds to a manifold in this image space.

for exploring these situations. If we view invariants in vision geometrically, we may consider them as representations of images that map all of an object's images to a single point in image space. That is, the image space representation of a model's image does not vary as the viewpoint from which we create the image does. When viewed geometrically, invariants have a natural extension. If we cannot represent an object's images at a single point in image space, we ask for the lowest-dimensional representation that is possible. We explore that question in this chapter.

There are many different ways of representing an image, and each representation will cause a model to produce a different kind of manifold in image space. There are also different ways of modeling the imaging process, using different classes of transformations such as perspective projection or orthographic projection. The type of projection used will determine which images a group of model features can produce, and hence to what manifold a model will correspond. So our goal is to find a representation of images and a type of projection which will produce the "best" mapping from groups of model features to manifolds.

In discussing representations of a model's images, we ignore the possible effects of sensing error on these images. Error is omnipresent, and if our indexing system cannot take account of error it is worthless. However, error can be handled in two ways. We might represent all the images that a model could produce when both changes in viewpoint and sensing error are considered. Then we would only have to compare a novel image against this set of images to perform robust indexing. Instead, we represent a model's error-free images only. Then, given a new image, we determine all the images that could be error-free versions of our new image, and compare this set of images to the error-free images the model could produce. So we may defer considering the effects of error until chapter 4, when we discuss practical aspects of implementing our indexing system.

For a mapping to be good it should meet several criteria. First, it is most useful if we can analytically determine the manifold that corresponds to each possible model group. Second, we would like to describe models with manifolds of the lowest possible dimension. This will contribute to the conceptual clarity of our representation. It will also make it easier to discretely represent our image space. In general, when we discretize image space, the amount of storage space required to represent each manifold will be exponential in the dimension of the manifold. For example, suppose we discretize a finite, 3-D image space by dividing the space into cubes. We can do this by cutting each dimension of the space into d parts, producing d^3 cubes. A line can pass through no more than $3d$ cubes, while a plane will pass through at least d^2 cubes. Since we will use discretizations of image space to allow us to perform indexing with table lookup, the dimensionality of models' manifolds and hence the number of discrete cells they intersect is of great importance. While a particular representation might cause different models to have manifolds of different dimensions, we will only judge a representation by the dimensionality of the highest-dimensional manifold it produces.

Third, we would like a representation that produces no false positive and no false negative correspondences. False negatives mean that a model's manifold does not represent all the images that the model can produce, while false positives mean that some images map to a model's manifold even though the model could not produce those images. Some of these errors may be implicit in our choice of a projection transformation. For example, scaled orthographic projection only approximates perspective projection, and so the set of images a model produces with scaled orthographic projection does not include all the images that the model could really produce, and includes some images the model could not produce. But in addition our representation of images might introduce some errors. We may choose a mapping whose domain is not the entire set of images. For example, we can imagine that by choosing a mapping from images to image space that ignores a small number of images we can reduce the dimensionality of the manifolds in image space. Models that produce these images will then find that their manifolds do not fully represent them. Or if our mapping from image to image space is many-to-one and it maps an image that a model could produce and an image that model could not produce to the same point in image space this will cause a model's manifold to correspond to images that it could not produce. We would like to avoid these errors, and in this chapter we will assume that no such errors are allowed, except for those errors implicit in our choice of a transformation. In chapter 5 we will explore the trade-offs that might be achieved by relaxing this goal in favor of other goals.

Fourth, it will also be useful if our manifolds have a simple form. We find it easiest to reason about a mapping that takes groups of model features to linear manifolds. Fifth, for a representation of images to be useful it should be continuous. That is, in the limit, a small change in an image should result in only a small change in the location of the point in image space that corresponds to the image. Without this condition, our representation will be unstable when even small amounts of error occur in sensing our image. Continuity is a very weak condition for an image representation to meet, but it will turn out to be an important assumption that allows us to prove that certain representations cannot be obtained.

Sometimes we can better meet these objectives by dividing our image space into orthogonal subspaces. That is we represent an image with a set of parameters, and use disjoint subsets of these parameters to form more than one image space. We can use this technique to represent a class of high dimensional manifolds more efficiently as the cross-product of lower dimensional manifolds.

There are a large range of questions we can ask about the best way to map a model to image space, because the problem can vary in many ways. First, we can consider different transformations from model to image. Some transformations are more accurate models of the image formation process, while others are more convenient mathematically, allowing us to derive more powerful results. In this chapter, we consider four types of projection: perspective projection, projective transformations, scaled orthographic projection, and linear projection. Second, there are many different kinds of geometric features that we would like to consider as part of an object model. In this chapter our most extensive results concern point features. We also consider oriented point features, that is points that have one or more associated orientation vectors. And we look at some non-rigid models, such as models that stretch along a single axis, or models with parts that can rotate about an axis. Third, since there is no representation that optimally satisfies all the goals described above, we

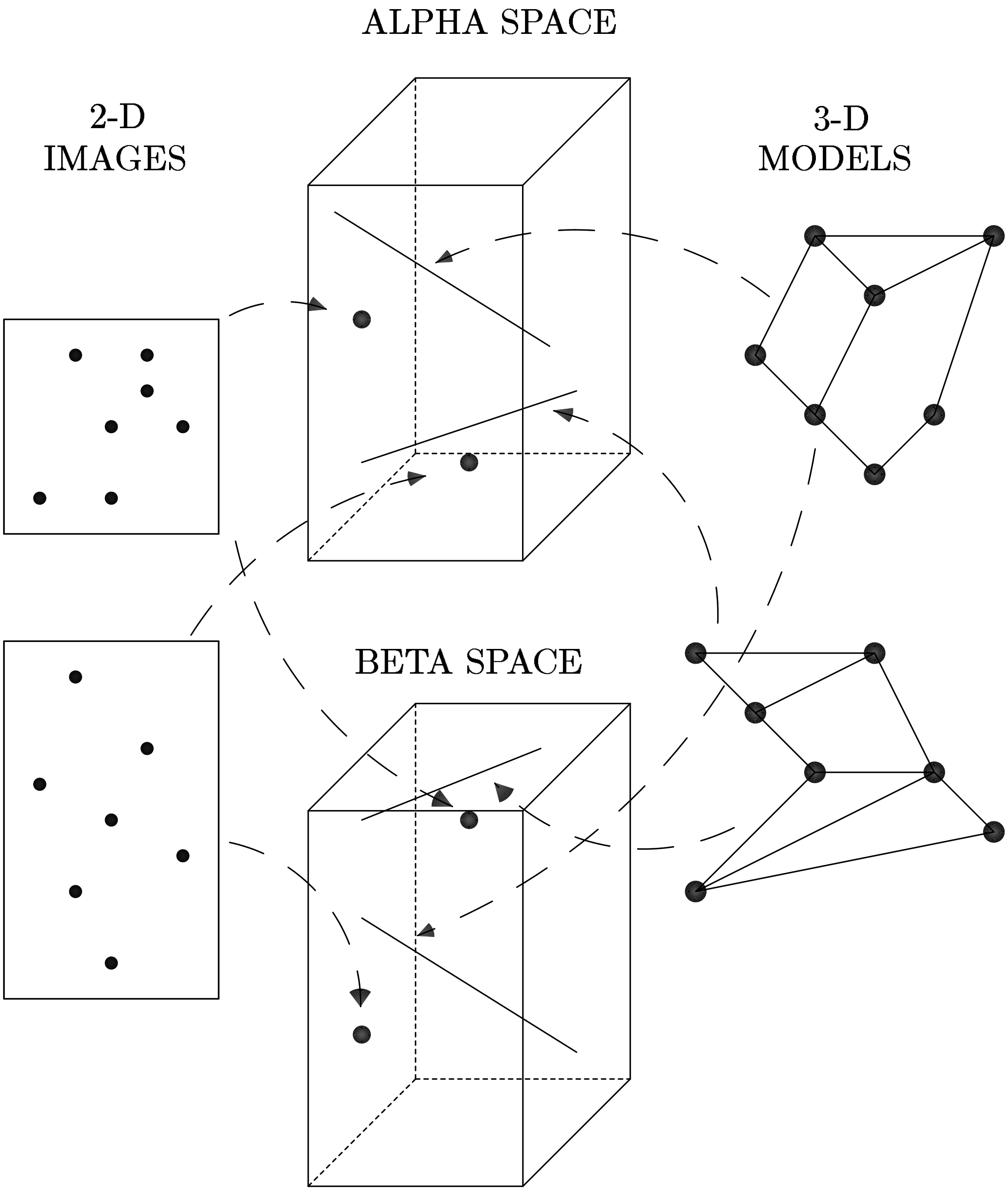


Figure 2.3: Later in this chapter we show how to decompose image space into two orthogonal parts. Each image corresponds to a point in each of these subspaces. Each model corresponds to a line in each subspace.

can explore different trade-offs. In particular, in chapter 5 we will consider the extent to which a lower-dimensional representation may be achieved by allowing some false positives or false negatives in our representation. In this chapter we determine the lowest-dimensional representation we can get when no such compromises are allowed.

We derive a number of lower bounds on the dimensionality of the manifolds required to represent a model's images. We show that assuming scaled orthographic projection, a 2-D manifold is required to represent these images for models consisting either just of point features, or for models consisting of oriented points. These bounds are independent of the image space that we use to represent a model's images, as long as the choice of image space does not introduce errors into the representation. Under perspective projection, a 3-D manifold is needed. We also show that if an object has rotational degrees of freedom this will increase the dimensionality of the manifolds. An object with a single part that can rotate about an axis will produce images that form a 3-D manifold, assuming scaled orthographic projection. Each additional rotating part will add another dimension to the model's manifold. If we think in terms of discretely representing these manifolds in a lookup table, we see that even in the simplest case a good deal of space is required, and that this problem becomes inherently more difficult to manage as our recognition task becomes more complicated.

We then show that for an important case, we can achieve a much simpler result by decomposing our image space into two orthogonal subspaces. We may represent models consisting of rigid point features with pairs of 1-D manifolds in these two spaces, as depicted in figure 2.3. Moreover, these manifolds are just lines. This is a big improvement because the amount of space required to represent 1-D manifolds discretely is much less than the amount required to represent a 2-D manifold. But just as significant is the fact that we achieve a very simple formulation of the problem of matching a 2-D image to a 3-D scene. We translate this into the problem of matching a point in a high-dimensional space to lines in this space. This provides us with a conceptually simple geometric interpretation of the matching problem, which leads to many of the theoretical results discussed in chapters 3 and 5.

In considering the images that a model can produce, we assume that our model consists of just isolated features. We therefore ignore the fact that a real object will occlude some of its own features from some viewpoints, and we focus on the different possible geometric configurations that model features may produce in an image. Work on aspect graphs takes exactly the opposite point of view, and determines which collections of model features may be viewed unoccluded from a single viewpoint while ignoring changes in the particular images that any given collection of features can produce. Work on aspect graphs is therefore complementary to and orthogonal to the work presented in this chapter. Some recent work on aspect graphs can be found in Gigus, Canny and Seidel[45], Kriegman and Ponce[69], and Bowyer and Dyer[15].

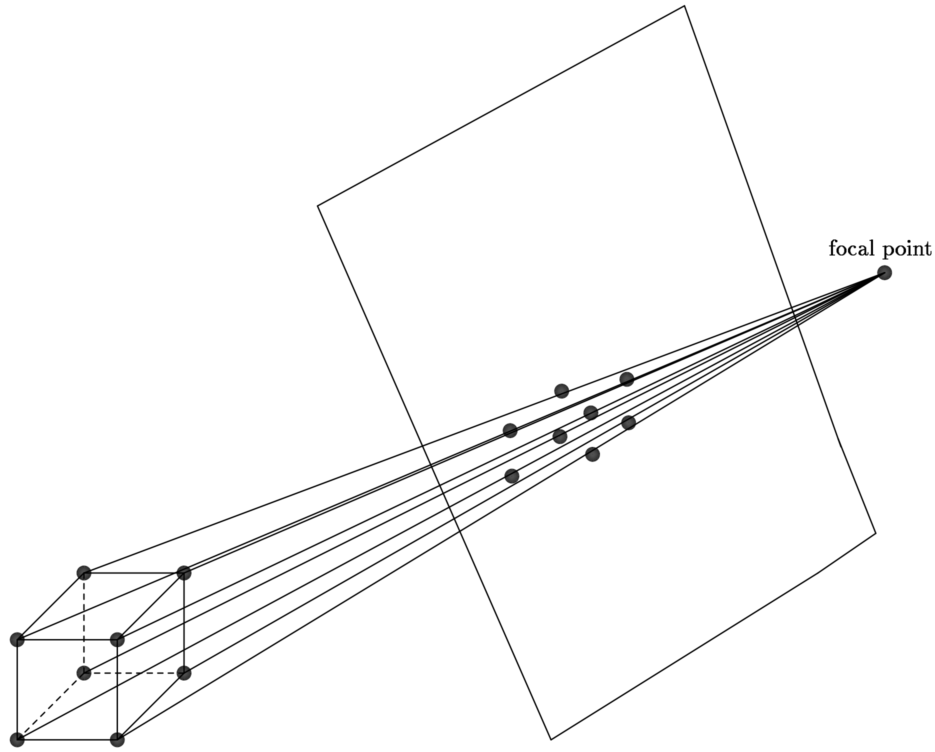


Figure 2.4: With perspective projection, lines connecting the model points and corresponding image points all meet at a single focal point.

2.2 Projection Transformations

We consider four different kinds of projection transformations in this chapter. Of these, perspective projection is the most realistic model of the way that cameras take photographs, and of the way that images are projected onto the human eye. However, for mathematical simplicity, we will use scaled orthographic projection as an approximation to perspective projection. We will also use a linear transformation that is even more convenient mathematically, and that we will show is a good approximation to the process of photographing an object and then viewing this photograph. Similarly, a projective transformation captures the process of imaging a planar object with perspective projection, then viewing the photograph. This section will describe and compare these projection models, but see Horn[53] for further details on the first two.

2.2.1 Perspective Projection

To describe perspective projection, suppose that we have a focal point lying behind an image plane. 3-D points in the world that are on the other side of the plane are imaged. The place where a line from the focal point to a scene point intersects the image plane indicates the location in the image where this scene point will appear. Figure 2.4 illustrates this. We will make use of only this geometric description of perspective projection, and will not need to model it mathematically.

We will only be considering the case where the focal length is considered one of the variables of the projection. In that case, the transformation has seven degrees of freedom because in addition to the focal length there are also six degrees of freedom in the location of the object relative to the camera. Assuming that we do not know the focal length is equivalent to assuming that we do not know the scale of the model we are trying to recognize because the whole world may be scaled without altering the picture. So if we want to characterize the set of images that a model of variable size might produce, whether or not we know the camera's focal length we may assume that the object size is fixed and the focal length is unknown.

2.2.2 Projective Transformations

We will also consider projective transformations of planar models. A projective transformation consists of a series of perspective projections. That is, a planar model, m, can produce an image, i, if there exists some intermediate series of images, $i_1, i_2, ...i_n$ such that i_1 is a perspective image of m, i_2 is a perspective image of i_1, ..., and i is a perspective image of i_n, as shown in figure 2.5. These images may be taken with any focal length.

Geometers have long studied projective transformations, and there are many books on the subject (see Tuller[102] for an introduction). Projective transformations are of interest because they are related to perspective projection, and at the same time, they form a group of transformations. Analytic formulations of projective transformations are well known, but for our purposes the simple geometric definition given above will suffice, along with a few facts that we will state later. Although we will not prove this here, it follows from the analytic formulation of projective transformations that they have eight degrees of freedom, and that, except in degenerate cases, there exists a projective transformation that will map any four model points to any four image points.

2.2.3 Scaled Orthographic Projection

Scaled orthographic projection provides a simple approximation to perspective projection. With this method, each scene point is projected orthogonally from the world onto the image plane, as shown in Figure 2.6. The resulting image can then be scaled

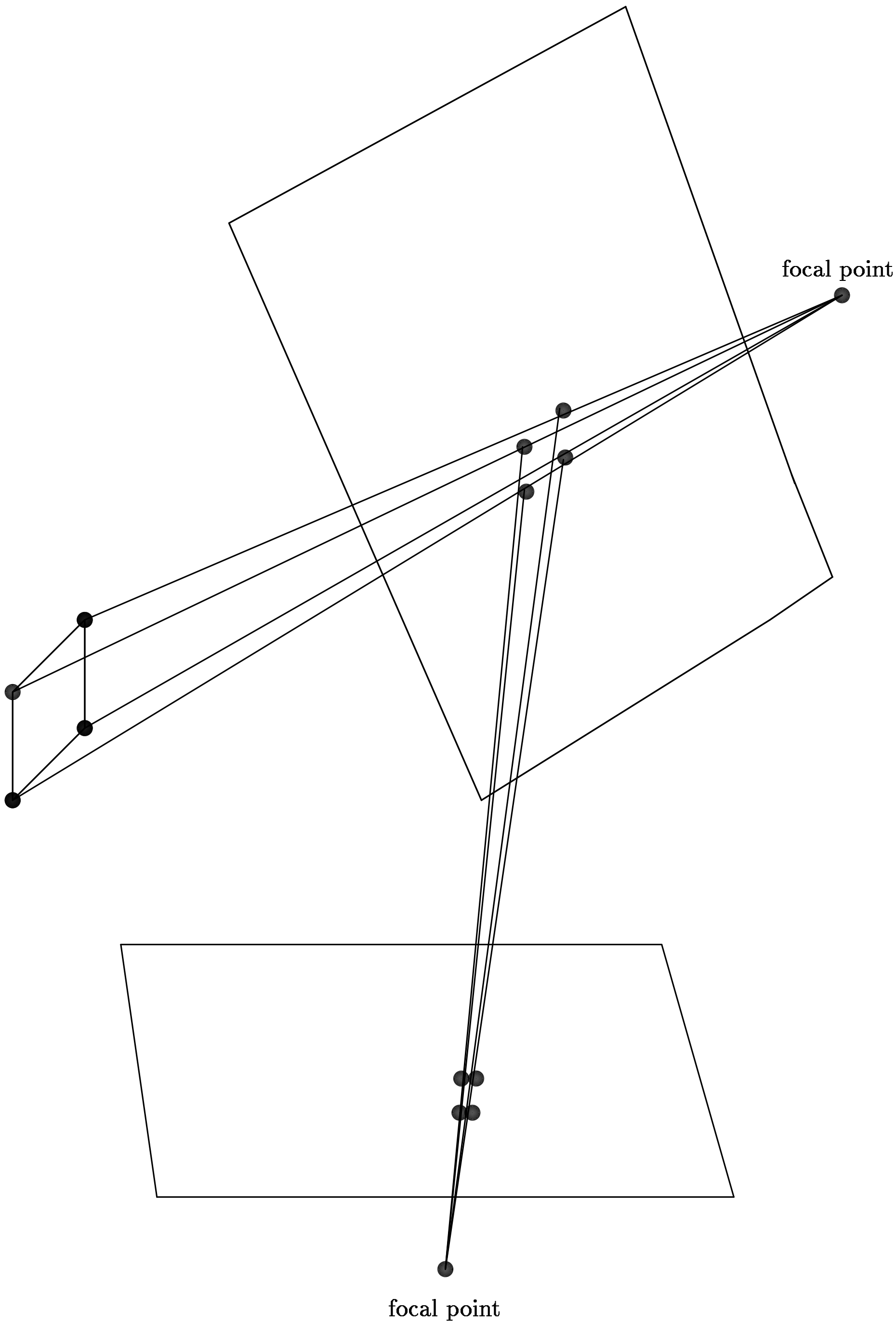


Figure 2.5: A projective transformation combines a series of perspective projections. We must begin with a planar model.

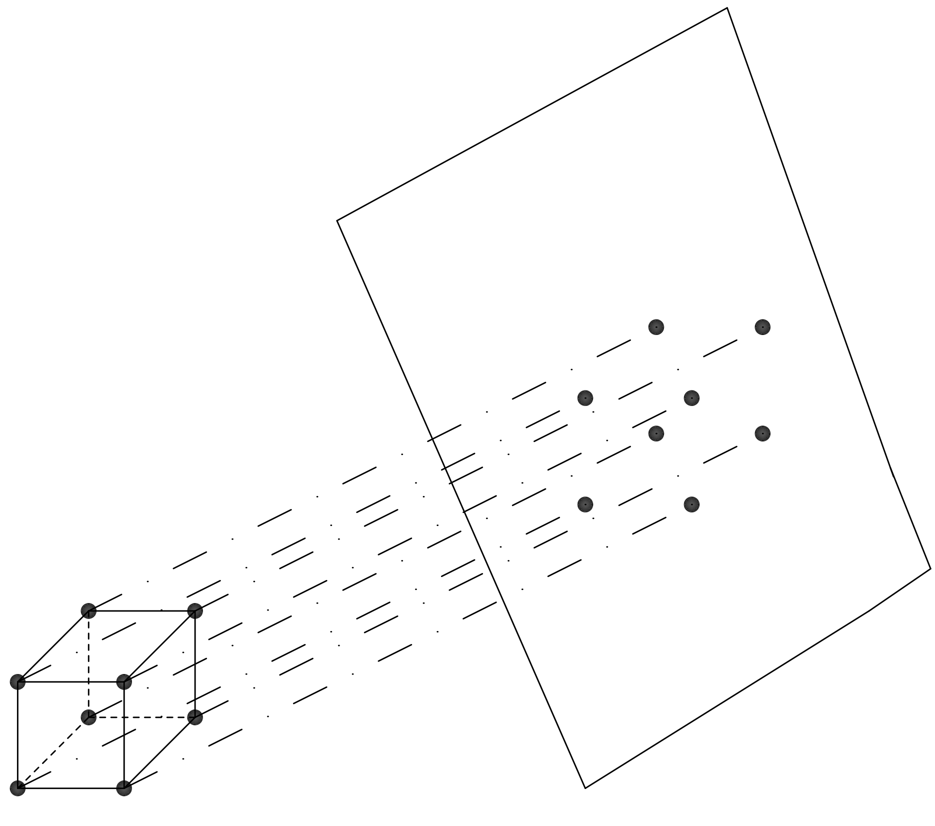


Figure 2.6: With orthographic projection, parallel lines connect the 3-D model points with the corresponding image points.

arbitrarily, to account for the change in the perceived size of an object as its distance varies. It is convenient to think of this projection as first viewing the object from any point on a surrounding viewing sphere, projecting the object in parallel onto an image plane that is tangent to the sphere at the viewpoint, and then arbitrarily scaling the resulting image, and rotating and translating it in the plane. Orthographic projection does not capture perspective distortion; for example 3-D parallel lines always project to 2-D parallel lines, while in perspective projection parallel lines converge as they recede from the viewer. However, orthographic projection is a perfect approximation to perspective projection when every scene point is equally far from the focal point, and is a good approximation as long as the relative depth of points in the scene is not great compared to their distance from the viewer. Orthographic projection has only six degrees of freedom, that is, an image depends only on the relative location of the object with respect to the image plane.

To describe orthographic projection mathematically, we assume that the image plane is the plane $z = 0$, and that the object is positioned arbitrarily, which is equivalent to assuming that the object is fixed and that we view it from an arbitrary viewpoint. To describe the image point produced by a 3-D scene point, $\mathbf{p} = (p_x, p_y, p_z)$, we assume that the point is arbitrarily rotated and translated in 3-D, and then projected onto the image plane and scaled. We may describe rotation with the rotation matrix $\mathbf{R}$, translation with the vector $\mathbf{t}$, and scaling with the scalar s. Assume that:

$$\mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \qquad \mathbf{t} = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

Multiplying $\mathbf{R}$ by a scene point is equivalent to applying a rigid rotation to that point as long as each row of $\mathbf{R}$ has unit magnitude, and as long as these rows are orthogonal. Projecting the rotated and translated model into the image is equivalent to removing the model's z component. Let $\mathbf{q} = (q_x, q_y)$ be the image point produced by $\mathbf{p}$. We find:

$$\begin{pmatrix} i_x \\ i_y \end{pmatrix} = s\left(\begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \end{pmatrix}\begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}\right)$$

One reason that scaled orthographic projection is mathematically convenient is that for planar models it is equivalent to a 2-D affine transformation. We express a 2-D affine transformation as an unconstrained $2x2$ matrix along with a translation vector. If our model is planar, we may assume that it lies in the plane $z = 0$, and that each model point is expressed $\mathbf{p} = (p_x, p_y)$. Then, letting:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

and letting $\mathbf{v} = (v_x, v_y)$, when we view our model from any viewpoint, assuming scaled orthographic projection, there exists some $\mathbf{A}$ and $\mathbf{v}$ such that:

$$\begin{pmatrix} q_x \\ q_y \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} p_x \\ p_y \end{pmatrix} + \begin{pmatrix} v_x \\ v_y \end{pmatrix}$$

for every model point, and for any $\mathbf{A}$ and $\mathbf{v}$ there exists some viewpoint of the planar model that produces the same image as this affine transformation. See Huttenlocher and Ullman[57] for discussion of this fact. This is very convenient, because in general orthographic projection is not a linear transformation, due to the constraints that the rotation matrix must meet. However, the fact that this projection is equivalent to an affine transformation in the planar case means that in that case it is a linear transformation.

2.2.4 Linear Transformations

To gain the advantages of linearity for 3-D models, we can generalize orthographic projection by removing the constraints that make $\mathbf{R}$ a rotation matrix. That is, we allow:

$$\mathbf{S} = \begin{pmatrix} s_{11} & s_{12} & s_{13} \\ s_{21} & s_{22} & s_{23} \end{pmatrix}$$

to be an arbitrary $3x2$ matrix, $\mathbf{u} = (u_x, u_y)$ to be an arbitrary translation vector and let:

$$\begin{pmatrix} q_x \\ q_y \end{pmatrix} = \left(\left(\begin{pmatrix} s_{11} & s_{12} & s_{13} \\ s_{21} & s_{22} & s_{23} \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} \right) + \begin{pmatrix} u_x \\ u_y \end{pmatrix} \right)$$

This is more general than scaled orthographic projection, because any image that can be created with a scaled orthographic projection can also be created with this transformation. To see this, for any $\mathbf{R}$, $\mathbf{t}$ and s that define a scaled orthographic projection, we just let $\mathbf{S} = s\mathbf{R}$ and let $\mathbf{u} = \mathbf{t}$, and we get an equivalent linear transformation. The reverse is not true. In fact, the linear transformation has eight degrees of freedom, while scaled orthographic projection has six degrees of freedom. This means that when we describe the manifold of images that a model produces with this linear transformation, we are describing a superset of the set of images the model could produce with scaled orthographic projection.

Due to its mathematical simplicity, this transformation has been used for a number of studies of object recognition and motion understanding (Lamdan and Wolfson[71], Ullman and Basri[105], Koenderink and Van Doorn[65], Shashua[95], Cass[26],[27], Breuel[19]). We now show a new result about this transformation, that it characterizes the set of images that can be produced by a photograph of an object. We will use this result both to understand the transformation better, and to produce a new, useful representation of this transformation.

Suppose we form an image of a 3-D model using scaled orthographic projection and then view this 2-D image from a new viewpoint, modeling this second viewing also as a scaled orthographic projection. Then this second projection is equivalent to applying an affine transformation to the model's image, as we have explained above.

We now show that the set of images produced by such pairs of transformations is equivalent to the set of images produced by a linear transformation of the model. To show this, we must show that for any $s, \mathbf{A}, \mathbf{R}, \mathbf{t}, \mathbf{v}$ there exist $\mathbf{S}, \mathbf{u}$ such that:

$$\mathbf{Sp} + \mathbf{u} = \mathbf{A}(s\mathbf{Rp} + \mathbf{t}) + \mathbf{v}$$

and similarly, that for any $\mathbf{S}, \mathbf{u}$ there exist $s, \mathbf{A}, \mathbf{R}, \mathbf{t}, \mathbf{v}$ so that this equality holds. The first direction of this equivalence is shown by letting $\mathbf{S} = \mathbf{A}s\mathbf{R}$ while letting $\mathbf{u} = \mathbf{At} + \mathbf{v}$. To show the second direction of the equivalence, we can let $\mathbf{v} = \mathbf{u}$, $\mathbf{t} = \mathbf{0}$ and $s = 1$. We must then show only that for any $\mathbf{S}$ there exist some $\mathbf{A}, \mathbf{R}$ such that $\mathbf{AR} = \mathbf{S}$. Let $\mathbf{S_1}, \mathbf{S_2}$ stand for the top two rows of $\mathbf{S}$ and let $\mathbf{R_1}, \mathbf{R_2}$ stand for the top two rows of $\mathbf{R}$. Also, let $a_{11}, a_{12}, a_{21}, a_{22}$ denote the elements of $\mathbf{A}$. That is, let:

$$\mathbf{S} = \begin{pmatrix} \mathbf{S_1} \\ \mathbf{S_2} \end{pmatrix} \qquad \mathbf{R} = \begin{pmatrix} \mathbf{R_1} \\ \mathbf{R_2} \end{pmatrix} \qquad \mathbf{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

Then:

$$\mathbf{AR} = \begin{pmatrix} a_{11}\mathbf{R_1} + a_{12}\mathbf{R_2} \\ a_{21}\mathbf{R_1} + a_{22}\mathbf{R_2} \end{pmatrix}$$

The condition, then, for finding $\mathbf{A}$ and $\mathbf{R}$ such that

$$\mathbf{AR} = \begin{pmatrix} \mathbf{S_1} \\ \mathbf{S_2} \end{pmatrix}$$

is that we can choose $\mathbf{R_1}$ and $\mathbf{R_2}$ so that we can express both $\mathbf{S_1}$ and $\mathbf{S_2}$ as linear combinations of $\mathbf{R_1}$ and $\mathbf{R_2}$. If we think of $\mathbf{R_1}, \mathbf{R_2}, \mathbf{S_1}, \mathbf{S_2}$ as points in a three-dimensional space, then the origin, $\mathbf{S_1}$, and $\mathbf{S_2}$ define a plane, so we may choose $\mathbf{R_1}$ and $\mathbf{R_2}$ to be any two orthonormal vectors in that plane, and they will span it. So there will be some linear combination of the two $\mathbf{R}$ vectors equal to each of the $\mathbf{S}$ vectors. We have shown that the set of images that can be formed by a linear transformation is the same as the set that can be formed by scaled orthographic projection followed by an affine transformation. In fact, this latter representation of the projection is redundant, since both scaled orthographic projection and an affine transformation allow for translation and rotation in the plane, and scaling. So we may more simply think of this as orthographic projection from some point on the viewing sphere, followed by an affine transformation, folding all of the planar transformations into the affine transformation.

This result suggests an hypothesis about the human ability to understand photographs. The fact that people have no difficulty in recognizing objects in photographs is somewhat puzzling, because a photograph of an object produces an image that the object itself could never produce. Yet we hardly even notice the distortion that results when we view a photograph from a position that is different from that of the camera that took the photograph. Figure 2.7 shows a photograph, and an affine transformation of it, as an example.

Cutting[37] discusses this phenomenon at greater length. He suggests that people rely heavily on projective invariants for image understanding, that is, on the properties of a planar object that do not vary as the object is viewed from different directions, assuming perspective projection. Such descriptions would be the same for both a view of a planar object, and for a view of a photograph of the object. Cutting, however, does not consider nonplanar objects. In this chapter, we show that considerable mathematical convenience is gained by considering linear transformations as a projection model from 3-D objects to 2-D images. We have just shown that a side-effect of this model is that a characterization of an object's images will include the images produced by photographs of the object. This suggests that human ability to interpret photographs may result from the fact that considerable computational simplicity is gained when one assumes that objects can also produce the images that could really only come from their photographs.

2.2.5 Summary

We can see that there are a variety of transformations available for modeling the projection from a model to an image. While perspective projection is the most accurate, scaled orthographic projection frequently provides a good approximation to it. Scaled orthographic projection can also be much more convenient to work with mathematically, particularly when models are planar. The fact that it requires six degrees of freedom, while perspective projection with an unknown focal length requires seven degrees of freedom suggests that the set of images produced by scaled orthographic projection will be smaller, making them easier to characterize at the potential cost of missing some of the images that a model can produce.

We also see that even greater simplicity can be achieved by also considering the images that a photograph of an object produces. In the case of scaled orthographic projection this leads to a linear projection model. When we begin with a planar model and project it repeatedly using perspective projection we have a projective transformation. These have been studied extensively by mathematicians. In both cases, we expand the set of images that we consider a model capable of producing to

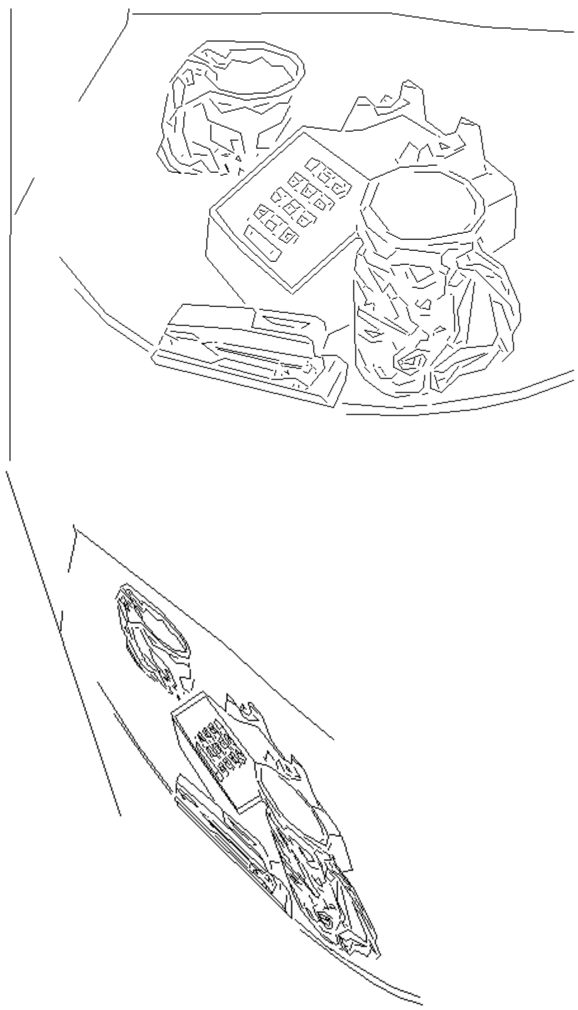


Figure 2.7: On top are edges describing a scene. Below, these edges after applying a fairly arbitrary affine transformation. Although the edges appear somewhat distorted, the objects in the scene are still recognizable. It is plausible that essentially the same perceptual strategies are used to understand both images. We hypothesize that in two images like these, the same image lines are grouped together, and the same basic properties of these groups are matched to our internal representation of the telephone in order to recognize it.

achieve greater simplicity.

This section has largely been a review of material that is well described in many places in the computer vision literature. However, it is our contribution to show that a linear projection model describes the images produced by a photograph of a model, assuming scaled orthographic projection. This provides a more intuitive understanding of the images produced with the linear projection model, which others have previously used.

2.3 Minimal Representations

This section contains a variety of results about representations of a model's images, but we can conveniently divide these results into two parts. First, in the case of a linear transformation and models with 3-D point features, we show that each model can be represented by a pair of lines in two orthogonal spaces. This result is really the centerpiece of this thesis. It is used to derive a variety of theoretical results in chapters 3 and 5, and it forms the basis of a useful indexing system. However, this is not the best representation imaginable, and so we lead up to this result with a series of negative results that show that representations that might be preferable are in fact impossible to achieve.

Then we consider the case of oriented point features, that is point features that have one or more directional vectors associated with them, and the case of articulated objects. These types of models are of practical value, and they are also interesting because they turn out to be fundamentally harder than the case of rigid point features. These results will form the basis for negative bounds on the space complexity of indexing, as well as for negative results about other approaches to recognition. As objects grow more complex, we will see that existing approaches to recognition grow inherently more complex.

2.3.1 Orthographic and Perspective Projection

Planar Models

We begin by reviewing existing methods of representing the images produced by planar models when they are viewed from arbitrary 3-D positions. This discussion will serve several ends. It will allow us to present a clearly optimal solution to the image representation problem for an interesting special set of models. It will recast some existing results in our more general framework. And it will allow us to introduce some specific mathematical results that will be used later.

An optimal solution to the image representation problem may exist when an *invariant* description of models exists, and much is known about invariants of planar models from classical mathematics. Tuller[102] describes some of this work from a mathematical perspective for the case of models consisting of planar points or lines. For our purposes, we may define an invariant description as a function of the images that, for any model, is constant for all images of that model.

To express this more formally, we will introduce some useful definitions and notations. We will let $\mathcal{M}$ stand for the set of all possible models, where a model is a particular group of features. The type of features should always be clear from context. Similarly, $\mathcal{I}$ stands for the set of all comparable images, and $\mathcal{T}$ for the set of all transformations. For example, for planar models under orthographic projection, an element of $\mathcal{T}$ will be a particular affine transformation. So an element of $\mathcal{T}$ is a function from $\mathcal{M}$ to $\mathcal{I}$. We will use i, t, m as variables indicating elements of their corresponding sets, and we will use $\mathbf{p_i}$ to stand for a model feature, and $\mathbf{q_i}$ to stand for the corresponding image feature. So for example, "$\exists t$ such that $i = t(m)$" means that i is a possible image of m.

Definition 2.1 *f, a function on $\mathcal{I}$, is an* **invariant** *function if $\forall m \in \mathcal{M}, \forall t_1, t_2 \in \mathcal{T}, f(t_1(m)) = f(t_2(m))$.*

That is, an invariant function produces the same value when applied to any image of an object. It is a property of the model's images that does not vary as our viewpoint varies. (This is the traditional mathematical definition of invariance, specialized to our domain, with the assumption that we are only concerned with invariants of weight 0).

Definition 2.2 *We call f a* **non-trivial invariant** *function if it is an invariant function, and $\exists i_1, i_2 \in \mathcal{I}, i_1 \neq i_2$ such that $f(i_1) \neq f(i_2)$.*

That is, if f is not just a constant function.

Definition 2.3 *We call f a* **complete invariant** *function if, $\forall i_1, i_2 \in \mathcal{I}$, $\forall m \in M$, if $f(i_1) = f(i_2)$ and if $\exists t_1 \in \mathcal{T}$ such that $i_1 = t_1(m)$ then $\exists t_2 \in \mathcal{T}$ such that $i_2 = t_2(m)$.*

That is, not only do all images of a model have the same value for f, but any image that has such a value for f must be a possible image of the model.

When an invariant function, f, exists, we can use it to define a representation of images in which each model's manifold is a single point. Our image space is just the range of f, that is, we let f be a mapping from images to image space. By the definition of an invariant, all of a model's images are then mapped to the same point in image space. If f is a complete invariant function that is continuous, then it provides us with a perfect representation of images. In addition to meeting all our other criteria, f will introduce no false positive errors, because only images that a model could have produced will be mapped to that model's representation in image space.

Planar point models with scaled orthographic projection

We begin by showing a complete invariant function for models consisting of planar points when these models are viewed from arbitrary 3-D viewpoints with scaled orthographic projection. As we have noted, in this case projection may be modeled as a 2-D affine transformation. This invariant is known from classical geometry, and has been used for computer vision by Lamdan, Schwartz and Wolfson[70]. Our discussion of it is modeled on their presentation.

Suppose our model consists of at least four points: $\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}, ...\mathbf{p_n}$. Assuming that the first three points are not collinear we may use them to define a new coordinate system for the plane, and represent the remaining points in this coordinate system. That is, we define the origin, $\mathbf{o}$, and two axes $\mathbf{u}, \mathbf{v}$ as:

$$\mathbf{o} = \mathbf{p_1} \qquad \mathbf{u} = \mathbf{p_2} - \mathbf{p_1} \qquad \mathbf{v} = \mathbf{p_3} - \mathbf{p_1}$$

Then we describe the i'th point with the *affine coordinates* (α_i, β_i). These coordinates describe the vector from $\mathbf{o}$ to $\mathbf{p_i}$ by its components in the directions $\mathbf{u}, \mathbf{v}$. That is:

$$\mathbf{p_i} - \mathbf{o} = \alpha_i \mathbf{u} + \beta_i \mathbf{v}$$

Lemma 2.1 *The set of affine coordinates that describe an image are an invariant function.*

Proof: Let the $2x2$ matrix $\mathbf{A}$, and the 2-D vector $\mathbf{t}$ define an affine transformation, and let $\mathbf{o}', \mathbf{u}', \mathbf{v}'$, the transformed version of our original basis, define a new basis in the transformed image. That is we let: $(\mathbf{o}', \mathbf{u}', \mathbf{v}') = (\mathbf{q_1}, \mathbf{q_2} - \mathbf{q_1}, \mathbf{q_3} - \mathbf{q_1})$, and then describe other transformed image points using this as a coordinate system. Then:

$$\begin{aligned}
\mathbf{q_i} &= \mathbf{A}\mathbf{p_i} + \mathbf{t} \\
&= \mathbf{A}(\mathbf{o} + \alpha_i \mathbf{u} + \beta_i \mathbf{v}) + \mathbf{t} \\
&= \mathbf{A}(\mathbf{p_1} + \alpha_i(\mathbf{p_2} - \mathbf{p_1}) + \beta_i(\mathbf{p_3} - \mathbf{p_1})) + \mathbf{t} \\
&= \mathbf{A}\mathbf{p_1} + \alpha_i \mathbf{A}(\mathbf{p_2} - \mathbf{p_1}) + \beta_i \mathbf{A}(\mathbf{p_3} - \mathbf{p_1}) + \mathbf{t} \\
&= \mathbf{o}' + \alpha_i \mathbf{u}' + \beta_i \mathbf{v}'
\end{aligned}$$

So affine coordinates are not changed when the model is viewed from a new angle, and constitute an invariant function.

Lemma 2.2 *Given any 3 non-collinear model points:* $(\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3})$ *and any non-collinear image points:* $(\mathbf{q_1}, \mathbf{q_2}, \mathbf{q_3})$*, there exists an affine transformation that maps the model points to the corresponding image points.*

Proof: The equations:

$$\mathbf{q_i} = \mathbf{A}\mathbf{p_i} + \mathbf{t}$$

give us six linear equations with six unknowns. The condition for these equations having a solution is equivalent to the points not being collinear.

Theorem 2.3 *The set of affine coordinates that describe an image are a complete invariant function.*

Proof: This requires us to show that any image with the same affine coordinates as a model could have been produced by that model. The image is fully described by $(\mathbf{q_1}, \mathbf{q_2}, \mathbf{q_3}, \alpha_4, \beta_4, ..., \alpha_n, \beta_n)$. We know from lemma 2.2 that the model could produce an image with any three points $\mathbf{q_1}, \mathbf{q_2}, \mathbf{q_3}$. From lemma 2.1 the model's affine coordinates are always preserved. Therefore, a model can produce any image that is described by the same affine coordinates as the model. □

This result will be quite useful to us in future sections. It has also been used extensively for the recognition of planar models. This was first done by Lamdan, Schwartz and Wolfson[70]. Their system computes the affine coordinates of quadruples of model points at compile time, and stores a pointer to each quadruple in a 2-D image space that represents (α_4, β_4). Then, at run time they perform matching by computing the affine coordinates of an image quadruple, and then a simple table lookup provides all model quadruples that could have produced this image quadruple. They combine these lookups using a voting scheme that we will not describe here. Affine coordinates are also considered for recognizing planar objects in: Lamdan and Wolfson[71], Costa, Haralick and Shapiro[34], and Grimson, Huttenlocher and Jacobs[49].

Planar point models with perspective projection or projective transformations

In this section we will present a complete invariant representation of points under projective transformations. This means that the representation is not changed by a series of perspective projections, so obviously this will also be an invariant representation for perspective projection, although not a complete one.

We begin by describing the *cross-ratio*. This is a function of four collinear points that is invariant under projective transformations. We will use it to build up a more general invariant description of coplanar points.

Definition 2.4 *Let A, B, C, D be four collinear points. Let $\|AB\|$ denote the distance between A and B. Then the value:*

$$\frac{\frac{||CA||}{||CB||}}{\frac{||DA||}{||DB||}}$$

is the **cross-ratio** *of the four points.*

An important theorem from projective geometry is:

Theorem 2.4 *The cross ratio of four points is invariant under projective transformations.*

We omit the proof of this theorem (see Tuller[102], for example). However, we note that this theorem depends on the fact that a projective transformation preserves the collinearity of points.

We now present an invariant representation of five general planar points. Call these points: $\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}, \mathbf{p_5}$. Let L_{ij} stand for the line that connects $\mathbf{p_i}$ and $\mathbf{p_j}$. Let $\mathbf{p'_1}$ be the point at the intersection of lines L_{12} and L_{34}. Let $\mathbf{p'_2}$ be the point at the intersection of L_{12} and L_{45}. See figure 2.8. Suppose a projective transformation maps each point $\mathbf{p_i}$ to a corresponding point $\mathbf{q_i}$. Note that because projective transformations preserve collinearity, for any such transformation $\mathbf{q'_1}$ will still be at the intersection of the line formed by $\mathbf{q_1}$ and $\mathbf{q_2}$, and the line formed by $\mathbf{q_3}$ and $\mathbf{q_4}$. A similar statement holds for $\mathbf{q'_2}$. Therefore, given our five initial points, we may compute the cross-ratio of the points $\mathbf{p_1}, \mathbf{p_2}, \mathbf{p'_1}, \mathbf{p'_2}$, and this cross-ratio will be invariant under projective transformations. We will call this invariant γ_5. Similarly, we locate $\mathbf{p'_3}$ at the intersection of L_{13} and L_{25}, and $\mathbf{p'_4}$ at the intersection of L_{13} and L_{45}, and compute a second invariant from the cross-ratio of the points $\mathbf{p_1}, \mathbf{p_3}, \mathbf{p'_3}, \mathbf{p'_4}$. We call this invariant ξ_5. Together, (γ_5, ξ_5) form a complete invariant description of the five points, but we will not prove the completeness of the description here. We will call these the *projective coordinates* of the fifth point.

To handle models with more than five points, we may substitute any i'th point for the fifth point in the above calculations, and compute (γ_i, ξ_i).

Other planar invariants

There has also been a good deal of work done on invariants of planar objects that are more complicated than points. There are general, powerful mathematical tools for deriving invariants of curves under transformations that form a group. However, there is not always a clear, computationally tractable means of determining these invariants. So some work has focused on deriving useful invariants in specific situations. There are also many practical problems that must be solved in order to use these invariants in vision. In particular, a real image must be turned into a mathematical object. For example, we must find derivatives of an image curve, or approximate it with an algebraic curve. Work has been done to understand and limit the effects of sensing error on these processes. We do not wish to describe this work in detail here because it is not directly relevant to what follows. Instead, we provide a brief overview so that the interested reader will know where to find more detailed presentations.

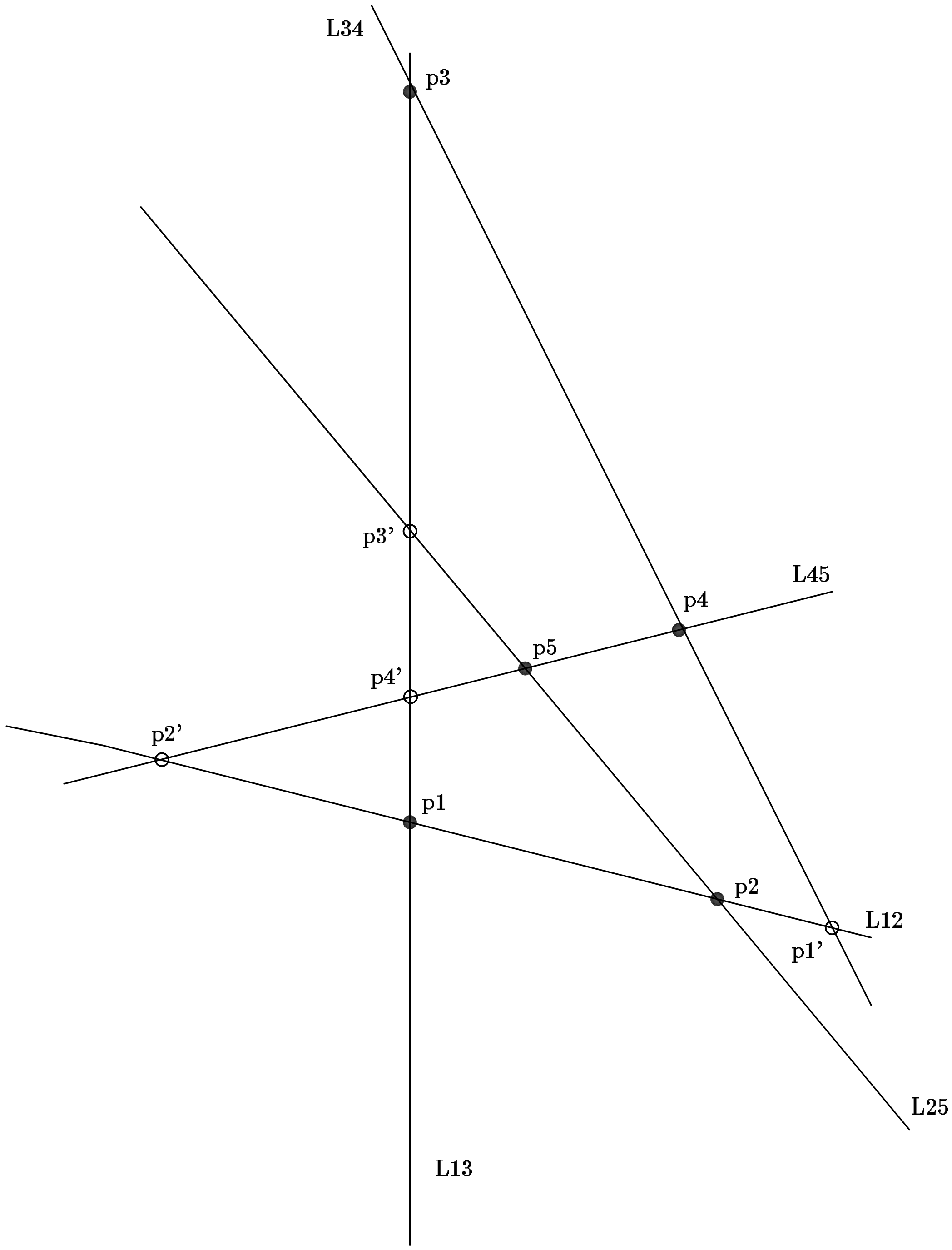


Figure 2.8: Points used to construct a projective invariant description.

Cyganski and Orr[38] suggested the use of a new, affine-invariant curvature. Cyganski, Orr, Cott and Dodson[39] use this to match a model which is a closed curve to its image by comparing a normalized graph of curvature versus length. In the error free case the graph of the image curvature is always identical to the graph of the model's curvature, except for 1-D shifting due to the fact that one does not have a canonical starting point for the curve.

Van Gool, Kenpenaers and Oosterlinck[106] provide a synthesis of these curvature invariants and the point invariants used by Lamdan, Schwartz and Wolfson. They show how information about the position and derivatives of points on a curve may be combined into a single invariant representation.

Weiss[111] suggests the use in machine vision of a large body of results concerning projective invariants of plane curves. Since perspective projection and scaled orthographic projection are special cases of projective transformations, Weiss' discussion applies to both cases. He provides a useful review of classical work on differential invariants, that is invariants based on local properties of a curve such as derivatives, and on algebraic invariants, that is, invariants of algebraic curves. Weiss also makes many specific suggestions for applying this classical work to problems in visual object recognition. This paper has played an influential role in bringing mathematical work on invariants to the attention of computer vision researchers.

In a more recent paper, Weiss[112] has attempted to come to grips with the practical problems of using invariants in the face of noise and occlusion. He provides differential invariants that require taking the fourth derivative of a curve, compared to the sixteen derivatives required by classical results. He then suggests methods for robustly finding the fourth derivative of a curve in spite of image error.

Forsyth et al.[44] also provide a useful review of general mathematical methods for finding invariants of planar curves. They then derive projective invariants of pairs of conics and use these for object recognition. They also consider the problem of finding invariant methods of approximating image curves with algebraic curves. Rothwell et al.[92] describes some further application of these ideas.

Other applications of invariants to vision may be found in a recent collection, edited by Mundy and Zisserman[85]. Cutting[37] provides a more general discussion of invariants from the point of view of perceptual psychology.

A number of useful differential and algebraic invariants have been derived. There are two main challenges to applying these invariants to machine vision. First of all, the effects of error on these invariants are not generally well understood. There has been some work on the stability of these invariants, showing that small amounts of error have only a small effect on the invariant. But it is not understood how to precisely characterize the effects of realistic image error on invariant representations. This is in contrast to the case of planar points, where we know precisely how a bounded amount of sensing error can effect our invariant description (see chapter 4).

Invariants of local descriptors of curves use high-order derivatives, and so tend to be particularly sensitive to error. However, invariants of more global curve descriptions tend to be sensitive to occlusion. As a result, much of the work on invariants assumes that these curves have been correctly segmented. To handle this problem, work is needed either on segmenting curves, or on developing invariant descriptions that are insensitive to occlusion. Essentially, one could say that work has proceeded on curve indexing without much attention yet to the grouping problem.

3-D Point Models

Unfortunately, there are no invariants when models consist of arbitrary collections of 3-D points which are projected into 2-D images. That is, it is not possible to define an image space in which each model is described by a single point. In this section, we determine what is the lowest possible dimension of manifolds in image space that represent general 3-D point models, assuming that there are no false positive or false negative errors (beyond any introduced by the projection model). In chapter 5 we consider what happens when errors are introduced.

An alternative to this approach is the use of *model based invariants*. Weinshall[110] has shown that given a particular quadruple of 3-D point features, one may construct a simple invariant function. Applying this function to a quadruple of image points tells us whether they could be a scaled orthographic projection of the model points. This function is an invariant for a restricted set of one model, and a set of such functions may be combined to handle multiple models that do not share a common image.

Scaled orthographic projection: Manifolds must be 2-D

Clemens and Jacobs[32] show that in the case of general 3-D models of point features and scaled orthographic projection, models must be represented by a 2-D manifold in any image space. In this section we will draw extensively on that discussion, reworking it only slightly to fit our present context. Hence, this section should be considered as joint work between the author and David Clemens.

First, we add a definition and prove a lemma that will be useful for nonplanar models.

Definition 2.5 *For nonplanar models, the first three model points will define a plane. We call this the* **model plane**.

Lemma 2.5 *Given any nonplanar model, and any affine coordinates, (α_4, β_4), there is always a viewing direction for which* $\mathbf{q_4}$ *(the image of the fourth model point) has coordinates (α_4, β_4) relative to the first three image points.*

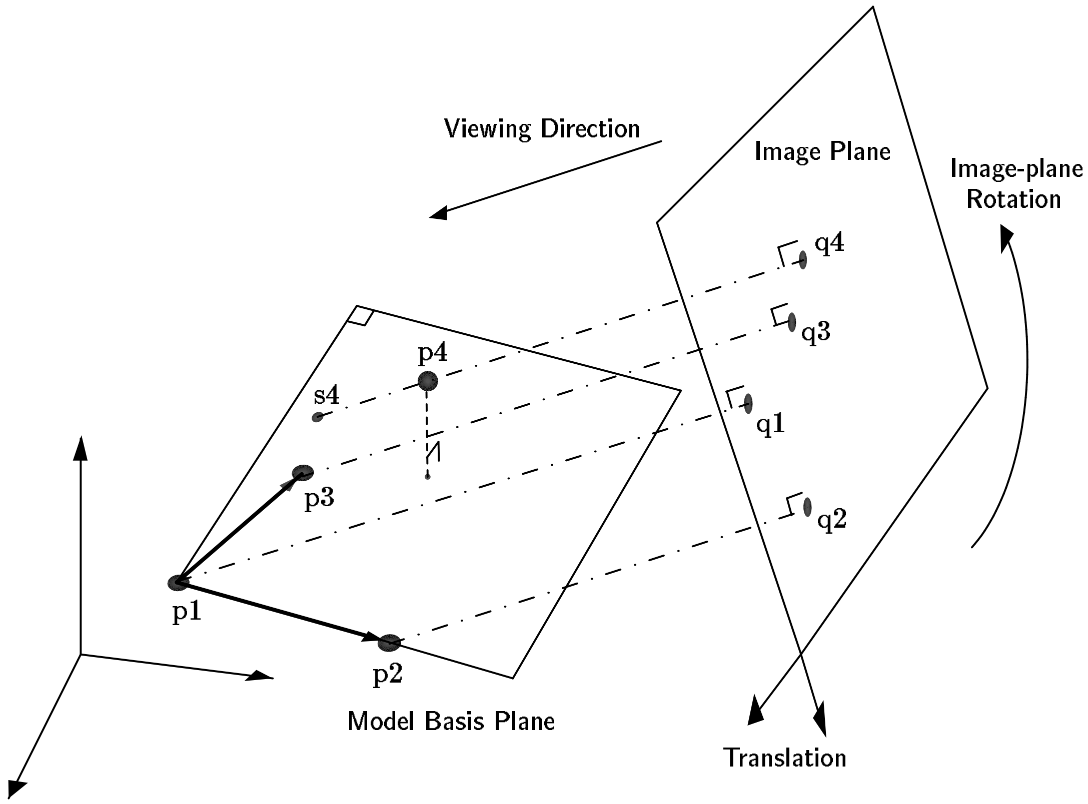


Figure 2.9: The image points $\mathbf{q_1}$, $\mathbf{q_2}$, $\mathbf{q_3}$, and $\mathbf{q_4}$ are the projections of the model points $\mathbf{p_1}$, $\mathbf{p_2}$, $\mathbf{p_3}$, and $\mathbf{p_4}$. The values of the image points depend on the pose of the model relative to the image plane. In the viewing direction shown, $\mathbf{s_4}$ and $\mathbf{p_4}$ project to the same image point. Note that $\mathbf{q_4}$ has the same affine coordinates as $\mathbf{s_4}$.

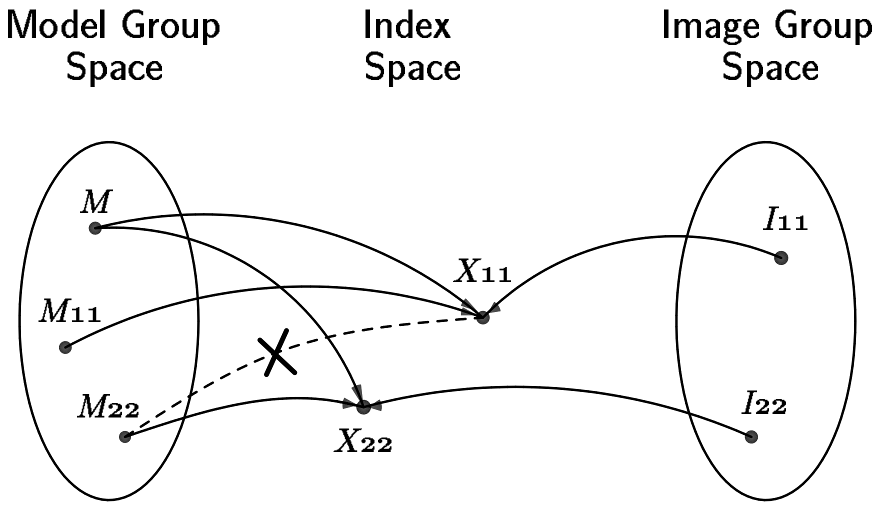


Figure 2.10: A model, m, can produce images $i_{1,1}$ and $i_{2,2}$, so its manifold must include the points these images map to in image space, $X_{1,1}$ and $X_{2,2}$. Model $m_{2,2}$ can also produce image $i_{2,2}$, so its manifold also includes $X_{2,2}$. Since $m_{2,2}$ could not produce $i_{1,1}$, its manifold must not include $X_{1,1}$. Therefore, $X_{1,1} \neq X_{2,2}$, and m's manifold must include at least two points.

Proof: Let $\mathbf{s_4}$ be the point in the model plane that has affine coordinates (α_4, β_4) with respect to the first three model points (see figure 2.9). When we view the model along a line joining $\mathbf{p_4}$ with $\mathbf{s_4}$, both $\mathbf{p_4}$ and $\mathbf{s_4}$ will project to $\mathbf{q_4}$. Since the projection of $\mathbf{s_4}$ will always have affine coordinates (α_4, β_4), $\mathbf{p_4}$ will also have these coordinates when looked at from this viewpoint. □

We may now show:

Theorem 2.6 *For any model, m, that has four nonplanar points, and any mapping from images to image space that does not introduce errors, m must correspond to a 2-D manifold in image space.*

Proof: First we show that for any nonplanar model there must be a one-to-one mapping from the points in the real plane to distinct points on the model's manifold. We choose any three points in m as basis points, that is as the points $(\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3})$ that will define the model plane and form an affine basis in it. We choose to denote as $\mathbf{p_4}$ any other model point that is not in the model plane. Then let θ be a variable representing the viewing direction, i.e. a point on the unit viewing sphere. We may let $(\alpha_4(\theta), \beta_4(\theta))$ stand for the affine coordinates of $\mathbf{q_4}$ (the projection of the fourth model point) as a function of θ.

We will first show that m's manifold must contain at least two points in image space, as illustrated in figure 2.10. Then we will generalize this to prove our theorem.

By lemma 2.5, there exists some θ_{11} such that $\alpha_4(\theta_{11}) = 1 = \beta_4(\theta_{11})$. Let $i_{(1,1)}$ stand for the entire projection of m when viewed from this orientation. That is, $i_{(1,1)}$ is the way the model appears when viewed so that its fourth point has the affine coordinates $(1,1)$. Let $X_{(1,1)}$ be the point in image space to which $i_{(1,1)}$ maps. Then $X_{(1,1)}$ must be part of m's manifold in image space, by our assumption that the manifolds must represent the models without error.

Let $m_{(1,1)}$ be a model in which all points are coplanar and equal to the points in $i_{(1,1)}$. Then clearly $m_{(1,1)}$ can also produce the image points $i_{(1,1)}$. This means that $X_{(1,1)}$ must also be part of $m_{(1,1)}$'s manifold.

Now there also exists a viewing direction, θ_{22}, for which the image of $\mathbf{p_4}$ will have affine coordinates $\alpha_4(\theta_{22}) = 2 = \beta_4(\theta_{22})$, as above. Let $i_{(2,2)}$ be the image that m produces when viewed from this orientation. $i_{(2,2)}$ will map to the point $X_{(2,2)}$ in the index space, and m's manifold must also include that point. Let $m_{(2,2)}$ be a planar model that is identical to $i_{(2,2)}$. Then $m_{(2,2)}$'s manifold must also include $X_{(2,2)}$. Since $m_{(2,2)}$ is a planar model, for any projection of $M_{(2,2)}$ its fourth point will have the affine coordinates $(2,2)$. But in $i_{(1,1)}$, the fourth image point has the affine coordinates $(1,1)$. By Lemma 2.1, there is no orientation for which $m_{(2,2)}$ can produce the image $i_{(1,1)}$. If $X_{(2,2)} = X_{(1,1)}$, then $i_{(1,1)}$ maps to model $m_{(2,2)}$'s manifold, even though model $m_{(2,2)}$ could not possibly have produced image $i_{(1,1)}$. So, by our assumption that our mapping to image space introduces no errors, $X_{(2,2)} \neq X_{(1,1)}$. That is, $m_{(1,1)}$ and $m_{(2,2)}$ must have disjoint manifolds, while m's manifold must include points in *each* manifold. Similarly, for any point in the plane, (i,j), we can create an image $i_{(i,j)}$ and a model $m_{(i,j)}$. $i_{(i,j)}$ will map to point $X_{(i,j)}$ in the image space, and $m_{(i,j)}$ and m will each include $X_{(i,j)}$ in their manifold. Also, similar reasoning will tell us that $X_{(i,j)} \neq X_{(i',j')}$, unless $i = i'$ and $j = j'$. So, there is a one-to-one mapping from points in the plane to distinct points in m's manifold in image space. We notice a few things about this proof. It applies equally well if m contains more than four points. Also, it does not depend on any particular representation of images. Although we use affine coordinates to describe an image, it is not assumed that this representation is used to define our image space. Finally, we have not so far assumed that our mapping from images to image space is continuous. If we make that additional assumption, we may conclude that each nonplanar model's manifold is at least two-dimensional in image space. This is due to a basic result in topology that any continuous, one-to-one mapping must preserve dimensionality (see, for example, [114]). Since a slight change in an image produces a slight change in its affine coordinates, we have a continuous one-to-one mapping from the plane to a portion of each model's manifold. So these manifolds must be at least 2-D.

In the course of the above proof, we have established the following lemma, which will be of use later:

Lemma 2.7 *The dimensionality of a model's manifold in any error-free image space is bounded below by the dimensionality of its manifold in affine space, when scaled orthographic projection is used.*

This lemma states that if we consider the set of all collections of affine coordinates that a model may produce in various images, the dimensionality of this set provides a lower bound on a model's manifold in any image space. In the above proof, we have shown that a model of rigid point features produces a 2-D manifold in the space of possible affine coordinates, and used that to show that these models must be represented by a 2-D manifold in any image space that does not introduce errors. Later we will use this lemma in the case where a non-rigid object can produce a greater than 2-D manifold of affine coordinates.

It is also true that:

Theorem 2.8 *For 3-D models and scaled orthographic projection there is an image space such that each model's manifold is no more than 2-D.*

This is accomplished by representing each image in any way that is invariant with scale, and rotation and translation in the plane. In that case, there is a direct correspondence between points on a unit viewing sphere and images of the model that map to different points in image space. So it is not hard to show our theorem using such a representation. A more careful proof is given in Clemens and Jacobs[32].

Perspective Projection

We may now show an analogous result for perspective projection. The reader should note that at no point in this section do we make use of the focal length of the projection. These results apply equally well whether or not the camera geometry, including focal length, is known or unknown. We show that any error-free mapping must take a model's images to at least a 3-D manifold in image space. This is of interest because it shows that the problem of indexing models under perspective projection is fundamentally more difficult than it is under scaled orthographic projection. This also suggests why it may be difficult to extend the results we will present in section 2.3.2 to the case of perspective projection.

Many of the same techniques may be used for this proof as were used in the case of scaled orthographic projection. With perspective projection, we may describe an image using the location of its first four points, and the projective coordinates of the remaining points with respect to these four points. In the case of orthographic projection we showed that, when viewed from all directions, a model could produce a 2-D set of affine coordinates. That is, the model could produce any values for (α_4, β_4). And we showed that two images with different affine coordinates would have to map to distinct points in image space. The same reasoning shows that in the case of perspective projection, any two images with different projective coordinates must map to different points in image space, and we need not repeat that argument here. It remains to characterize the set of projective coordinates that a 3-D model might produce when viewed from different directions.

We consider a model with at least six points, $(\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}, \mathbf{p_5}, \mathbf{p_6})$. We again refer to the plane formed by the first three points as the model plane. We will also ignore some degenerate cases. For example, we assume that the lines connecting $\mathbf{p_4}, \mathbf{p_5}$ and $\mathbf{p_6}$ are not parallel to the model plane, and that none of these three points lie in the model plane.

Since the model has six points, images of it will have four projective coordinates, $\gamma_5, \xi_5, \gamma_6, \xi_6$. In Appendix A we show that any model can produce any set of values for three of the projective invariants that describe the model's images. However, this derivation requires some simple tools from analytic projective geometry, and is not self-contained. So in this section we will use a geometric argument to show that for any model, and for any values of γ_5 and ξ_5, there will be an image of the model that has those projective coordinates. We will then show that for any pair of γ_5, ξ_5 coordinates there will also be a range of values that γ_6 can have. This is sufficient to show that a model's images map to a 3-D surface in the space of projective coordinates. And the geometric derivation may provide some useful insight into the problem.

We now define three special points on the model plane. For a fixed focal point, $\mathbf{f}$, the viewing lines connecting $\mathbf{f}$ to $\mathbf{p_4}, \mathbf{p_5}$ and $\mathbf{p_6}$ intersect the model plane at points that we will call $\mathbf{s_4}, \mathbf{s_5}$ and $\mathbf{s_6}$ respectively. As with scaled orthographic projection, our task now is to determine the set of projective coordinates that can be produced by the coplanar points: $\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{s_4}, \mathbf{s_5}, \mathbf{s_6}$. This is exactly the set of projective coordinates produced by images of the model. Our strategy will be to first provide a geometric description of the possible locations of $\mathbf{s_4}, \mathbf{s_5}$ and $\mathbf{s_6}$ in the model plane. We will use this to show that the projective coordinates (γ_5, ξ_5) can take on any values. We also show that the value of ξ_5 is independent of the values of γ_5 and γ_6. Then we consider the possible pairs of values that can occur for (γ_5, γ_6). We show that this set of values is a 2-D portion of the γ_5-γ_6 space. Since for any allowable pair of (γ_5, γ_6) values we can get any value for ξ_5, this tells us that the set of projective coordinates a model can produce forms at least a 3-D manifold in the space of possible projective coordinates (projective space).

First, we note that $\mathbf{s_4}$ can appear anywhere in the model plane. A line connecting any point on the model plane with $\mathbf{p_4}$ determines a set of possible locations for $\mathbf{f}$ which will place $\mathbf{s_4}$ at that point on the model plane. We now define two new special points which will help us to determine the compatible locations of $\mathbf{s_4}, \mathbf{s_5}$ and $\mathbf{s_6}$. $\mathbf{p_4}$ and $\mathbf{p_5}$ determine a line. We will call the point at which this line intersects the model plane $\mathbf{r_5}$. Similarly, let $\mathbf{r_6}$ stand for the point where the line connecting $\mathbf{p_4}$ and $\mathbf{p_6}$

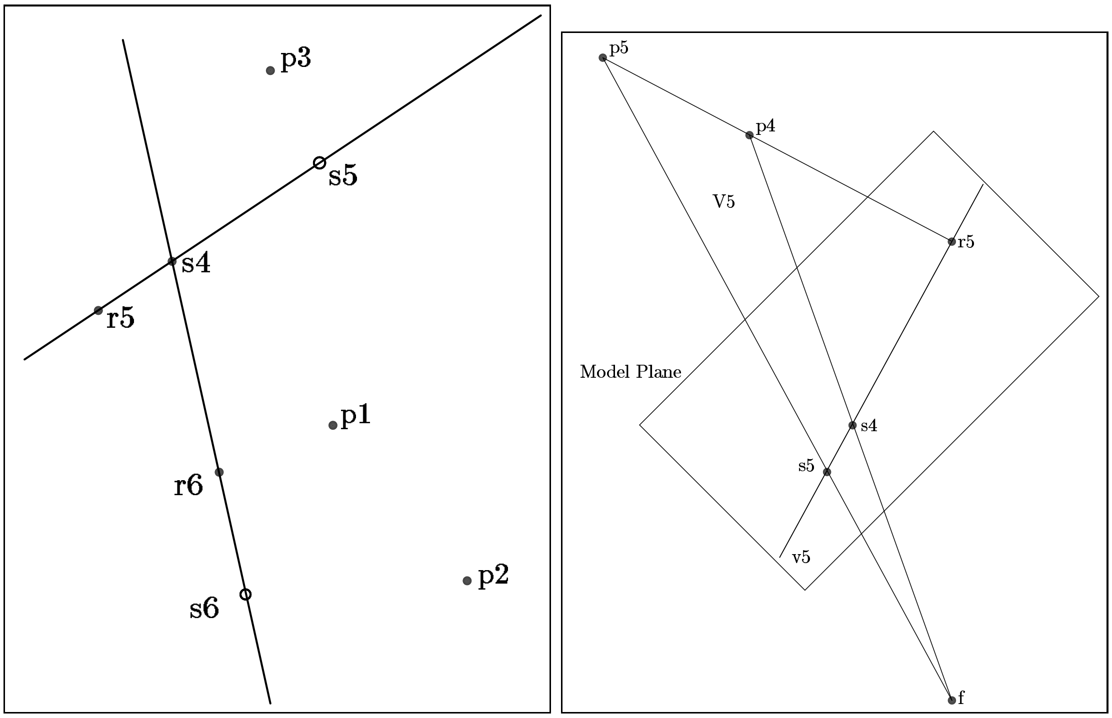


Figure 2.11: On the left, we show the points that lie on the model plane. We illustrate the constraints that $\mathbf{s_4}$, $\mathbf{s_i}$ and $\mathbf{r_i}$ must be collinear. On the right, we show the relationship between these points and the related model points, demonstrating why these collinearity constraints must hold.

intersects the model plane. Note that $\mathbf{r_5}$ and $\mathbf{r_6}$ are intrinsic characteristics of the model that do not depend on the viewpoint. The layout is illustrated in figure 2.11.

The three points $\mathbf{f}$, $\mathbf{p_4}$ and $\mathbf{p_5}$ form a plane, which we will call V_5. $\mathbf{s_4}$ and $\mathbf{s_5}$ are in V_5, since they are on the lines connecting $\mathbf{f}$ to $\mathbf{p_4}$ and $\mathbf{p_5}$. The intersection of V_5 and the model plane is then the line connecting $\mathbf{s_4}$ and $\mathbf{s_5}$. Call this line v_5. Since the line connecting $\mathbf{p_4}$ and $\mathbf{p_5}$ is in V_5, this means that $\mathbf{r_5}$ is also in V_5. Since $\mathbf{r_5}$ is defined to be in the model plane, this means that $\mathbf{r_5}$ is also on v_5. Therefore, once we specify the location of $\mathbf{s_4}$, we have also specified a simple geometric constraint on the location of $\mathbf{s_5}$; we know that it must lie on the line determined by $\mathbf{s_4}$ and $\mathbf{r_5}$.

Furthermore, $\mathbf{s_5}$ can fall anywhere on this line. To see this, we suppose that $\mathbf{s_4}$ and $\mathbf{s_5}$ are anywhere in the model plane, subject to the constraint that they be collinear with $\mathbf{r_5}$, and then determine a focal point such that the lines from the focal point to $\mathbf{p_4}$ and $\mathbf{p_5}$ includes $\mathbf{s_4}$ and $\mathbf{s_5}$. From our assumptions, there is a single line that contains $\mathbf{s_4}$, $\mathbf{s_5}$ and $\mathbf{r_5}$, and another line that connects $\mathbf{p_4}$, $\mathbf{p_5}$ and $\mathbf{r_5}$, and so together, these five points lie in a single plane. This plane will also include the line that connects $\mathbf{s_4}$ and $\mathbf{p_4}$, and the line that connects $\mathbf{s_5}$ and $\mathbf{p_5}$. Therefore, these two lines either intersect, or are parallel (intersect at infinity), they cannot be skewed. If they intersect, their point of intersection provides a focal point from which $\mathbf{p_4}$ and $\mathbf{s_4}$ project to the same place in the image, and similarly for $\mathbf{p_5}$ and $\mathbf{s_5}$. If the lines are parallel, then assuming a focal point at infinity, in the direction from $\mathbf{s_4}$ to $\mathbf{p_4}$, produces the appropriate projection. We have therefore shown that the images that the model produces can be described by saying that $\mathbf{s_4}$ may be anywhere in the model plane, and that $\mathbf{s_5}$ can be anywhere on the line connecting $\mathbf{r_5}$ and $\mathbf{s_4}$. Similarly, once we know the location of $\mathbf{s_4}$ we know that $\mathbf{s_6}$ can lie anywhere on the line connecting $\mathbf{s_4}$ and $\mathbf{r_6}$.

We now show that for any model, and for any values of (γ_5, ξ_5), there exists a viewpoint from which the model's image has those projective coordinates. To do this we do not need to explicitly discuss the viewpoint, but only the locations of $\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{s_4}$ and $\mathbf{s_5}$, since the invariant values that these points produce in the model plane will be preserved in the image. First, we note that the value of γ_5 is fully determined by the location of $\mathbf{s_4}$. To see this, recall that γ_5 is a cross ratio based on four points. Two of these points, $\mathbf{p_1}$ and $\mathbf{p_2}$ are independent of the viewpoint. A third point, $\mathbf{p_1'}$, depends on the intersection of the line formed by $\mathbf{p_1}$ and $\mathbf{p_2}$ with the line formed by $\mathbf{p_3}$ and $\mathbf{s_4}$, which is determined by the location of $\mathbf{s_4}$. And the fourth point is dependent on the line formed by $\mathbf{s_4}$ and $\mathbf{s_5}$. However, since this is also the line formed by $\mathbf{s_4}$ and $\mathbf{r_5}$, this line is also determined by the geometry of the model and the location of $\mathbf{s_4}$.

This has two implications. We note that if one of the points used to compute a cross-ratio varies along all the points of the line while the other three points remain fixed, then all possible values of the cross-ratio will occur. Now first, suppose we want to produce a particular value of γ_5. We may choose any values of $\mathbf{p_1'}$ and $\mathbf{p_2'}$ that produce this cross ratio. Then the intersection of the line from $\mathbf{p_2'}$ to $\mathbf{r_5}$ with the line connecting $\mathbf{p_1'}$ and $\mathbf{p_3}$ will provide us with a location of $\mathbf{s_4}$ that will produce this value of γ_5. Second, note that once we have fixed $\mathbf{s_4}$, three of the points used to compute the cross-ratio ξ_5 are fixed. The remaining point is the intersection of the line from $\mathbf{s_5}$ to $\mathbf{p_2}$ with the line connecting $\mathbf{p_1}$ and $\mathbf{p_3}$. As $\mathbf{s_5}$ varies along the line connecting $\mathbf{r_5}$ and $\mathbf{s_4}$, this intersection point can appear anywhere on the line connecting $\mathbf{p_1}$ and $\mathbf{p_2}$. Therefore, ξ_5 may take on any value. Together we see that we may choose $\mathbf{s_4}$ and $\mathbf{s_5}$ to produce any pair of values for (γ_5, ξ_5).

We conjecture that any model may also produce any values for (γ_5, γ_6), and therefore, any values for $(\gamma_5, \xi_5, \gamma_6)$, since the values of ξ_5 that can appear in an image are independent of the values for any γ_i (which all depend only on the location of $\mathbf{s_4}$). However, we have not found a simple way of proving this. Furthermore, our primary goal in this section is to show a weaker result, that a model's images form a 3-D manifold in projective space. We show that now. As we have noted, it remains only to show that a model's images fill up a 2-D portion of γ_5-γ_6 space.

Suppose that $\mathbf{s_4}$ varies along a line that includes $\mathbf{r_5}$ and $\mathbf{p_3}$, but not $\mathbf{r_6}$. Then all the points used to calculate γ_5 will remain fixed, and in fact γ_5 will always equal 1. However, as $\mathbf{s_4}$ varies along this line, so will the line connecting $\mathbf{s_4}$ and $\mathbf{r_6}$, and so γ_6 will assume all possible values. We may therefore easily choose two unequal values, c_1 and c_2, such that our model can produce an image with any pair of projective coordinates (γ_5, γ_6) such that $\gamma_5 = 1$ and $c_1 \leq \gamma_6 \leq c_2$. Our choice of c_1 and c_2 gives us a segment of the line connecting $\mathbf{r_5}$ and $\mathbf{p_3}$. As $\mathbf{s_4}$ varies along this line segment, values of γ_6 between c_1 and c_2 are produced.

Now we choose another line segment parallel to this one, and approaching it very closely. We consider the values of γ_5 and γ_6 that are produced as the second line segment approaches the first one. A range of values of γ_5 are produced, but this range collapses down to 1 as the line segments converge. The range of values for γ_6 converge to the range of values from c_1 to c_2. Therefore, for any small δc we can find a δg that is small enough that as the line segments get close and γ_5 is always within δg of 1, then γ_6 takes on all values between $c_1 + \delta c$ and $c_2 - \delta c$. That is, we have defined a small rectangle of possible locations for $\mathbf{s_4}$ such that, as $\mathbf{s_4}$ varies among locations in that rectangle, γ_5 and γ_6 take on all pairs of values within a small rectangle of possible values. This shows that a model can produce a 2-D manifold of values for the parameters (γ_5, γ_6), and so a 3-D manifold of values for the parameters $(\gamma_5, \xi_5, \gamma_6)$. This demonstrates that indexing with perspective projection will inevitably require us to represent a higher-dimensional surface than will indexing with scaled orthographic projection.

We also note that the projective coordinates that a model produces will be no more than a 3-D manifold. In general, knowing the values of γ_5 and γ_6 fixes the location of $\mathbf{s_4}$, and hence of other γ values, while knowing the value of ξ_5 will determine the location of the focal point.

2.3.2 Linear Projection

We now turn to the images produced by 3-D point models with a linear projection model. We will see that, as with orthographic projection, we still need 2-D manifolds to represent these images in a single image space. But these manifolds may be decomposed into two 1-D manifolds in two image subspaces. And all of these manifolds are linear, and easily determined by analytic means. This result, which was first presented in Jacobs[62], essentially reduces the indexing problem to one of matching points to 1-D lines. Short of a zero-dimensional representation of models' images, this is the best for which we could hope.

To show this result, we will use the affine invariant representation of images introduced in section 2.3.1. We describe each image with the parameters: $(\mathbf{o}, \mathbf{u}, \mathbf{v}, \alpha_4, \beta_4, ...\alpha_n, \beta_n)$ denoting an affine coordinate frame and then the image's affine coordinates.

The first three of these parameters provide no information about the model that produced an image, and so we may ignore them. To see this, suppose that we view our model, m, with a scaled orthographic transformation which we will call $\mathbf{V}$, followed by an affine transformation we will call $\mathbf{A}$, producing the image, i. That is:

$$\begin{aligned} \mathbf{AV}m &= i \\ &= (\mathbf{o}, \mathbf{u}, \mathbf{v}, \alpha_4, \beta_4, ...\alpha_n, \beta_n) \end{aligned}$$

Then, for any values of $(\mathbf{o}', \mathbf{u}', \mathbf{v}')$, we want to show that m may produce the image with parameters: $(\mathbf{o}', \mathbf{u}', \mathbf{v}', \alpha_4, \beta_4, ...\alpha_n, \beta_n)$. We know from lemma 2.2 that there is an affine transformation, $\mathbf{A}'$ that maps $(\mathbf{o}, \mathbf{u}, \mathbf{v})$ to $(\mathbf{o}', \mathbf{u}', \mathbf{v}')$ (except for the degenerate case of collinearity), and from lemma 2.1 that this will leave the image's affine coordinates unchanged. Therefore,

$$\begin{aligned} \mathbf{i}' &= (\mathbf{o}', \mathbf{u}', \mathbf{v}', \alpha_\mathbf{4}, \beta_\mathbf{4}, ...\alpha_\mathbf{n}, \beta_\mathbf{n}) \\ &= \mathbf{A}'\mathbf{AVm} \end{aligned}$$

Since affine transformations form a group, we can combine $\mathbf{A}$ and $\mathbf{A}'$ into a single affine transformation. This means that there is a linear projection of m that produces image i'. We may therefore ignore the parameters $(\mathbf{o}, \mathbf{u}, \mathbf{v})$ in describing the images that a model can produce, since we know that these may take on any values.

We may also now ignore the affine transformation portion of the projection, because this has no effect on the remaining, affine-invariant image parameters. We have therefore shown:

Lemma 2.9 *To describe the images that a model produces with linear projections, we need only consider the affine coordinates produced in images as the model is viewed from each point on the viewing sphere.*

The remaining image parameters form an image space that we will call *affine space*. An image with n ordered points is mapped into a point in a $2(n-3)$-dimensional affine space by finding the affine coordinates of the image points, using the first three as a basis. We divide the affine space into two orthogonal subspaces, an α*-space*, and a β*-space*. The α*-space* is the set of α coordinates describing the image, and the β-space is defined similarly. The affine space is then equal to the cross product of the α*-space* and the β*-space*, and each image corresponds to a point in each of these two spaces. We now show that the images of any model map to the cross product of lines in α*-space* and β*-space*.

We know from lemma 2.5 that a model can produce an image containing any values for (α_4, β_4). We now express the remaining affine coordinates of an image as a function of these values and of a model's properties. Figure 2.12 shows a view of the five points $(\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}, \mathbf{p_j})$. We will consider degenerate cases later, but for now we assume that no three points are collinear, and no four points are coplanar. First we define two new points in the model plane. Let $\mathbf{p'_4}$ be the point in the model plane that is the perpendicular projection of $\mathbf{p_4}$. That is, the line from $\mathbf{p'_4}$ to $\mathbf{p_4}$ is orthogonal to the model plane. Since $\mathbf{p'_4}$ is in the model plane, we can describe it with affine coordinates, using the first three model points as an affine coordinate system. We call the affine coordinates of $\mathbf{p'_4}$: (a_4, b_4). Similarly, we define $\mathbf{p'_j}$ to be the point in the model plane perpendicularly below $\mathbf{p_j}$, with affine coordinates (a_j, b_j). Let us assume that from the current viewpoint, the fourth image point, $\mathbf{q_4}$, has affine coordinates (α_4, β_4), and so does the point in the model plane $\mathbf{s_4}$. That is, the viewing direction is on a line through $\mathbf{s_4}, \mathbf{p_4}$, and $\mathbf{q_4}$. Then a parallel line connects $\mathbf{p_j}$ with its image point, $\mathbf{q_j}$. We will call the point where this line passes through the model plane: $\mathbf{s_j}$. We will call the affine coordinates of $\mathbf{s_j}$ in the model plane: (α_j, β_j). So from the viewpoint depicted in figure 2.12 the model's image has affine coordinates (α_4, β_4) and (α_j, β_j)

We now relate these coordinates. The triangle $\mathbf{p_4 p'_4 s_4}$ will be similar to the triangle $\mathbf{p_j p'_j s_j}$, because the lines $\mathbf{p_4 s_4}$ and $\mathbf{p_j s_j}$ are parallel viewing lines, and the lines $\mathbf{p_4 p'_4}$ and $\mathbf{p_j p'_j}$ are both orthogonal to the model plane. If we let r_j be the ratio of the height of $\mathbf{p_4}$ above the model plane to the height of $\mathbf{p_j}$ above the model plane, than r_j is the scale factor between the two triangles. So: $(\mathbf{s_4} - \mathbf{p'_4}) = r_j(\mathbf{s_j} - \mathbf{p'_j})$, and therefore:

$$(\alpha_j, \beta_j) = (a_j, b_j) + \frac{((\alpha_4, \beta_4) - (a_4, b_4))}{r_j}. \tag{2.1}$$

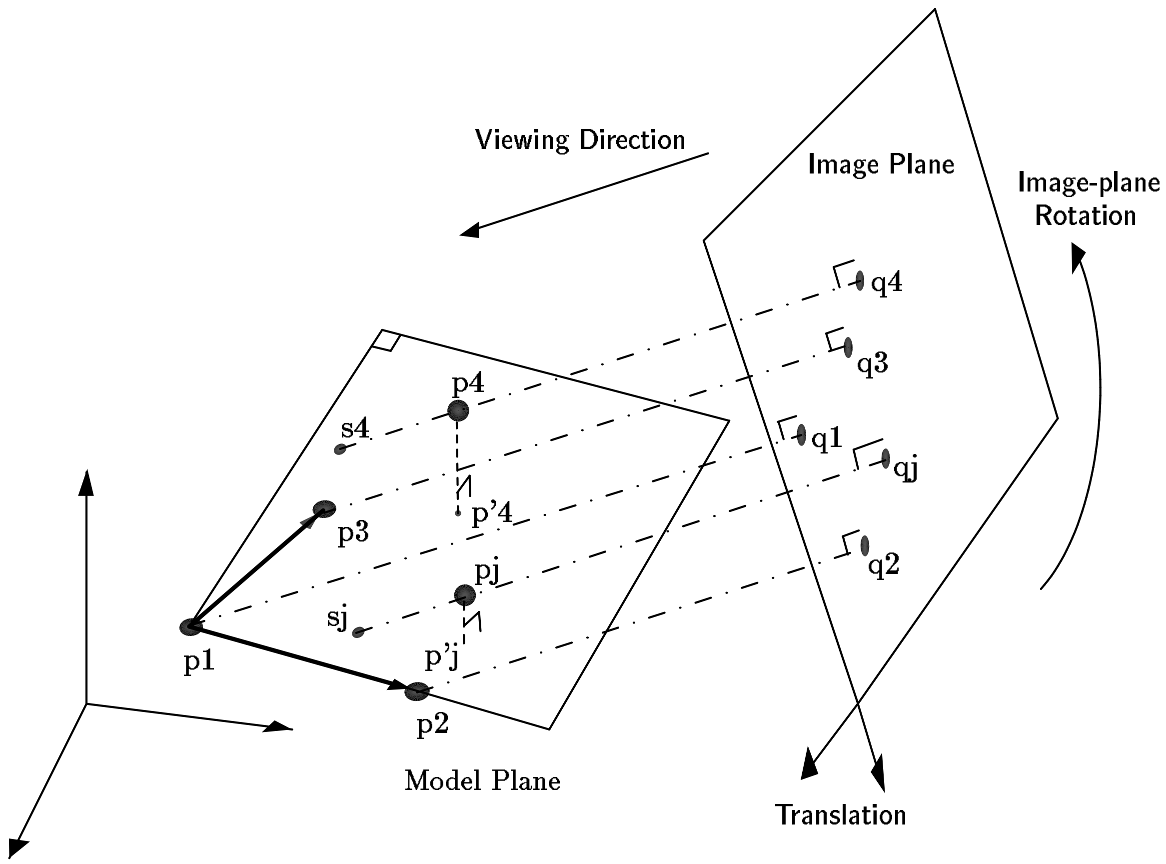


Figure 2.12: The image points $\mathbf{q_1}$, $\mathbf{q_2}$, $\mathbf{q_3}$, $\mathbf{q_4}$, and $\mathbf{q_j}$ are the projections of the model points $\mathbf{p_1}$, $\mathbf{p_2}$, $\mathbf{p_3}$, $\mathbf{p_4}$, and $\mathbf{p_j}$, before the affine transform portion of the projection is applied. The values of the image points depend on the pose of the model relative to the image plane. In the viewing direction shown, $\mathbf{s_4}$ and $\mathbf{p_4}$ project to the same image point. $\mathbf{p'_4}$ is in the model plane, directly below $\mathbf{p_4}$. Note that $\mathbf{q_4}$ has the same affine coordinates as $\mathbf{s_4}$.

This equation describes all image parameters that these five points may produce. For any image, this equation will hold. And for any values described by the equation, there is a corresponding image that the model may produce, since, from Lemma 2.5 we know that for any values (α_4, β_4), there is a view of $\mathbf{p_4}$ that produces these values. A model produces a series of these equations, which, taken together, describe a 2-D plane in affine space.

Taking the α component of these equations, we get the system of equations:

$$\begin{aligned}
\alpha_5 &= a_5 + \frac{(\alpha_4 - a_4)}{r_5} \\
&\cdot \\
&\cdot \\
&\cdot \\
\alpha_n &= a_n + \frac{(\alpha_4 - a_4)}{r_n}
\end{aligned}$$

Since the values: $a_4, ..., a_n$ and $r_4, ..., r_n$ are constant characteristics of the model, these are linear equations that describe a line in α-space. Similarly, we get:

$$\begin{aligned}
\beta_5 &= b_5 + \frac{(\beta_4 - b_4)}{r_5} \\
&\cdot \\
&\cdot \\
&\cdot \\
\beta_n &= b_n + \frac{(\beta_4 - b_4)}{r_n}
\end{aligned}$$

These equations are independent. That is, for any set of α coordinates that a model may produce in an image, it may still produce any feasible set of β coordinates.

Notice that for any line in α-space, there is some model whose images are described by that line. It is not true that there is a model corresponding to any pair of lines in α-space and β-space because the parameters r_j are the same in the equations for the two lines. This means that the two lines are constrained to have the same directional vector, but they are not further constrained.

There are also degenerate cases in which this derivation does not hold. If some of the model points are coplanar, than some of the r_j are infinite, and the lines are vertical in those dimensions. If all the model points are coplanar, the affine coordinates of the projected model points are invariant, and each model is represented by a point in affine space. If the first three model points are collinear, then the lines are undefined.

This is the lowest-dimensional complete representation possible of a model's images, assuming a linear projection transformation. The same proof used in the scaled orthographic case shows that a continuous 2-D manifold must be represented in the linear case. And it is not possible to decompose such a surface into the cross-product of zero-dimensional manifolds. So any complete representation must involve at least the cross-product of two 1-D manifolds.

2.3.3 Linear Projections of More Complex Models

We now consider two ways of making our models more complex: one in which point features have one or more directional vectors attached to them, a second in which objects have rotational degrees of freedom. These are important generalizations. A number of recognition systems use oriented point features, and many real objects have rotational degrees of freedom. It is also valuable for us to consider new kinds of models in order to get an idea of how well the results we have developed might extend to more challenging domains. We will see that it is possible to analytically characterize the images produced by models with oriented points. However we will also see that our most valuable results do not extend to this case. Models of oriented points map to 2-D hyperboloids in image space. We prove that these hyperboloids may not be decomposed into pairs of 1-D manifolds as we have done before. This places a quadratic bound on the space required to index oriented points using our straightforward, index table approach. Then we show that as rotational degrees of freedom are added to a model of point features, the dimensionality of a model's manifold grows. We show that this dimensionality cannot be reduced even by allowing false positive matches. These results tell us that the indexing problem becomes inherently much more difficult as we consider more complex, realistic objects.

3-D Oriented Point Models

The manifolds are hyperboloids

By an oriented point we mean a point with one or more directional vectors attached to it. For example, we might detect corners in an image for which we know not only the location of the corner, but also the directions of the two or three lines that intersect to form the corner. Alternately, we might distinguish special points on curves such as curvature discontinuities or extrema, and make a feature from the location of that point combined with the curve tangent. These two situations are illustrated in figure 2.13. In both cases we can consider our model as having 3-D vectors of unknown magnitude associated with each 3-D point. We then consider our image as containing the 3-D projections of these features.

Oriented points have been used in a variety of recognition systems. For example, Thompson and Mundy[100], Huttenlocher and Ullman[57], Bolles and Cain[13], and Tucker, Feynman and Fritzsche[101] use vertices as features. Other systems have used

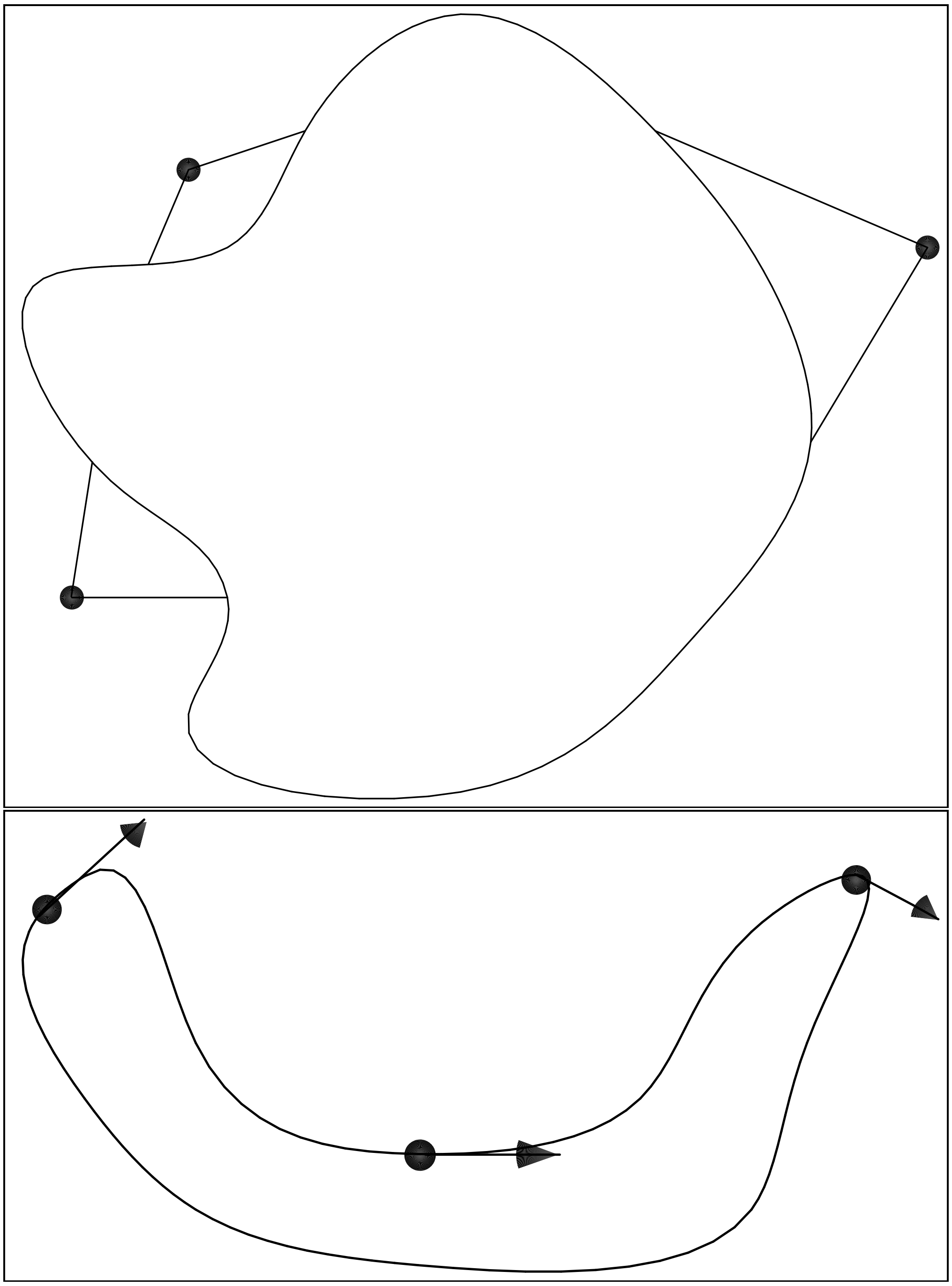


Figure 2.13: This figure shows simple examples of images with three oriented points each. Above, the points are vertices, shown as circles. For each point, we know two directional vectors from the lines that formed the vertex. Below, we assume that we can locate some distinguished points along the boundary of a curve, and can determine the tangent vector of the curve at those points.

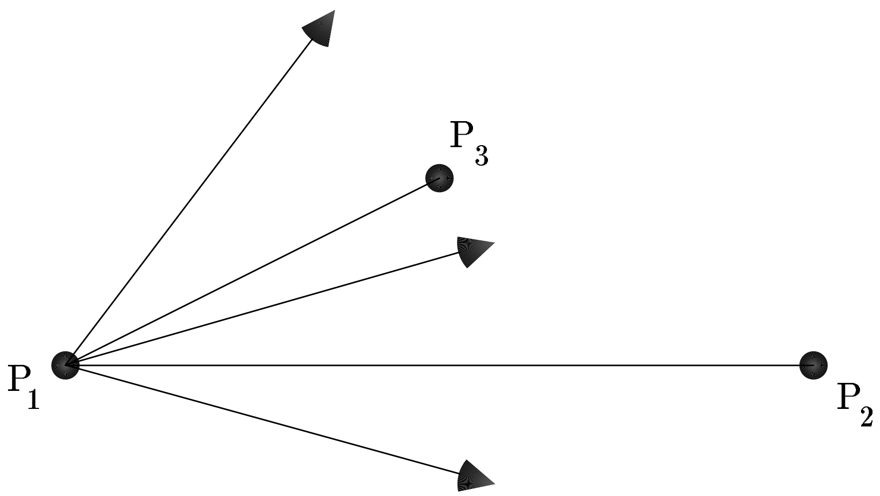


Figure 2.14: The three points shown are used as an affine basis, and the slopes of the vectors are found in this coordinate system.

points and tangents to characterize curves, such as Asada and Brady[1], Marimont[76], Mokhtarian and Mackworth[82], Cass[26],[27], and Breuel[20]. These features are valuable because they are local and powerful. It doesn't take too much of the image to reliably locate a point and its orientation, but this feature provides us with more information than a point feature alone.

We now derive the manifolds in image space that describe such models' images. To do this, we first extend our affine invariant representation to handle oriented points. Then we show that using this representation, each model corresponds to a 2-D manifold in image space that is a hyperboloid when we consider three dimensions of the image space.

To simplify our representation, we assume that each model contains at least three oriented points. We then use the images of these three points to define an affine basis as we did before, and describe the points' orientation vectors using this basis. Our image consists of points with associated vectors. The location of these vectors is irrelevant, so without loss of generality we may locate them all at the origin (see figure 2.14). We describe any additional image points using their affine coordinates, and we describe each orientation vector by its *affine slope*.

Definition 2.6 *To find the* **affine slope** *of a vector at the origin, we just take the affine coordinates* (α, β) *of any point in the direction of the vector, and compute* $\frac{\alpha}{\beta}$.

It is easily seen from the properties of affine representations that the affine slope of a vector is well defined and is invariant under affine transformations. This representation of vectors is equivalent to an affine invariant representation derived by Van

Gool et al.[106] using different methods. We use affine slope to define a new image space that combines point and vector information. We describe an image with the affine coordinates of any points beyond the first three, $(\alpha_4, \beta_4, ...\alpha_n, \beta_n)$, and with the affine slopes of all vectors, which we will call $(\theta_0, ..., \theta_m)$. Together these values produce an *affine slope space*. As before, the problem of determining a model's manifold becomes one of determining the set of affine-invariant values it may produce when viewed from all points on the viewing sphere.

We solve this problem using the following strategy. First we introduce some special point features, called r_i, that are related to the vectors that we really need to consider. We then determine the manifolds produced by the combination of these new points and the points in the model. Since we have only point features to deal with, this manifold can be represented as a simple pair of lines in some α and β spaces. We then use these manifolds to determine the manifolds in affine slope space that represent our actual model.

We begin by introducing some special 3-D points. With every vector, $\mathbf{v_i}$, we associate some point, $\mathbf{r_i}$ that is in the direction $\mathbf{v_i}$ from the origin. We denote the points of the model by $\mathbf{p_i}$. We will describe images of the points $(\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}, ...\mathbf{p_n}, \mathbf{r_0}, ...\mathbf{r_m})$ by the affine coordinates: $(\alpha_4^p, ...\alpha_n^p, \alpha_0^r, ...\alpha_m^r)$ and $(\beta_4^p, ...\beta_n^p, \beta_4^r, ...\beta_m^r)$. Now we may use our previous results to describe these images using lines in these α and β spaces. We will call these lines A and B respectively.

We can describe A with a parameterized equation of the form:

$$A = \mathbf{a} + k\mathbf{w}$$

$\mathbf{a}$ is any point in α space on the line A. We denote the coordinates of $\mathbf{a}$ as: $(a_4^p, a_5^p, ...a_n^p, a_0^r, a_1^r, ...a_m^r)$. $\mathbf{w}$ is a vector in α space that expresses the direction of A, and we denote its coordinates as: $(w_4^p, ...w_n^p, w_0^r, ...w_m^r)$. k is a variable. As k varies, we get the points on the line A. Similarly, we let:

$$B = \mathbf{b} + c\mathbf{w}$$

Note that $\mathbf{w}$ is the same in both equations, because A and B must have the same directional vector.

However, in practice we cannot know the images of the points $\mathbf{r_i}$. The vectors that we detect in the image will provide us only with the direction to the $\mathbf{r_i}$ points' images, not their actual location. So A and B are not directly useful, we must use them to determine the affine slopes that can occur in an image. To do this, we note that if the $\mathbf{p_i}$ and $\mathbf{r_i}$ points together can produce an image with affine coordinates: $(\alpha_4^p, \beta_4^p, ...\alpha_n^p, \beta_n^p, \alpha_0^r, \beta_0^r, ...\alpha_m^r, \beta_m^r)$, then the $\mathbf{p_i}$ points and the $\mathbf{v_i}$ vectors can produce an image that is described in affine slope space as: $(\alpha_4^p, \beta_4^p, ...\alpha_n^p, \beta_n^p, \frac{\alpha_0^r}{\beta_0^r}, ...\frac{\alpha_m^r}{\beta_m^r})$.

We will now derive a set of equations that describe a model's manifold in the affine slope space $(\alpha_4, \beta_4, ...\alpha_n, \beta_n, \theta_0, ..., \theta_m)$. These equations will express the set of affine coordinates and slopes that a model can produce as a function of θ_0, θ_1, and the characteristics of the model. We start with the equations:

$$\begin{aligned}
\theta_i &= \frac{\alpha_i^r}{\beta_i^r} = \frac{a_i^r + kw_i^r}{b_i^r + cw_i^r} \\
\alpha_j &= \alpha_j^p = a_j^p + kw_j^p \\
\beta_j &= \beta_j^p = b_j^p + cw_j^p
\end{aligned}$$

for any choice of k and c, and over all values of i and j. That is, we pick any set of affine coordinates that can be produced by the model points and the constructed $\mathbf{r_i}$ points, and use these to determine affine coordinates and slopes that we could actually find in the image.

We ignore the degenerate cases where α_0^r and β_0^r are constant over the lengths of the lines A and B. These are the cases in which the first model vector is coplanar with the first three model points. Then we may choose for $\mathbf{a}$ and $\mathbf{b}$ those points on A and B for which $a_0^r = 0$ and $b_0^r = 0$, This gives us the equation:

$$\theta_0 = \frac{a_0^r + kw_0^r}{b_0^r + cw_0^r} = \frac{k}{c}$$

This implies

$$c = \frac{k}{\theta_0}$$

We can use this to get:

$$\begin{aligned}
\theta_1 &= \frac{a_1^r + kw_1^r}{b_1^r + \frac{kw_1^r}{\theta_0}} \\
\theta_1(b_1^r\theta_0 + kw_1^r) &= a_1^r\theta_0 + kw_1^r\theta_0 \\
k(w_1^r\theta_1 - w_1^r\theta_0) &= a_1^r\theta_0 - b_1^r\theta_0\theta_1 \\
k &= \frac{\theta_0(a_1^r - \theta_1 b_1^r)}{w_1^r(\theta_1 - \theta_0)}
\end{aligned}$$

So we can express k and c in terms of the first two affine slopes we detect in the image and properties of the model. This allows us to express each remaining image parameter, both affine slopes and the α and β coordinates that describe other image points, as a function of these first two affine slopes and properties of the model. We find:

$$\begin{aligned}
\alpha_j^p &= a_j^p + w_j^p k \\
&= a_j^p + \frac{w_j^p\theta_0(a_1^r - \theta_1 b_1^r)}{w_1^r(\theta_1 - \theta_0)}
\end{aligned}$$

$$
\begin{aligned}
\beta_j^p &= b_j^p + w_j^p c \\
&= b_j^p + \frac{w_j^p(a_1^r - \theta_1 b_1^r)}{w_r^p(\theta_1 - \theta_0)} \\
\theta_i &= \frac{a_i^r + w_i^r k}{b_i^r + w_i^r c} \\
&= \frac{a_i^r + \frac{w_i^r \theta_0 (a_1^r - \theta_1 b_1^r)}{w_1^r(\theta_1 - \theta_0)}}{b_i^r + \frac{w_i^r (a_1^r - \theta_1 b_1^r)}{w_1^r(\theta_1 - \theta_0)}} \\
&= \frac{a_i^r w_1^r(\theta_1 - \theta_0) + w_i^r \theta_0 (a_1 - \theta_1 b_1^r)}{b_i^r w_1^r(\theta_1 - \theta_0) + w_i^r(a_1^r - \theta_1 b_1^r)} \\
&= \frac{-b_1^r w_i^r \theta_0 \theta_1 + (a_1^r w_i^r - a_i^r w_1^r)\theta_0 + a_i^r w_1^r \theta_1}{-b_i^r w_1^r \theta_0 + (b_i^r w_1^r - b_1^r w_i^r)\theta_1 + a_1^r w_i^r}
\end{aligned}
$$

So we have an analytic form describing all images of the model. We now show in the case of 3 points and 3 vectors that this form describes a 2-D hyperboloid in a 3-D image space.

We introduce the following abbreviations:

$$
\begin{aligned}
c_1 &= a_1^r w_2^r \\
c_2 &= a_2^r w_1^r \\
c_3 &= b_1^r w_2^r \\
c_4 &= b_2^r w_1^r \\
x &= \theta_0 \\
y &= \theta_1 \\
z &= \theta_2
\end{aligned}
$$

We note that c_1, c_2, c_3, c_4 are properties of the model, and that models may be chosen to produce any set of these values. So, the set of manifolds that can be produced is precisely described by:

$$
z = \frac{-c_3 xy + (c_1 - c_2)x + c_2 y}{-c_4 x + (c_4 - c_3)y + c_1}
$$

$$
-c_4 xz + (c_4 - c_3)yz + c_1 z + c_3 xy - (c_1 - c_2)x - c_2 y = 0 \tag{2.2}
$$

Adopting the notation of Korn and Korn (pp. 74-76)[68] we find:

$$
\begin{aligned}
I &= 0 \\
D &= -2c_3 c_4 (c_4 - c_3) \\
A &= (c_1 c_4 - c_2 c_3)^2 \geq 0
\end{aligned}
$$

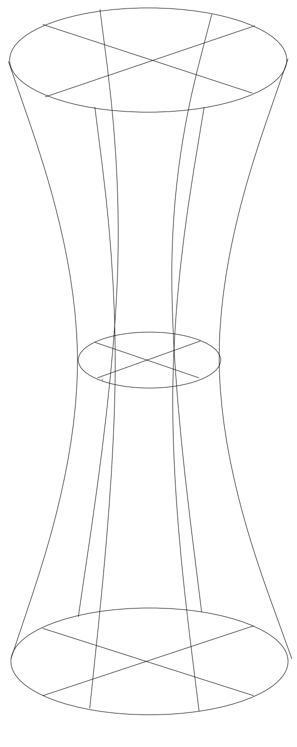


Figure 2.15: A solution to equation 2.2 is an hyperboloid of one sheet, shown in this figure.

This tells us that when we look at three dimensions of affine slope space, we find that a model's manifold is a hyperboloid of one sheet (see figure 2.15). We also see that we can find a model corresponding to any hyperboloid that fits equation 2.2. For our purposes, we do not need to consider the degenerate cases in detail.

We have shown that the images of a model of oriented points can be described by a 2-D manifold. We can see that at least a 2-D manifold is needed if we use a single image space, with the same argument we used in section 2.3.1. To summarize this argument, we can show that for any non-degenerate model there exists a viewpoint which will produce any values for θ_0 and θ_1. At the same time, planar models produce constant values of θ_0 and θ_1. Thus, every image of the model with different values of θ_0 and θ_1 must map to a different point in image space.

The manifolds cannot be decomposed

We now show that there is no way of dividing image space to decompose these manifolds into two 1-D surfaces in two image subspaces. Our proof will assume that each model contains at least three points, and three or more vectors. We assume that any configuration of points and vectors is a possible model. We also assume a continuous mapping from images to our image space, and to any image subspaces. We show that if such a decomposition of image space exists, that this restricts the kinds of intersections that can occur between manifolds, and that the class of manifolds produced by oriented point models do not meet these restrictions. By considering just the intersections in image space that occur between manifolds of different models, we get a result that will apply to any choice of image space, since the intersections of manifolds reflect shared images that will map to the same place in any image space.

We will suppose the opposite of our proposition, that there exist two images subspaces such that any model maps to a 1-D curve in each space. Then when two manifolds intersect in image space, we can determine the places where they intersect by taking the cross product of the intersections of their 1-D manifolds in the two image subspaces. Suppose that two models' manifolds intersect in image space in a 1-D curve. Then our decomposition of image space must represent this curve as the cross product of a 1-D curve in one image space, and a point in the second image space. This means that in one of the two image subspaces, the two curves that represent the two models must overlap, so that their intersection is also a curve and not just a point.

This observation allows us to formulate a plan for deriving a contradiction. We pick a model, M, with manifold H. We then choose a point P on H (that is, P corresponds to an image of M). We define p and p' as the points that correspond to P in the first and second image subspaces respectively. We will construct five new, special models, M_1, M_2, M_3, M_4, M_5. Each of these model's manifolds will intersect H

in a 1-D curve in image space. We call these curves K_1, K_2, K_3, K_4, K_5. Each of these curves will contain P, by construction. Then, since each curve maps to a curve in one image subspace, and a point in the other, we may assume without loss of generality that K_1, K_2, and K_3 map to the curves k_1, k_2, and k_3 in the first image subspace, and to the points r_1, r_2 and r_3 in the second image subspace. Then, in order for the curves K_1, K_2, and K_3 to all include the point P, it must be that $r_1 = r_2 = r_3 = p'$, and that k_1, k_2, and k_3 all intersect at the point p in the first image subspace. We will call the curve that represents M in the first image subspace k. k_1, k_2, and k_3 must all lie on k because they come from the intersection of M and other models. It is possible that two of these curves intersect only at p if they end at p, and they occupy portions of k on opposite sides of p. But with three curves, two at least (suppose they are k_1 and k_2) must intersect over some 1-D portion of k [2] . Since they both intersect at p' in the other image space, this will tell us that K_1 and K_2 intersect over some 1-D portion of image space. We will then derive a contradiction by showing that in fact all of the curves, K_1, K_2, K_3, K_4, K_5, intersect each other only at a single point, P. So, to summarize the steps needed to complete this proof, we will: construct the point P and the models $M, M_1, M_2, M_3, M_4, M_5$ so that each additional model's manifold intersects M's in a 1-D curve that includes P. We will then show that these curves intersect each other only at P, that is, that M and any two of the other models have only one common image.

For these constructions, we will choose our models to be identical and planar, except for their first three orientation vectors. Therefore, in considering the intersection of these models' manifolds, we need only consider their intersection in the coordinates $(\theta_0, \theta_1, \theta_2)$, since their remaining coordinates will always be constant, and will be the same for each model. Therefore, when we speak of a the coordinates of a point in image space, we will only consider these three coordinates. And to describe the values for $(\theta_0, \theta_1, \theta_2)$ that a model can produce, we need only give the values for c_1, c_2, c_3, c_4 that will describe the model's hyperboloid in $(\theta_0, \theta_1, \theta_2)$ space.

It is easy to see from equation 2.2 that, in general, any two of these hyperboloids will intersect in a set of 1-D surfaces, and any three hyperboloids will intersect only at points, and in the line $\theta_0 = \theta_1 = \theta_2$, as noted above. Therefore, any general set of six hyperboloids chosen to intersect at a common point will fulfill our needed construction.

[2] Actually, we are glossing over a somewhat subtle point. To prove that k_1 and k_2 must intersect over a 1-D portion of k requires the use of a theorem from topology which states that connectivity is a topological property. Briefly, a 1-D curve is connected if there are no "holes" in it. For example, the curve $y = 0$ is connected, but if we remove the point $(x, y) = (0, 0)$, it is not. Since the set of images that a model produces is connected in affine space, we can show that the curves k_1 and k_2 must be connected, and from that we may show that they must both contain the same 1-D region on one side of p.

We can also prove this result another way, which will perhaps strengthen the reader's intuitions about these hyperboloids. Let H be a 3-D hyperboloid, and P be an arbitrary point on it. We derive a contradiction after assuming that we can decompose H into two 1-D curves in two image subspaces. Suppose that P is represented again by the two points p and p' in the two image subspaces. Choose any other point Q on H. Referring to equation 2.2 we see that knowing two points of a hyperboloid gives us two linear equations in the four unknowns that describe the hyperboloid. Therefore, we may readily find a second hyperboloid, H', that also includes P and Q, but that does not coincide with H. As noted above, in general H and H' intersect in a 1-D curve, which must correspond to a curve in one image space, and to either p or p' in the other. In particular, this means that Q must correspond to either p or p'. Since Q is an arbitrary point, all points on H must correspond to either p or p'. This contradicts our assumption that H is represented by the cross-product of two curves.

Articulated Models

We now consider the manifolds produced by articulated objects. In particular, this section will consider objects composed of point features with rotational degrees of freedom. We assume that an object consists of some parts. Each part has a set of point features that are rigid relative to each other. However, there may be a fixed 3-D axis, about which a part may rotate. We assume this axis is defined by a 3-D line, implying a single degree of freedom in the rotation. Since we cannot know from a single image whether an object is articulated or rigid, we assume that any representational scheme we consider must handle both rigid and non-rigid objects. We consider just rotations for simplicity, however it will be clear that our main results will extend to a variety of other object articulations.

Many real objects have rotational degrees of freedom. For example, when we staple with a stapler, or when we open it to add staples, we are rotating a part of the stapler about an axis. Similarly, a pair of scissors or a swivel chair have rotational degrees of freedom, and much of the articulation in an animal's limbs can be modeled as rotational degrees of freedom. So this is a practical case to consider.

It is also important to push our approach into a more challenging domain such as this in order to see how far it can be taken. We find that as the number of rotational degrees of freedom of our model increases, so does the dimensionality of our objects' manifolds. This tells us that figuring out how to index such complex objects is not simply a matter of determining an object's manifold. Because of their high dimensionality, significant challenges remain to uncover methods of efficiently representing the relevant information that these manifolds convey.

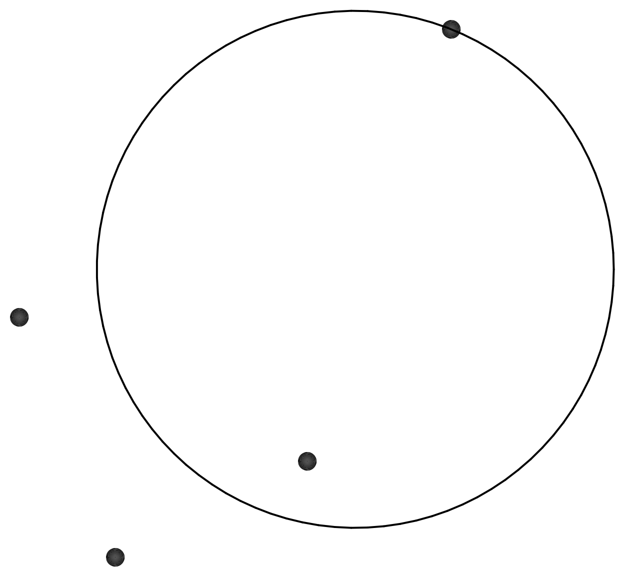


Figure 2.16: The simplest example of an interesting planar articulated object. The fourth point rotates in a circle in the plane, as the other three points are fixed.

Planar models

We begin with the case of planar models with rotational degrees of freedom, such as a pair of scissors. We assume in this case that even as the object rotates, all of its points remain coplanar. In the simplest case, we have a base part of three points, which we can consider rigid, and a rotating part that consists of just one point. This rotating point is coplanar with the base points, but may rotate in a circle about any point in the plane. This simple case is illustrated in figure 2.16.

We find the affine coordinates these four points may produce in an image by rewriting the equation for a euclidean circle in the affine coordinate frame defined by the base points. This is equivalent to transforming the model so that it's first three points map to the points $(0,0),(1,0),(01)$ in the plane, and finding the equation for the transformed circle. From elementary affine geometry we know that a applying an affine transformation to a circle will produce an ellipse. So every object with rotations maps to an ellipse in affine space while rigid objects continue to map to points.

If we allow more points in the base of the object, our model's images have more parameters in affine space. These additional parameters are constant. If we add more points to the rotating part, these points' affine coordinates each trace an ellipse in affine space. Together, they map to a 1-D curve in affine space that is elliptical in each component.

The same reasoning used above allows us to see that this is a bound on the dimensionality of the manifolds. Again, we may not compress together two points in affine space without confusing two rigid objects, and so we may not compress these 1-D curves in affine space into points. This shows that there are no complete invariants for rotating planar objects. In fact, we may show that the only invariants available for such objects are the invariants that might be computed separately from the objects' parts. If we do not know which points of the object belong to which parts, we may not compute any invariants at all.

If we increase the number of rotating parts, the dimensionality of the manifolds increases also. For example, when an object has two rotating parts with one point each, its images are described by an ellipse in α_4-β_4 space for the first point, and by an ellipse in α_5-β_5 space for the second point. Together these give us a 2-D manifold in a 4-D affine space. As we allow additional rotational degrees of freedom in our model, the dimensionality of the model's manifold increase by one in a similar way. In all these cases, we may decompose these manifolds into a series of 1-D manifolds, one for each rotating part, as long as we can tell from the image which points come from which parts.

Nonplanar Models

We now consider 3-D models with parts that have rotational degrees of freedom. We begin by proving that a general 3-D object with two parts must be represented by a 3-D manifold in image space. We then show that we can generalize this result to show that an object with n parts must be represented by an n-D manifold.

We first suppose that an object has two parts, P_1 and P_2. Assume that P_1 contains at least three points which we use to define the model plane, and whose projection we will use as our affine basis. Assume also that P_2 contains at least two points, $\mathbf{p_1}$ and $\mathbf{p_2}$, and is allowed to rotate about some axis line that we will call L. We refer to any particular rotation of P_2 as a *configuration* of the model. We first show that for almost any object, there are two configurations of the object whose manifolds intersect at most at a single point in affine space.

For any configuration of the model, its images will correspond to a plane in affine space and to lines in α and β space. The slope of these lines will be the height of $\mathbf{p_1}$ above the model plane divided by the height of $\mathbf{p_2}$ above the model plane. If the lines corresponding to two different configurations have different slopes in these affine coordinates, then the lines of the configurations can intersect in at most a single point. Suppose that $\mathbf{p_1}$, $\mathbf{p_2}$ and L are not all coplanar. Then as $\mathbf{p_1}$ and $\mathbf{p_2}$ rotate about L, there will be a point at which $\mathbf{p_1}$ is receding from the model plane, while $\mathbf{p_2}$ is approaching the model plane. This means that the ratio of their heights must be changing, and so the slope of the lines corresponding to these configurations will be changing. So we can readily find two model configurations whose manifolds correspond to lines with different slopes and intersect in at most a point in affine space. Suppose now that $\mathbf{p_1}$, $\mathbf{p_2}$ and L are planar. Let r_1 be the distance from $\mathbf{p_1}$ to L, and let r_2 be the distance from $\mathbf{p_2}$ to L. As P_1 rotates about L, the two points are displaced in the same direction with a magnitude that depends on their distance from L. So the change in $\mathbf{p_1}$'s height above the model plane will be $r_1 k$ and the change of $\mathbf{p_2}$'s height will be $r_2 k$, for some k. This means that the ratio of the two heights will be held constant only when that ratio is $\frac{r_1}{r_2}$. Again, if this ratio changes then the model will have two configurations that produce at most one common set of affine coordinates. So in general, the only case in which a rotating model does not correspond to two separate planes in affine space is when the model's second part is completely planar with its axis of rotation, and the ratios of the heights of all the points in this part are equal to the ratios of their distance from the axis of rotation.

We now show that except for this special case, a rotating model will correspond to at least a 3-D manifold in affine space. We know the model's manifold will include two planes that intersect at only a point. We also know that intermediate configurations of the model will correspond to a continuous series of planes. This continuous series of planes will form a 3-D manifold. We now have from lemma 2.7:

Theorem 2.10 *Except in a special degenerate case, a model with a rotating part must correspond to a 3-D manifold in any error-free image space.*

We note that this proof may be extended in a straightforward way to handle additional parts. For example, if a model has three parts, we know that holding one part rigid, the model produces a 3-D manifold in image space. We can then show that rotating that part produces a second 3-D manifold that intersects this manifold at most in a 1-D manifold. Therefore, we see that as both parts rotate they produce a 4-D manifold. In general, a model with n rotating parts corresponds to an n-D manifold in any image space.

Open Questions

There remain unanswered a number of interesting questions related to the ones we have addressed in this chapter. While we have shown that the most space-efficient way of representing a 3-D model's images produced by a linear transformation is with 1-D manifolds, we do not know whether 1-D manifolds can describe a model's images under the non-linear projection models, scaled orthographic projection and perspective projection. Such a decomposition of the non-linear manifolds would be quite useful, because it would allow us to match a rigid object to an image using an indexing system that requires essentially the same amount of space as the system described in this thesis, while also distinguishing between an image of an object and an

Table of Results				
Model	Projection	Space lower bound	Lower bound when space divided	Analytic description of manifold
3-D Points	Orthographic	2-D		
	Perspective	3-D		
	Linear	2-D	1-D	Linear
Oriented 3-D Points	Orthographic	2-D	2-D	
	Linear	2-D	2-D	Hyperboloid
Points, with N rotating parts.	Orthographic	(N+2)-D		
	Linear	(N+2)-D		

Table 2.1: A summary of this chapter's results.

image of a photograph of the object. We also have not shown whether the manifolds of non-rigid objects can be decomposed.

We have also presented a very simple representation of a model's images as a 2-D plane, or a pair of lines, when our projection model is linear. Is it possible to represent a model's images with a 2- or 3-D linear surface when the projection model is scaled orthographic or perspective projection? If it is, reasoning about matching problems under these projection models might be greatly simplified. Weinshall[110] has shown interesting related results which from our point of view express a model's manifold as a linear subspace when scaled orthographic projection is used. This subspace is of a higher dimension than that which is needed to describe a model's manifold non-linearly, however.

Finally, there is much work to be done in understanding the manifolds that represent the 2-D images of 3-D curves. We have characterized these manifolds when the location of a point on a curve and the first derivative of the curve at that point are known, but not when additional derivatives are known, nor when a polynomial representation of an entire curve is known. We expect that these manifolds will be 2-D when a linear transformation is used, but we do not have an analytic description of them, and we do not know whether they might be decomposed.

2.4 Conclusions

In this chapter, we have explored the problem of finding the simplest representation possible of a model's images under a variety of circumstances. In doing so we have produced two kinds of results, which we summarize in table 2.1. On the one hand, we have derived simple, analytic descriptions of the set of images that models of point, or oriented point features may produce under a linear transformation. On the other hand, we have shown lower bounds on the manifolds that these and other models produce under a variety of transformations. These two kinds of results serve two different ends.

The first set of results are the most directly useful, and we explore their consequences throughout much of the rest of this thesis. We have shown that we can represent a 3-D point model's images as a pair of 1-D lines in two high-dimensional spaces. This is especially useful for indexing, and a significant improvement on past results. In chapter 4 we show how we can use it to build a practical indexing system. We also show that even with this representation of a model's images, space is at a premium, suggesting that indexing using a 2-D manifold to represent a model's images is not very practical.

We also show that a 3-D model consisting of oriented points may be represented by a 2-D hyperboloid in a high-dimensional space. Both of these representations can help us to understand other approaches to recognition and matching. They provide us with a simple, geometric formulation of the matching problem as one of comparing points in a high-dimensional space to manifolds for which we have an analytic description. In chapters 3 and 5 we demonstrate some of the power of this formulation of the problem by analyzing several different approaches to recognition and motion understanding.

We also demonstrate a variety of lower bounds on the space required to represent a model's images. These bounds make two points. First, they show that the analytic results that we have derived are optimal. We need at least two 1-D manifolds to represent the images of a 3-D model of point features, and at least a 2-D manifold to represent a model of oriented point features. Also, although our results are derived for linear projections, we show that using scaled orthographic projection instead would not reduce these space requirements. Our picture is not quite complete, however. We know we cannot get better representations using scaled orthographic projection, but perhaps we could derive representations that are just as good. This would be useful since scaled orthographic projection is a more accurate model of the process of forming a single image from a 3-D model. Except for this issue, though, we know that for two interesting kinds of models we have derived the best possible representations of their images.

Our lower bounds are also interesting because they tell us that some indexing tasks are fundamentally more difficult than others. Images of oriented points must be represented with a 2-D manifold that cannot be decomposed, while the manifold produced by simple points can be constructed from manifolds that are only 1-D. We also see that indexing with perspective projection necessarily requires us to represent a model by representing a 3-D manifold, while only a 2-D manifold is required for scaled orthographic or linear projections. We do not know whether this 3-D manifold can be decomposed into smaller sub-manifolds. And we see that representing the images of objects with rotational degrees of freedom inherently requires more and more space as the number of rotational degrees of freedom grows. Even as we have produced useful solutions to some indexing problems, we have shown that others may become very difficult. We have a concrete method of characterizing the difficulty of an indexing problem, and we have shown how hard some problems may be.

This chapter is also of interest because it demonstrates how one may generalize the notion of an invariant. Invariant representations have attracted a great deal of attention from mathematicians, psychologists, photogrammetrists and computer vision researchers. For the situations of greatest interest, projection from 3-D to 2-D, there are no invariant descriptions, however. We suggest that an invariant may be thought of as a 0-D representation of the set that results from transforming an object. When invariants do not exist, it seems natural to generalize them by pursuing the lowest dimensional representation that is possible. We have shown that interesting tight bounds may be found when invariants do not exist.

Chapter 3

Implications for Other Representations

In this chapter we consider some results of others that are related to those presented in chapter 2. This serves several purposes. We acknowledge the relationship of some past work to our present results. Also, since our results are more powerful, we may rederive some of these past results more simply, or at least in a different way. At the same time, since this past work had applied only to point features, we may now see what happens when we try to extend these results to oriented points.

3.1 Linear Combinations

3.1.1 Point Features

Ullman and Basri[105] show that any image of a model of 3-D points can be expressed as a linear combination of a small set of basis images of the object. That is, given a few views of an object, $i_1 ... i_n$, and any new view, i_j, we can find coefficients $a_1 .. a_n$ so that:

$$i_j = \sum_{k=1}^{n} a_k i_k$$

where we multiply and sum images by just multiplying and summing the cartesian coordinates of each point separately. This idea is refined independently by Basri and by Poggio[89] into the following form.

Suppose we have a model, m, with n 3-D points. i_1 and i_2 are two images of m. We describe each image with cartesian coordinates, and assume there is no translation in the projection. Let $\mathbf{x_1}$ be an n-dimensional vector containing all of i_1's x coordinates, and let $\mathbf{y_1}$ be an n-dimensional vector of its y coordinates. Similarly, define $\mathbf{x_2}$ and

$\mathbf{y_2}$ for i_2. Take any new image of m, i_j, and define $\mathbf{x_j}$ and $\mathbf{y_j}$. Then Basri and Poggio show that there exist a_0, a_1, a_2 and b_0, b_1, b_2 such that:

$$\begin{aligned}
\mathbf{x_j} &= a_0\mathbf{x_1} + a_1\mathbf{y_1} + a_2\mathbf{x_2} \\
\mathbf{y_j} &= b_0\mathbf{x_1} + b_1\mathbf{y_1} + b_2\mathbf{x_2}
\end{aligned}$$

This tells us that the x and y coordinates of a new image are a linear combination of one and a half views of the object, that is, of the x and y coordinates of one view, and either the x or the y coordinates of a second view. Another way to think of this result is that $\mathbf{x_1}, \mathbf{y_1}, \mathbf{x_2}$ span a 3-D linear subspace in $\mathcal{R}^n$ that includes all sets of x or y coordinates that the object could later produce. We omit a proof of this, but note that the proof is based on a linear transformation. That is, when an arbitrary $3x2$ transformation matrix is used instead of a rotation matrix, as described in section 2.2, then we can show that these 3-D linear subspaces precisely characterize the sets of x and y coordinates that the model can produce.

We now show how a similar result is evident from our work. We have shown that in the space formed by the affine basis and affine coordinates of an object, $(\mathbf{o}, \mathbf{u}, \mathbf{v}, \alpha_4, \beta_4, ...\alpha_n, \beta_n)$, each model's images lie in an 8-d linear subspace, that includes a plane in α-β space, and any possible values for $\mathbf{o}, \mathbf{u}, \mathbf{v}$. Similarly, when translation is included, the linear combinations approach shows that a model's images form an 8-d linear subspace in cartesian coordinates, which is an equivalent representation. The difference between the approaches is that we ignore the $\mathbf{o}, \mathbf{u}, \mathbf{v}$ parameters, producing a 2-D linear subspace which may be factored into two 1-D linear subspaces. In the linear combinations approach, the 8-D subspace may be factored into two 4-D subspaces. Our approach also implies a result similar to the one and a half views result described above. Given the α coordinates of any two views of a model, we may determine the line in α space that describes all the α coordinates the model might produce. In fact, any point on this line is a linear combination of the original two points used to determine the line. And since the directions of the α and β lines are the same, if we are given the α coordinates of two images of a model, and the β coordinates of one image, we may also determine the line in β space that describes the model.

The primary difference between our result and linear combinations, then, is the dimensionality of the linear spaces we produce. This is in fact the crucial problem of indexing; how can we most efficiently represent the images a model produces? The linear combinations work does not address this problem because it is not concerned with indexing. The linear combinations result is used rather for representing models and reconstructing new views of these models.

In addition to the implications for indexing, there is also a significant gain in conceptual clarity when we lower the dimensionality of our representation of images.

It is hard to formalize this, but it is often easier to visualize the matching problem if we can talk in terms of matching points to lines instead of matching them to 3- or 4-D linear subspaces. And if we attempt to make use of tools from computational geometry in performing this matching, we can expect that the complexity of these tools may well depend on the dimensionality of the objects we are matching.

3.1.2 Oriented Point Features

We now use results from chapter 2 to show that the linear combinations result can not be extended to oriented points. To do this it will be sufficient to consider the case where each model consists of three points and three vectors. Recall that in this case we may in general represent a model's images with a 2-D hyperboloid in a 3-D space. It might seem obvious from this that the linear combinations idea will not apply. Given a 2-D hyperboloid in a 3-D space, it is easy to pick four points on the hyperboloid that span the entire 3-D space. This means that in general, any four images of any model can be linearly combined to produce any possible image, and the linear combinations idea is true only in the trivial sense that with enough images we may express any other image as a linear combination of those images.

However, things are not this simple. Linear combinations might be true of one representation of images, but not true of another. For example, with point features the cartesian coordinates of one image are linear combinations of other images of the same model, but this might not be true of polar coordinates. So we must prove that all images of a model are not a linear combination of a small set of images, regardless of our choice of representation for an image. Since we know that the three basis points of the image convey no information about the model, the real question is whether some alternate representation of affine slope might map each model's images into a linear subspace. So we ask whether there is a continuous, one-to-one mapping from affine slope space, that is the space defined by $(\theta_0, \theta_1, \theta_2)$, into another space which maps every hyperboloid in affine slope space into a linear subspace. From elementary topology we know that any continuous one-to-one mapping will map our 3-D affine slope space into a space that is also 3-D, and that it will map every 2-D hyperboloid into a 2-D surface. So the question is whether these hyperboloids might map to 2-D planes in a 3-D space? That is, can we choose a different affine invariant representation of orientation vectors so that a model's images form a 2-D plane in the new space given by this representation?

To answer this, we must look at the particular set of hyperboloids that correspond to possible models. We assume that an appropriate mapping exists for linear combinations, and derive a contradiction. First, we recall that the line $\theta_0 = \theta_1 = \theta_2$ is part of the equation for each hyperboloid corresponding to a possible model. Call this line L. L is a degenerate case; the actual set of images a model produces does not include L, but it includes images that are arbitrarily close to L. Suppose we apply a continuous one-to-one mapping, call if f, that takes one of these hyperboloids, H, to a plane, $f(H)$. Then $f(L)$ is a 1-D curve such that for any point on the curve, there is a point on $f(H)$ arbitrarily close to that curve point. This can only happen if $f(L)$ lies on $f(H)$. That is, we can omit $f(L)$ from a model's manifold without problems, but we have shown that if this manifold is linear, then the requirement that our representation be continuous tells us that $f(L)$ must lie in this linear space.

Since L is part of every model's hyperboloid, this means the $f(L)$ must be a 1-D curve at which all models' manifolds intersect, in our new space. If all models' manifolds are 2-D planes in this new space, they can only intersect in a line. So $f(L)$ must be a line at which all models' planes intersect. But this means that no models' planes can intersect anywhere else in our new space. However, we have already shown that in general all the hyperboloids that represent models intersect at other places than the line L. f must preserve these intersections, so a contradiction is derived. This tells us that it is never possible to represent the images produced by a model of oriented points using linear combinations, except in the trivial sense.

The implications of this result, however, depend on what one thinks is important about the linear combinations result. If it is the linearity of the images, then our result concerning oriented points is a significant setback. It does seem that part of the impact of the linear combinations work is that the linearity of a model's images was unexpected and striking. And it is certainly true that linear spaces can lead to simpler reasoning than non-linear ones. However, a large part of the importance of the linear combinations work is that it provides a simple way of characterizing a model's images in terms of a small number of images, without explicitly deriving 3-D information about the model. And we may still do that with oriented points. Our computations are no longer linear, but we may still derive a simple set of equations from a few images of oriented points that characterize all other images that could be produced by the model, without explicitly determining this model's 3-D structure. We explore this further in the next section.

3.2 Affine Structure from Motion

3.2.1 Point Features

Koenderink and van Doorn[65] and Shashua[95] have also noted that two views of an object can be used to predict additional views, and have applied this result to motion understanding. Koenderink and van Doorn show that the *affine structure* of an object made of 3-D points can be derived from two views, assuming scaled orthographic projection. Affine structure is that part of the object's geometry that remains unchanged when we apply an arbitrary 3-D affine transformation to the object's points. For example, given two views of five corresponding points, they compute a 3-D affine invariant representation of the fifth point with respect to the first four. Then, given the location of the first four points in a third image, the location of the fifth point may be determined. This result is particularly significant because it is known (Ullman[103]) that three views of an object are needed to determine the object's rigid structure when images are formed with scaled orthographic projection.

Our representation of a models' images as lines in α and β space is fundamentally equivalent to Koenderink and van Doorn's 3-D affine invariant representation of a model. First, both representations factor out the effects of translation. Our representation then considers the set of images that a model can produce when a general $3x2$ matrix is applied to the model points. Koenderink and van Doorn assume that the model may be transformed with a general $3x3$ matrix and then projected into the image. Projection eliminates the effects of the bottom row of the matrix. Therefore the set of images that a model can produce with our projection model is the same as the set of images that a model can produce when it is transformed with a 3-D affine transformation and then projected into an image with scaled orthographic projection. The two methods represent the same information, but Koenderink and van Doorn's representation makes explicit what we know about an object's 3-D structure, while we make explicit what we know about the images that a model can produce.

For this reason, it is easy for us to rederive Koenderink and van Doorn's applications of this result to motion. As we pointed out above, two views of an object (actually, $1\frac{1}{2}$ views) suffice to determine the images that the object can produce. Given the location of four points in a third view we may determine (α_4, β_4) in this view. These values can be used to determine the affine coordinates of all remaining points, because they determine a single point on each line in α and β space.

3.2.2 Oriented Point Features

We may now consider what happens when we try to extend Koenderink and van Doorn's result to oriented point features. We find that four views are needed to determine the affine structure of some oriented points. We consider a model with three points and three orientation vectors; larger models may be handled similarly. From chapter 2 we know that for any hyperboloid of the following form:

$$-c_4xz + (c_4 - c_3)yz + c_1z + c_3xy - (c_1 - c_2)x - c_2y = 0$$

there is a model whose images are described by this hyperboloid, where x, y, z are the affine slopes of the three image vectors, and c_1, c_2, c_3, c_4 are parameters of the model, which may take on any values. Determining the affine structure of the model is equivalent to finding the values of c_1, c_2, c_3, c_4. If we do not know these values, we do not know the set of images that the model can produce and so we can not know the model's affine structure.

Every image of the model gives us a single equation like the one above, which is a linear equation in four variables. We need four independent equations to solve for these variables, and hence we need at least four views of the object to find its affine structure. Given three views of the object, there will still be an infinite number of different hyperboloids that might produce those three images, but that would each go on to produce a different set of images.

This result is easily extended to four or more oriented points. However, it only takes three views to determine the rigid structure of four or more oriented points. To compute this, we can first use the locations of the points in three views to determine their 3-D location, as shown by Ullman[103]. This tells us the 3-D location of each oriented point and each viewing direction, but not the 3-D direction of the orientation vectors. A view of an orientation vector at a known 3-D location restricts that vector to lie a plane. That is, the vector in the image gives us the x and y, but not the z coordinates of the unknown 3-D vector in the scene; and so all vectors that fit these x and y coordinates lie in a plane. So, for each orientation vector, two views tell us two different planes that include the vector. As long as our viewpoints are not identical, these planes intersect in a line, which tells us the direction of the orientation vector.

It might seem paradoxical that from three views we can determine the rigid structure of oriented points, while we need four views to determine their affine structure. But keep in mind that a view of an object provides us with less information about the object if we assume the view was created with a linear transformation than if we assume scaled orthographic projection.

This result is interesting because it demonstrates a significant limitation to extending the affine structure from motion work. Koenderink and van Doorn suggested that affine structure is an intermediate representation that we can compute with less information than is required to determine rigid structure. However, we see that this is not always true.

3.3 Conclusions

We see that our results from chapter 2 subsume some past work that has been done with linear transformations. Both linear combinations and affine structure from motion are obvious implications of our results, which also provide a low-dimensional representation of a model's images. We also see that just as indexing is fundamentally harder with oriented point features than with simple points, other results derived for point features cannot be extended to oriented point features. This calls into question the relevance of these results to the interpretation of complex images.

Chapter 4

Building a Practical Indexing System

Until now we have discussed indexing rather abstractly. We have focused on determining the best continuous representation of a model's images in the absence of error. To perform indexing with a computer we must represent these images discretely. For our system to work on real images we must account for sensing error. This chapter will address these issues.

To take stock of the problems that remain for us, let us review the steps that are performed by the indexing system that we have built.

1. We apply lower-level modules to images of the model at compile time, as part of the model building process, and to images of the scene at run time. These include:

   (a) Edge detection.

   (b) Straight-line approximations to edges.

   (c) Grouping. We use line segments to locate groups of point features that are likely to come from a single object. The grouping module described in chapter 6 outputs groups of points along with some information about how they should be ordered.

2. At compile time:

   (a) We find ordered collections of point features in the model that the grouping system is likely to group together in images.

   (b) We determine the lines in α and β space that describe the images that each ordered sequence of points can produce, using the results of chapter 2.

   (c) We discretely represent these images in hash tables.

3. At run time:

   (a) We find ordered groups of points in an image.

   (b) For each ordered group, we look in the hash tables to find matching groups of model points.

   (c) Indexing produces matches between a small number of image and model points, which we use to determine the location of additional model features. We use these extra features to verify or reject the hypothesized match.

We will discuss grouping in chapter 6. In this chapter, we describe the remaining issues: points 2b, 2c, 3b, and 3c. We first show how to fully account for the effects of error in our system by analytically characterizing the models that are compatible with an image when we allow for bounded amounts of sensing error. This allows us to build an indexing system which is guaranteed to find all feasible matches between an image group and groups of model features. We then discuss some of the issues involved in discretizing our index spaces. Finally, we show that our representation of a model's images lends itself to a simple method of determining the appearance of the model based on the location in the image of a few of its points.

We use these results to build an indexing system, and then we measure its performance. We answer four questions: how much space does the system require? how much time does it require? how many matches does it produce? and is the system accurate? The space requirements of the system are easily measured. In addition to some fixed overhead, the run time of the system will depend on how many cells we must examine in index space in order to account for error. The number of matches that the system will produce tells us the speedup that indexing can provide over a raw search through all possible model groups. Additionally, we measure the effect on this speedup of a number of different simplifications that we have made. We use a linear transformation instead of scaled orthographic projection. Since a linear transformation is more general, a set of model points might be matched to a set of image points with a linear transformation, but not a scaled orthographic transformation. And we make some simplifications in order to handle error. We also simplify when we discretize our space. So we run experiments to individually determine the effects of each of these choices. Finally, we need to check that our indexing system will produce the correct matches, that it will match real image points to the model points that actually produced them. We are assured that this will happen if our assumptions about error are true in the world, but we need to test these assumptions empirically. In the end, we have a practical indexing system whose performance we can characterize both theoretically and empirically.

4.1 Error

In chapter 2 we described how to represent a model's images as if there were no sensing error. To handle real images and models, though, we must account for some error. We choose a simple bounded error model. We assume that there is uncertainty in the location of each image point, but that this uncertainty is no more than ϵ pixels. That is, the actual, error-free image point must lie within ϵ pixels of the sensed point. We do not attempt to make any use of a probability distribution on the amount of error, or to characterize it in any other way. Of course we expect that occasionally this bounded error assumption may be violated, causing us to miss a potential match, just as we expect to miss other possible matches through occlusion or through failures in feature detection. We also assume that our model of the object is essentially error-free. This is partly because we can form very good models of the 3-D structure of an object by measuring it, if necessary, and partly because we assume that any error that does occur in the model can be thought of as contributing a small additional error to the image points.

One could imagine now extending our previous work by characterizing the set of images that a model can produce from all viewpoints, considering all possible amounts of error. Then we could represent all those images in a lookup table, and given a new image, look at a single point to find all the models that could produce exactly that image. This would not be a good idea. The problem is that our reduction in the dimensionality of an model's manifold in image space would not be applicable if we also allowed for error. We saw that we could characterize all the α coordinates that a model can produce without reference to the actual location of the model's first three points in the image, or to the model's β coordinates. However, the effect of a small amount of error on these affine coordinates depends very much on the particular location of the image points used as an affine basis. For example, if our first three image points form a right angle and are far apart, then a small change in the location of one of these points may have only a small effect on the affine coordinates of a fourth point. If our first three image points are nearly collinear, then a small change in one of the points can make them arbitrarily close to collinear, which causes the affine coordinates of another point to grow arbitrarily large. Figure 4.1 illustrates this effect. If we cannot ignore the locations of our first three image points, then the dimensionality of our models' manifolds will have to grow in order to include all this information. We could not possibly represent such big manifolds discretely.

So instead we account for error at run time. Then when we consider the effect of error we only have to deal with the scale and particular configuration of points in a single known image, and determine volumes in affine space that represent all the images consistent with that image and bounded error. That is, these volumes represent all the affine coordinates that our image could produce when its points are

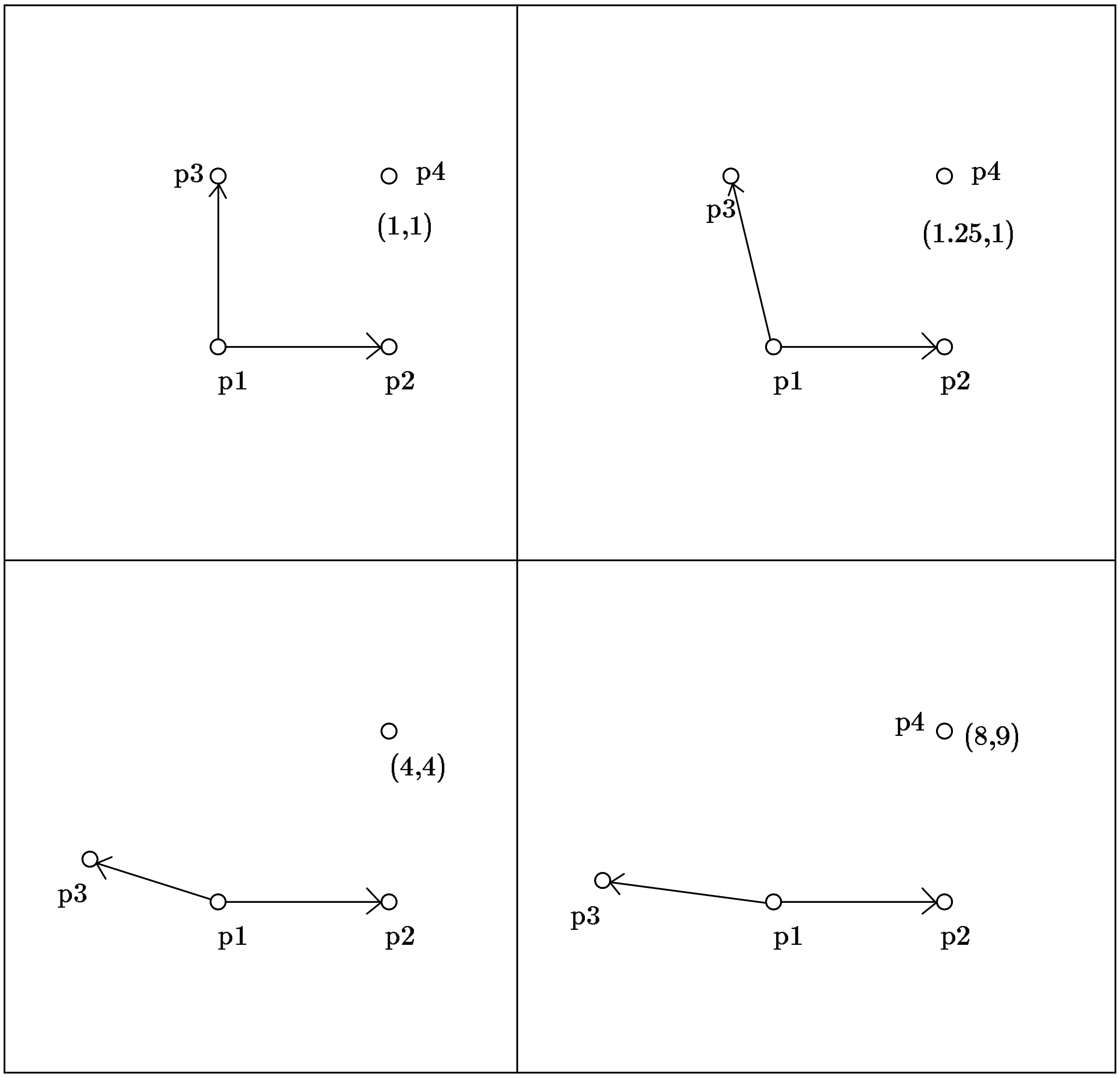


Figure 4.1: On the top, we show a fairly stable basis. A small change in $\mathbf{p_3}$ (upper right) has a small effect on the affine coordinates of $\mathbf{p_4}$. On the bottom an equally small change in $\mathbf{p_3}$ has a much larger effect on these affine coordinates.

perturbed within error discs. If we account for error exactly, then the results are equivalent whether we consider error at compile time, matching a thickened manifold to a point, or at run time, matching a manifold to a thickened point. Either way, an image would be matched to a model if and only if the model could produce that image, allowing for perturbations from error. The difference between accounting for error at compile time or run time then lies in the ease of implementing either approach, and in a trade-off of space for time. Accounting for error at compile time would require more space, but would allow us to index at a single point instead of over a volume. While in general we prefer to accept a space penalty to save run time, in this case the space required to account for error at compile time is too great; it is not practical to attempt to represent high-dimensional manifolds that have been thickened by error.

We determine the exact affine coordinates that are consistent with a noisy image only for the case of four image points. The difficulty with handling more points is that perturbations in the first three points will affect the affine coordinates of the other points all at once. So while we can determine what affine coordinates the fourth point has as we perturb the first three points, and we can determine what affine coordinates the fifth point has, we do not determine which affine coordinates they both produce at once, and which ones they can each produce, but not at the same time. Our results do allow us to provide a conservative bound on the affine coordinates compatible with an image, because we can individually bound the affine coordinates of each point. For each point, we determine the maximum and minimum α and β coordinates it can produce. We combine these bounds to find rectanguloids in α and β space that contain all the α and β coordinates that could be compatible with the image. This process is shown schematically in figure 4.2. This allows us to build an indexing system in which we analytically compute volumes in index space. By looking in these volumes for models to match an image, we are guaranteed to find all legitimate matches to the image. But we may also find superfluous matches as well.

4.1.1 Error with Four Points

We use a somewhat round-about route to find the affine coordinates that four points can produce when perturbed by bounded error. We begin by considering a planar model of four points, and an image of three points. Projection of the planar model is described by an affine transformation, and the affine coordinates of the fourth model point are invariant under this transformation. Given a match between three image and model points, it is simple to determine the nominal location that the fourth model point should have, in the absence of error. We then describe the locations it may have when we account for bounded error in each image point. We call this set of locations the *potential locations* of the point. We show that the potential locations of

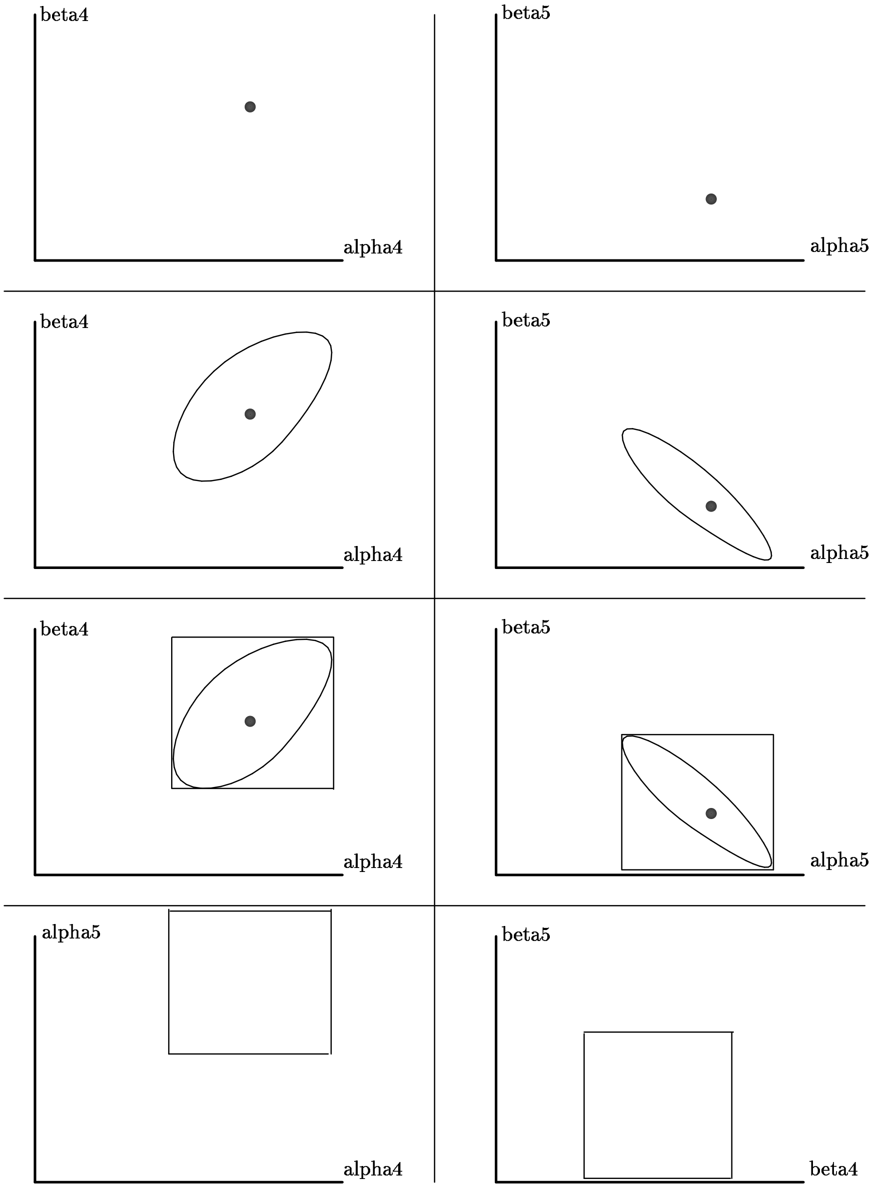


Figure 4.2: Without error, the fourth and fifth image points have individual affine coordinates, as shown on top. When we precisely account for error in each point separately, we find regions of α_4-β_4 space consistent with the fourth point, and regions of α_5-β_5 space consistent with the fifth point. We simplify by placing rectangles about these regions. Then we may represent this error equivalently with rectangles in α_4-α_5 space and β_4-β_5 space.

a fourth model point are described by a surprisingly simple expression which depends only on the affine coordinates of the point. Using this expression, we can determine whether a set of four model points is compatible with four image points. We show that this is equivalent to determining which affine coordinates the four image points can produce.

Consider four model points, $\mathbf{p_1}$, $\mathbf{p_2}$, $\mathbf{p_3}$, $\mathbf{p_4}$, and three image points, $\mathbf{q_1}$ $\mathbf{q_2}$, and $\mathbf{q_3}$. We match the three image points to the corresponding model points ($\mathbf{q_1}$ matched to $\mathbf{p_1}$ etc...). Let α_4 and β_4 be the affine coordinates of the fourth model point with respect to the first three model points, which are defined because the model is planar.

Let us describe the sensing error with the vectors $-\mathbf{e_i}$. This means that the "real" location to which the i'th model point would project in the absence of sensing error is $\mathbf{q_i} + \mathbf{e_i}$. Let $\mathbf{q_i'} = \mathbf{q_i} + \mathbf{e_i}$. Then our assumptions about error are expressed as: $||\mathbf{e_i}|| \leq \epsilon$. Let $\mathbf{u'} = \mathbf{q_2} - \mathbf{q_1}$ and $\mathbf{v'} = \mathbf{q_3} - \mathbf{q_1}$. Let $\mathbf{r_4} = \mathbf{q_1} + \alpha_4\mathbf{u'} + \beta_4\mathbf{v'}$. That is, if we use the match between the first three image and model points to solve for an affine transform that perfectly aligns them, and apply this transform to $\mathbf{p_4}$, we will get $\mathbf{r_4}$.

Let $\mathbf{q_4'}$ stand for the "real" location of $\mathbf{p_4}$ in the image, that is, the location at which we would find $\mathbf{p_4}$ in the absence of all sensing error. $\mathbf{p_4}$ will actually appear in the image somewhere within a circle of radius ϵ about $\mathbf{q_4'}$, because of the error in sensing $\mathbf{p_4}$'s projection. So, our goal is to use $\mathbf{q_1}, \mathbf{q_2}, \mathbf{q_3}, \mathbf{p_1}, \mathbf{p_2}$ and $\mathbf{p_3}$ to first determine the set of potential locations of $\mathbf{q_4'}$, and then thicken this set by ϵ to find all the potential locations where we might sense the projection of $\mathbf{p_4}$.

For a particular set of error values,

$$\mathbf{q_4'} = (\mathbf{q_1} + \mathbf{e_1}) + \alpha_4((\mathbf{q_2} + \mathbf{e_2}) - (\mathbf{q_1} + \mathbf{e_1})) + \beta_4((\mathbf{q_3} + \mathbf{e_3}) - (\mathbf{q_1} + \mathbf{e_1}))$$

This is because $(\mathbf{q_1}+\mathbf{e_1}, \mathbf{q_2}+\mathbf{e_2}, \mathbf{q_3}+\mathbf{e_3})$ are the locations of the error-free projections of the model points in the image. So we can find $\mathbf{q_4'}$ using these three points as a basis, knowing that $\mathbf{q_4'}$ will have the same affine coordinates with respect to this basis that $\mathbf{p_4}$ has with respect to a basis of $(\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3})$.

$$\begin{aligned}
\mathbf{q_4'} &= \mathbf{q_1} + \mathbf{e_1} + \alpha_4\mathbf{q_2} + \alpha_4\mathbf{e_2} - \alpha_4\mathbf{q_1} - \alpha_4\mathbf{e_1} + \beta_4\mathbf{q_3} + \beta_4\mathbf{e_3} - \beta_4\mathbf{q_1} - \beta_4\mathbf{e_1} \\
\mathbf{q_4'} &= \mathbf{q_1} + \alpha_4(\mathbf{q_2} - \mathbf{q_1}) + \beta_4(\mathbf{q_3} - \mathbf{q_1}) + \mathbf{e_1} + \alpha_4\mathbf{e_2} - \alpha_4\mathbf{e_1} + \beta_4\mathbf{e_3} - \beta_4\mathbf{e_1} \\
&= \mathbf{r_4} + \mathbf{e_1}(1 - \alpha_4 - \beta_4) + \mathbf{e_2}\alpha_4 + \mathbf{e_3}\beta_4
\end{aligned}$$

When we allow the $\mathbf{e_i}$ to range over all vectors with magnitude less than ϵ, this defines a region of potential locations of $\mathbf{q_4'}$. Note that $\mathbf{r_4}$ depends only on the location of the image points, and is independent of the error vectors.

In the above expression, $\mathbf{r_4}$ is fixed and the expressions involving $\mathbf{e_1}, \mathbf{e_2}$, and $\mathbf{e_3}$ can each alter the values of $\mathbf{q_4'}$ within circles of radii $\epsilon|1 - \alpha_4 - \beta_4|$, $\epsilon||\alpha_4|$, and $\epsilon|\beta_4|$

respectively. Since each of these expressions is independent of the others, adding them together produces a circle centered at $\mathbf{r_4}$ whose radius is the sum of the radii of each of the individual circles, that is, a circle of radius: $\epsilon(|1-\alpha_4-\beta_4|+|\alpha_4|+|\beta_4|)$.

When we consider that we may have an error of ϵ in sensing the fourth image point as well, this expands the region by ϵ. We find that the potential locations of the fourth image point, such that there exists some orientation and bounded sensing error that aligns the image and model points, is a circle, centered at $\mathbf{r_4}$, with radius $\epsilon(|1-\alpha_4-\beta_4|+|\alpha_4|+|\beta_4|+1)$

This result has a surprising consequence. For a given set of three model points matched to three image points, the size of the space of potential locations of a fourth model point in the image will depend only on the affine coordinates of the fourth point. It will not depend on the appearance of the first three model points. That is, it will not depend on the viewing direction. Even if the model is viewed nearly end-on, so that all three model points appear almost collinear, or if the model is viewed at a small scale, so that all three model points are close together, the size of the potential locations of the fourth model point in the image will remain unchanged.

However, since the viewing direction does greatly affect the affine coordinate system defined by the three projected model points, the set of possible affine coordinates of the fourth point will vary greatly. For example, changes in the scale of projection simply shrink the affine coordinate frame, and so multiply the size of the feasible region for the fourth image point in this frame. We must account for this variation in affine coordinates in order to perform indexing. The fact that this variation depends on the configuration of the image points that are used as a basis explains why it is convenient to account for error at run time, when the locations of the image basis points are known.

Partial solutions to the problem of determining the effect of error on planar model matching have previously been produced in order to analyze the performance of different recognition algorithms. In Huttenlocher and Ullman[57], the affine transform is found that aligns three image and model points. This is used to find the pose of a three-dimensional model in the scene. Remaining model points are then projected into the image, and matching image points are searched for within a radius of 2ϵ of the projected model points. To analyze the effectiveness of using an error disc of 2ϵ when matching projected model points to image points, Huttenlocher[56] considered the case of planar objects, as we have. He was able to show bounds on the potential locations of model points in the image in certain simple cases. These bounds depended on assumptions about the image points. Here we have shown exact bounds on the potential location of model points, and we show that these bounds do not depend on characteristics of the image points. For example, we may now see exactly when a circle of 2ϵ, which alignment uses, is the correct description of the potential location of a projected model point (when $0 < \alpha_4, \beta_4$ and $\alpha_4 + \beta_4 < 1$), and when it is too small.

In order to analyze the geometric hashing approaches to recognition of Lamdan, Schwartz and Wolfson[70], researchers have also considered the effects of error on the affine coordinates that are compatible with four image points. Recall that in geometric hashing, the invariance of the affine coordinates of four planar models is used for indexing. Lamdan and Wolfson[72] and Grimson and Huttenlocher[48] have placed bounds on the set of affine coordinates consistent with an image under special assumptions, and used these bounds to analyze the effectiveness of geometric hashing. Costa, Haralick, and Shapiro[34] have also discussed methods for estimating the potential numerical instability of affine coordinates, and suggested ways of dealing with this instability. We are now in a position to precisely characterize the set of affine coordinates that are consistent with a set of image points, assuming bounded error. This characterization has also been used to analyze the effectiveness of alignment and geometric hashing, in Grimson, Huttenlocher and Jacobs[49].

Suppose we have image points, $\mathbf{q_1}, \mathbf{q_2}, \mathbf{q_3}$ and $\mathbf{q_4}$. Let the affine coordinates of $\mathbf{q_4}$ with respect to $(\mathbf{q_1}, \mathbf{q_2}, \mathbf{q_3})$ be $(\bar{\alpha}, \bar{\beta})$. If a fourth model point has affine coordinates (α_4, β_4) with respect to the three other model points, we would like to know whether the model could match the image. If we match the first three image and model points, we know that the fourth model point will match any image point within $\epsilon(|1-\alpha_4-\beta_4|+|\alpha_4|+|\beta_4|+1)$ of $\mathbf{r_4}$, where here $\mathbf{r_4} = \mathbf{q_1} + \alpha_4(\mathbf{q_2} - \mathbf{q_1}) + \beta_4(\mathbf{q_3} - \mathbf{q_1})$. So the model and image can match if and only if the distance from $\mathbf{r_4}$ to $\mathbf{q_4}$ is less than or equal to: $\epsilon(|1 - \alpha_4 - \beta_4| + |\alpha_4| + |\beta_4| + 1)$. That is, if and only if:

$$||\mathbf{q_4} - (\mathbf{q_1} + \alpha_4(\mathbf{q_2} - \mathbf{q_1}) + \beta_4(\mathbf{q_3} - \mathbf{q_1}))|| \leq \epsilon(|1 - \alpha_4 - \beta_4| + |\alpha_4| + |\beta_4| + 1)$$

that is:

$$\begin{aligned} & ||(\mathbf{q_1} + \bar{\alpha}(\mathbf{q_2} - \mathbf{q_1}) + \bar{\beta}(\mathbf{q_3} - \mathbf{q_1})) - (\mathbf{q_1} + \alpha_4(\mathbf{q_2} - \mathbf{q_1}) + \beta_4(\mathbf{q_3} - \mathbf{q_1}))|| \\ = \; & ||(\bar{\alpha} - \alpha_4)(\mathbf{q_2} - \mathbf{q_1}) + (\bar{\beta} - \beta_4)(\mathbf{q_3} - \mathbf{q_1})|| \\ \leq \; & \epsilon(|1 - \alpha_4 - \beta_4| + |\alpha_4| + |\beta_4| + 1) \end{aligned}$$

Following Grimson, Huttenlocher and Jacobs, if we let $u = ||\mathbf{q_2} - \mathbf{q_1}||, v = ||\mathbf{q_3} - \mathbf{q_1}||$ and let ψ be the angle formed by the vectors from $\mathbf{q_1}$ to $\mathbf{q_2}$ and $\mathbf{q_3}$, then the above equation becomes:

$$\begin{aligned} & ((\bar{\alpha} - \alpha_4)u)^2 + ((\bar{\beta} - \beta_4)v)^2 + 2(\bar{\alpha} - \alpha_4)(\bar{\beta} - \beta_4)uv(\cos\psi) \\ & \qquad \leq \epsilon^2(|1 - \alpha_4 - \beta_4| + |\alpha_4| + |\beta_4| + 1)^2 \end{aligned}$$

Note that all the values in this equation are derived from the four image points, except for α_4 and β_4. So this equation tells us which affine coordinates a model may have and still match our image, to within error bounds. This is the same as telling

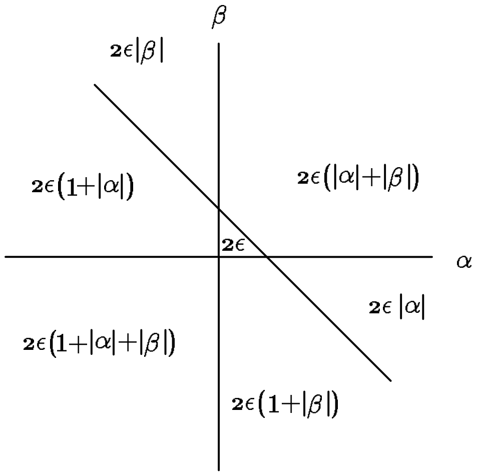


Figure 4.3: There are seven different expressions without absolute values that describe the radius of the error regions. Which expression applies to a model depends on whether $\bar{\alpha} < 0$, $\bar{\beta} < 0$ and $\bar{\alpha} + \bar{\beta} < 1$.

us which affine coordinates our image may produce if we perturb each image point by no more than ϵ pixels. So this equation describes a region of α_4-β_4 values that correspond to models that could produce our image.

We now wish to deal with the pesky absolute value signs in this equation, so that we can describe the boundary it gives. These absolute values mean that if we divide the α-β plane into seven parts, then the radius of the error circle for a model is described by a simple expression without absolute values which depends on which region its affine coordinates fall in, as shown in figure 4.3. We can find the boundary of affine coordinates consistent with an image by combining seven different, simpler boundaries.

As an example, for a particular image we just consider the models for which $\alpha_4 > 0$, $\beta_4 > 0$ and $\alpha_4 + \beta_4 > 1$. In that case, the radius of the error region associated with the model is $2\epsilon(\alpha_4 + \beta_4)$. So for a given image, one possible set of models consistent with that image are the models for which $\alpha_4 > 0, \beta_4 > 0, \alpha_4 + \beta_4 > 1$ and

$$\begin{aligned}
& ((\bar{\alpha} - \alpha_4)u)^2 + ((\bar{\beta} - \beta_4)v)^2 + 2(\bar{\alpha} - \alpha_4)(\bar{\beta} - \beta_4)uv(\cos\psi) \\
= \; & \epsilon^2(-(1 - \alpha_4 - \beta_4) + \alpha_4 + \beta_4 + 1)^2 \\
= \; & 4\epsilon^2\alpha_4^2\beta_4^2
\end{aligned}$$

This is the equation for a conic in α_4, β_4. Combined with the inequalities constraining α_4 and β_4 we get the intersection of a conic and a region in α_4-β_4 space, where this intersection may well be empty. In general, we can remove the absolute value signs by writing down seven different conic equations, and intersecting each one with a different region of α_4-β_4 space. When we intersect each conic with the appropriate region of α_4-β_4 space, we get a set of (α_4, β_4) values consistent with the image. The union of these seven sets of values gives us the entire set of values consistent with the image.

4.1.2 Error for Our System

We use this precise bound on the effects of error with four points to compute a conservative error volume for indexing. To do this, given an image group with n points, for each affine coordinate of α_4, ... α_n, β_4, ... β_n we compute the minimum and maximum value that coordinate can have. For each point, we compute the intersection of each of the seven conics with the appropriate region in affine space, and find the maximum and minimum α and β values that point can have over all seven possibilities. Taking the α and β values separately, we have two rectanguloids in α and β space. We use each rectanguloid to separately access each of the spaces. Looking at all the cells in index space that the α rectanguloid intersects, we are guaranteed to find all the models that could have produced α values compatible with our image. We find all the models with compatible β values as well, and then intersect the results to find all models consistent with our image. In principle the need for intersection reduces the asymptotic performance of our system, because the α values alone will have less discriminatory power than both α and β values combined. But in practice this intersection takes very little time.

This indexing method will produce unnecessary matches for two reasons. First, we are assuming that the affine regions compatible with each additional image point are independent. We do not take account of the fact that as one of the three basis points is perturbed by error, it will affect all the image's affine coordinates at once. Understanding the nature of this interaction is a difficult problem that has not been solved. Second, we are bounding each α value independently of its corresponding β value. This is equivalent to putting a box around the conics that describe a point's compatible α and β coordinates, a box whose sides are parallel to the α and β axes. These conics describe ellipses in most cases. If an ellipse is elongated, and on a diagonal to the α and β axes, then putting a box around it is a very conservative bound. This second simplification makes it easier to decide how to access our lookup table, but in principal we could compute which cells in α space are compatible with a particular set of cells in β space, and make our lookup more precise.

4.2 Building the Lookup Table

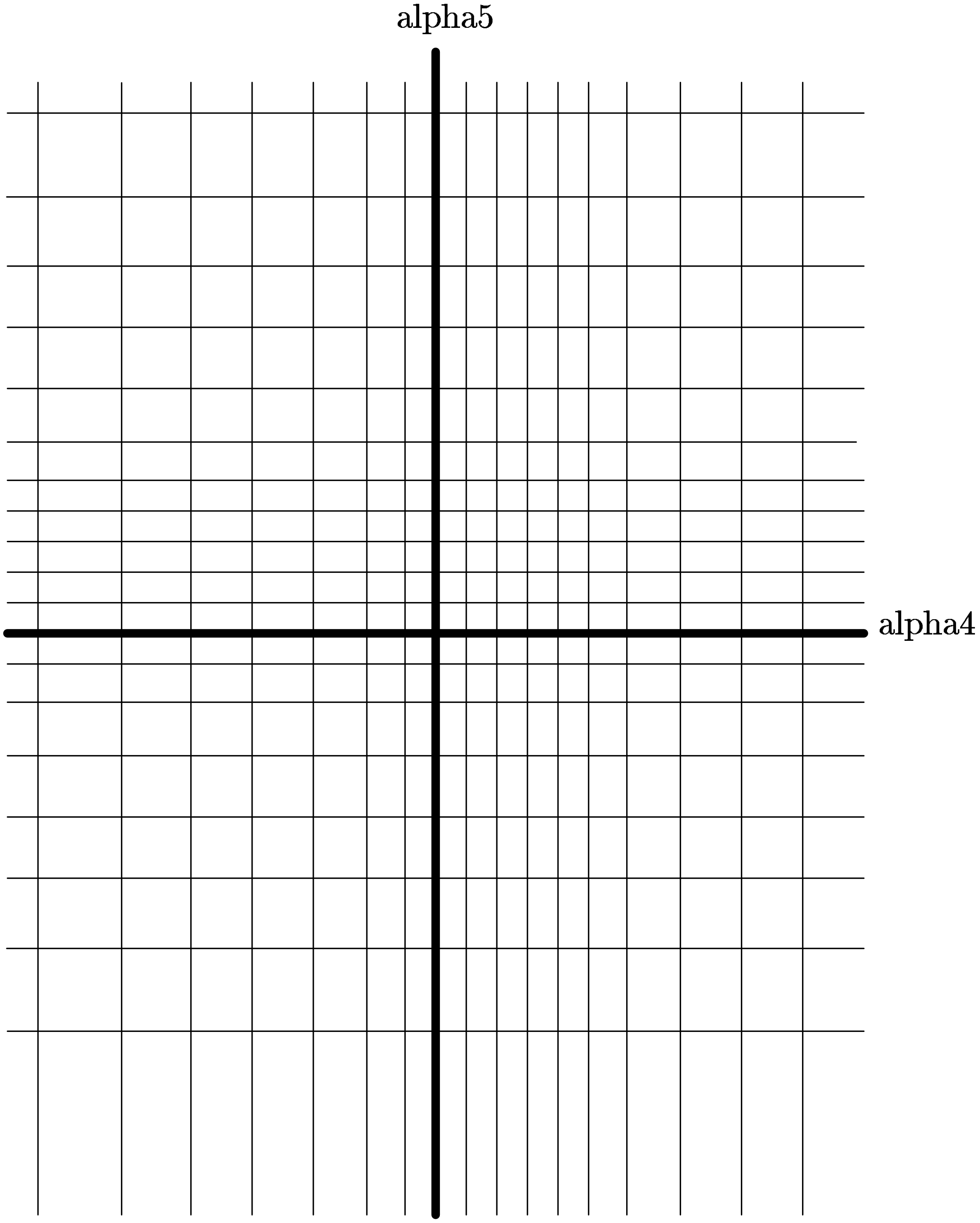


Figure 4.4: An example of the grid spacing used in the lookup table, for a two-dimensional table.

We have now described how to analytically map both images and models to regions in α and β space where we can match them. To implement this approach in a computer, we must divide α and β space into discrete cells so that we can finitely represent these continuous regions. We do this in a straight-forward way, dividing each dimension of each space into discrete units, so that the entire spaces are tesselated into rectanguloids. The only question that arises with this approach is in how an individual dimension of each space should be divided.

Because the error regions grow larger as the affine coordinates grow larger, we do not discretize the affine spaces uniformly. To decide how to discretize the spaces, we note that the size of our error regions is directly dependent on the location of the first three image points, which we cannot know at compile time, and on the expression: $\epsilon(1 + |1 - \alpha_4 - \beta_4| + |\alpha_4| + |\beta_4|)$. Let us consider how this second expression varies with α_4 for the simple case where $\beta_4 = 0$. This expression is a constant 2ϵ if the α_4 coordinate is between 0 and 1. Outside that region it grows linearly. So although the size of the error rectanguloids depends on many factors, we can determine the way in which their size varies with α_4 when other variables are held constant. We choose to discretize α and β space according to this variation. Therefore, we uniformly discretize each coordinate of the affine spaces for values between 0 and 1, and then discretize into ranges that grow linearly with the affine coordinate for other values. Figure 4.4 illustrates this discretization for a two-dimensional affine space.

We also can only represent a finite portion of the space, so we ignore all affine coordinates with an absolute value of 25 or greater. This threshold is set fairly arbitrarily, but it is easy to see that if a set of image points have affine coordinates greater than 25, then the size of the error regions they give rise to will also be quite large.

We ran an experiment to show that this is a good way to discretize the space. We randomly formed sets of four image points, by picking four points from a uniform distribution inside a square 500 pixels wide. For each set of four points, we computed the range of α values with which it was consistent, assuming error of five pixels. In Figure 4.5 we plot the middle of these ranges against their width, after averaging together ranges with similar middle values. This shows that it is true that the error ranges with which we will access the table are fairly constant between 0 and 1, and grow linearly outside those values. Note that the width of the error rectangles plotted in Figure 4.5 dip downward as α approaches 25 or -25, because we excluded α ranges that would not fit entirely in the lookup table. These experiments also show that we will access the extreme parts of the table with bigger rectanguloids, so it would be wasteful to discretize these parts finely.

Given our discretization of affine space, we then just compute which cells are intersected by each line that represents a model's images. At those cells we place a pointer to the group of model points that the line represents. Although groups of

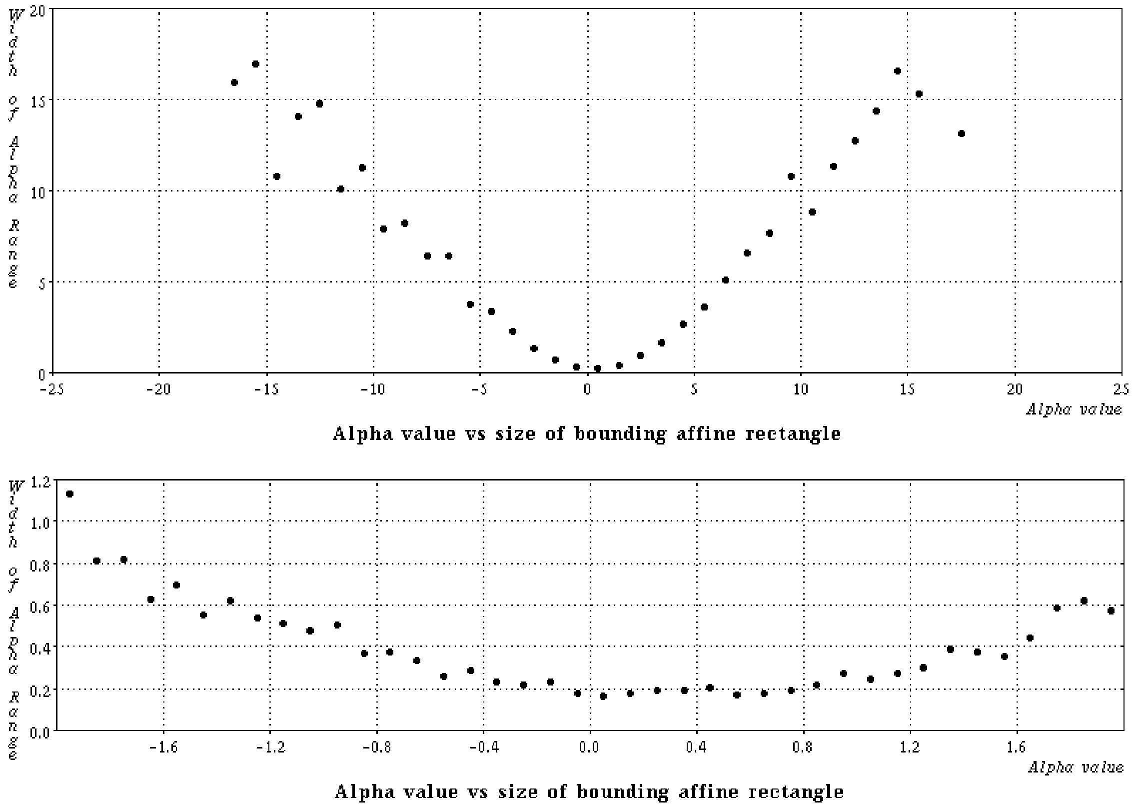


Figure 4.5: These graphs show how the size of error regions grow in one of their dimensions as a function of one of the affine coordinates of the fourth image point. The bottom figure is a blowup of the graph, near 0. We can see from the above figures that the size of error rectangles is constant for $0 \leq \alpha_4 \leq 1$, and grows linearly outside that range.

different sizes are represented by lines in spaces of different dimensions, we physically store all active cells in a single α and a single β hash table.

We could compute the lines in α and β space that describe a group of point features easily from their 3-D structure. Instead, it is more convenient to determine the lines directly from a series of 2-D images of the object. We use the system described in chapter 6 to find groups of point features of an object in a number of different images. We then establish a correspondence between these point features by hand. Therefore, for a particular group of ordered model points, we have available the 2-D locations of these points in a set of images. In all images we use the same three points as a basis, and find the affine coordinates of the other points with respect to this basis. We are deferring until chapter 7 a discussion of how we order these groups of points, and how we choose points for the basis of the groups. For each image, then, we have a set of affine coordinates that we may treat as a point in α space and a point in β space. We then fit a line to the points in each of these spaces. This gives us two lines that describe all the images that the model could produce. We use a simple least squares method to do this line fitting. We could perhaps do better by taking into account the fact that the stability of each affine coordinate of each image is different, but we have not found that necessary.

4.3 Performing Verification

We have now described how to build and access our indexing table. We may also make use of some of the tools that we have developed to efficiently verify hypothetical matches produced by our indexing system (step 3c in the outline at the beginning of the chapter). Above, we show that we can build our lookup table without deriving an explicit model of the 3-D structure of an object. In the same way, we can also generate new images of the object, in order to perform verification. This is quite similar to the linear combinations work of Ullman and Basri[105]. The basic idea is that given a match between some image and model points, we have a point in affine space matched to a line in affine space. Due to image error, the point produced by the image will not fall exactly on the line corresponding to the model. By projecting the point onto the line, we can find a set of affine coordinates which the model could have produced, and which are near the affine coordinates found in the image. We can then use these affine coordinates to determine the affine coordinates of all the other model points.

To implement this, we first select line segments in images of the model that come from the object. We match the end points of these line segments by hand. Then we construct lines in α and β space, separate from the lines we use for indexing, which represent all the points on the object, including the ones we use for indexing and the endpoints of the line segments that model the object. For every triple of points that we will use as a basis for indexing, we also use this triple as a basis for describing all the points.

Let us illustrate this with an example. Suppose indexing matches model points $\mathbf{p_1}$, $\mathbf{p_2}$, $\mathbf{p_3}$, $\mathbf{p_4}$, $\mathbf{p_5}$, $\mathbf{p_6}$ to image points $\mathbf{q_1}, \mathbf{q_2}, \mathbf{q_3}, \mathbf{q_4}, \mathbf{q_5}, \mathbf{q_6}$, and suppose based on this match we wish to project model points $\mathbf{p_1}, \mathbf{p_2}, ...\mathbf{p_n}$ into the image. At compile time, we use the points $\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}$ as a basis, and compute the two lines in the affine spaces that describe all images of the model when the image of these three points are used as a basis. (This is done at compile time for all possible basis triples). Call these two lines L_1 and L_2. These lines are in $(n-3)$-dimensional affine spaces, $(\alpha_4...\alpha_n)$ and $(\beta_4...\beta_n)$, because they represent the locations of $n-3$ points using the first three points as a basis. Our six matched image points map to two points in the 3-dimensional affine spaces $(\alpha_4, \alpha_5, \alpha_6)$ and $(\beta_4, \beta_5, \beta_6)$. Call these points $\mathbf{a_1}$ and $\mathbf{b_1}$. By projecting L_1 and L_2 into these lower dimensional spaces, we get lines that describe the possible images that the first six model points can create. By finding the point on the projection of L_1 closest to $\mathbf{a_1}$ we find the α coordinates of the image of the model that best match the image points. Similarly, we find the appropriate β values. These values determine locations on L_1 and L_2 that tell us the affine coordinates of all the model points in the image that will best fit the matched image points. Without explicitly computing the viewing direction we have computed the appearance of the model, when seen from the correct viewing direction. (A different method must be used if the matched model points are coplanar, because in that case their affine coordinates provide no information about viewing direction).

In addition to determining the effects of the viewing direction on the image, we must also allow for the effects of the affine transformation portion of the projection. However, once we have determined the affine coordinates of all the projected model points, it is straightforward to apply a least squares method to find the affine transformation that optimally aligns the image points with the projected model points.

4.4 Experiments

This section describes experiments measuring the performance of the indexing system. The main issue at compile time is the space required during step 2c, when the hash table is built. At run time, we will measure the number of steps required to access the table during step 3b, and the number of matches that this step produces. In the experiments described in this section, we represented a model of a telephone in our indexing tables. In chapter 7 we will describe how we formed groups of point features on the telephone. For our purposes in this chapter, it is sufficient to say that we explicitly represented 1,675 different groups of between six and eleven ordered point

Indexing Space	
Group Size	Number Entries
6	110
7	128
8	154
9	198
10	230
11	270

Table 4.1: This table shows the average amount of space required to represent the lines that correspond to our model's images, with $d = 50$. Each row of the table shows the average number of cells intersected by a line corresponding to a group of a particular number of points. This is the number of cells intersected in just one space, α or β.

features from the telephone, and measured the performance of the indexing system with this collection of groups.

We can analytically bound the space required by the indexing system. It will be at most linear in the discretization of the lookup table, and in the dimensionality of the space. A line passing through a high-dimensional space is monotonically increasing or decreasing in each dimension. Therefore, if we cut each dimension of the table into d discrete intervals, each time the line passes from one cell to another, one (or more) of the dimensions is changing value. There can only be d such transitions in each dimension, for a total of $(n-3)d$ transitions in the $n-3$ dimensional space in which a group with n points is represented. Therefore, the maximum space required to represent a group with two lines in two spaces is $2(n-3)d$. In table 4.1 we show the actual number of table entries made in experiments with $d = 50$. We can see that our bound on the space requirements is correct, and a reasonable approximation to the amount of space actually used.

The time required to perform indexing will depend on the number of cells at which we must look in the index table, and the number of entries found in these cells. The number of cells examined is exponential in the dimensionality of the table, because the volume of a rectanguloid grows exponentially in its dimension. If the side of a rectanguloid is typically smaller than the width of a cell in the table, than finer discretization will not increase the number of cells examined, but in general the number of cells examined grows with the level of discretization. Assuming for purposes of exposition that each side of each rectanguloid has width r, and that the whole index space has width M, than a rectanguloid will intersect $(\lceil \frac{rd}{M} \rceil)^{n-3}$ cells. This gives us a rough idea of the actual behavior of the system. In practice, we find that for $d = 100$ and $\epsilon = 5$ pixels, a rectanguloid typically intersects about 6^{n-3} cells in the table. This is starting to grow quite large, indicating that we should not tesselate the space any finer than this, and perhaps not quite so finely. On the other hand, keep in mind that the work performed for each cell is simply a hash table lookup.

We also find that occasionally we may index using a group of image points with an unstable basis or high affine coordinates. This can produce a very large rectanguloid that intersects a large number of cells. Looking at all these cells may require excessive computation. One solution is to simply ignore such image groups, which are likely to be less useful for recognition in any case. A better solution might be to discretize the space into several separate tables, using several different values of d. Then at run time, when we know the size of the rectanguloid that a particular group of image features produces, we may choose to look in a table with the appropriate resolution for that rectanguloid. This would guarantee both that we do not need to look in too many cells, and that our discretization would not introduce too much error.

When looking up a rectanguloid in an affine space, we have to take the union of everything we find in every cell at which we look. We then have to take the intersection of the result of looking in two rectanguloids in two spaces. These unions and intersections take time that is linear in the number of entries we find. If table entries are uniformly distributed throughout the lookup space, then, given that there are d^{n-3} cells in a space, and about $d(n-3)$ entries per model group, if there are N model groups than there is an average of

$$\frac{N(n-3)}{d^{n-4}}$$

entries in each cell, for a total number of objects looked at by a rectanguloid of about:

$$N(n-3)d\left(\frac{r}{M}\right)^{n-3}$$

For the values mentioned above ($d = 100$, $\frac{dr}{M} = 6$) and for $n = 7$, and $N = 1000$, we would expect about five objects to be found by each rectanguloid. This should be an underestimate, however, because in reality objects will not be uniformly distributed about the index space. We would expect both models and images to form clusters.

Overall we can see that indexing requires reasonable amounts of time if we are careful when deciding how to discretize the table. Space requirements are also modest for a single group of model points, although they may become an issue if we need to represent large numbers of model groups.

Until now, our discussion just shows that indexing can be practical in terms of space and time. The most important question, though, is how useful can indexing be? How much can we gain by using a lookup table to find geometrically consistent matches instead of explicitly considering all matches. To determine this we want to measure the speedup provided by indexing, that is, for a group of n image points, we want to know the likelihood that a group of n model points that did not produce these image points will still be matched to them by indexing. We will call the inverse of this likelihood the *speedup* the system provides, because this is the reduction in the number of matches that we consider with indexing, as compared to raw search.

There are several factors that might reduce the speedup of our system, and we wish to measure them individually. First, we are using a linear transformation for indexing, which will match an image to more models than would scaled orthographic or perspective projection. Second, we make two somewhat different approximations when we use rectanguloids as the volumes that access the lookup table: we are placing a rectangle about the error region associated with four points, which is typically an ellipse, and we are assuming that error has an independent effect on the affine coordinates of each image point. Third, by representing the lookup table discretely, we will make approximations. We may match an image to a model because the model's line and the image's rectanguloid intersect the same cell, but do not intersect each other. So our goal is to determine the overall speedup that our system provides, and to separately measure the effect of each of these approximations.

We begin with some analytic comments on this speedup, and then present the results of experiments.

We first consider the speedup that an ideal indexing system can produce when images formed with scaled orthographic projection contain a bounded amount of sensing error.[1] The expected speedup will depend on n, and we denote it $s(n)$. We show that there are constants k and j such that $k^{n-3} \leq s(n) \leq j^{n-3}$, in the case where image points are chosen independently from a uniform random distribution on the image.

First we show that $s(n) \geq k^{n-3}$. The speedup for a given image group will depend on the number of model groups that can appear like the image group, within error bounds. Suppose n is four. We know that at least one pose of the model exists which will perfectly align the first three model points with the first three image points. About this pose will be a set of poses for which the three model points project to within fixed error bounds of the image points. As the first three points gyrate through this set of poses, the projection of the fourth model point sweeps out some region of the image. Call this region I_4 (this is the potential location, for 3-D models and scaled orthographic projection). If the fourth image point is within error bounds of I_4, then a pose exists that makes the model look like the image. Let I'_4 be those locations of the fourth point such that it is within error bounds of I_4. Under the

[1] The following analysis appeared in Clemens and Jacobs[32], and is joint work between the author and David Clemens.

uniform distribution assumption, the probability that a random image point will fall within I'_4 is the area of I'_4 divided by the image area. The inverse of this is the speedup produced by indexing with four points, compared to checking all models. We call this ratio k.

Suppose $n = 5$. The average speedup from indexing will be at least k^2. To see this, we can form a region I'_5 for the fifth model point in just the same way we formed I'_4. That is, I'_5 does not depend on the fourth model point or the fourth image point. There is a pose that aligns the image and model groups to within error bounds only if the fifth image point falls inside I'_5, *and* the fourth image point lies in I'_4. Call this event A_2. Since the two regions are constructed independently, and since the image points are chosen independently, the probability of A_2 is equal to the product of the probability of each event occurring separately. This implies a speedup of k^2. However, the speedup is even greater: event A_2 is a necessary condition by construction of I'_4 and I'_5, but not a sufficient condition, since it must also be the case that there is a *single* pose that aligns both points. In general, indexing will produce a speedup of at least k^{n-3}.

We now show that for some j, $s(n) \leq j^{n-3}$. The speedup of indexing can only be decreased by accounting for error. Therefore, if we ignore the error in the first three image points, but consider the error in subsequent points, we may derive a loose upper bound on $s(n)$. Even without varying the pose that aligns the first three points, there is an error region in the image due to the error in the fourth point. This error region will occupy some proportion, $1/j$, of the image area. For each additional point there is another error region of the same size, since error in each point is independent. Analogous to the lower bound, the upper bound is therefore j^{n-3}.

j is just $\frac{\pi\epsilon^2}{A}$, where A is the image's area. Suppose, for example, that the error bounds on each image point are a circle of radius five pixels, and the image is 500 pixels by 500 pixels. Then $j \approx 3200$. With radius three, $j \approx 8800$.[2]

This same argument applies equally well to a linear transformation, if we replace the number 3 with the number 4, since a correspondence between four points is needed to determine an object pose in this case. So in general we see that the potential speedup of indexing increases exponentially with the group size.

Intuitively, the fact that one more point is needed to determine a linear transformation than to determine a rigid transformation suggests that in using a linear transformation we will forfeit the extra speedup that would be produced by using one extra point. That is, we would expect a group of size n with scaled orthographic projection to produce the same speedup as a group of size $n+1$ with a linear transformation.

We now perform some experiments to determine these actual speedups. We per-

[2]This ends the extract from Clemens and Jacobs.

Num. Pts.	ϵ		Ideal Bound Orth.	Ideal Bound Linear	Analytic Comparison	Table $d = 50$	Table $d = 100$ Matches	Num. Possible
4	5	Ran.	270					60,000
		Real	46					60,000
5	3	Ran.		$10,000 <$	78.1	23.1	42.4	10,000
	5	Ran.	$12,500 <$[1]	$1,110$[2]	39.7[2]	16.2[2]	26[2]	$12,500$[1] $10,000$[2]
		Real	2,500					10,000
	7	Ran.		1,110	26.2	12.4	18.0	10,000
	10	Ran.		385	17.5	8.88	12.5	10,000
6	3	Ran.		$10,000 <$	1,250	357	178	10,000
	5	Ran.		$10,000 <$	526	189	303	10,000
		Real					244	7
	7	Ran.		$10,000 <$	233	106	156	10,000
	10	Ran.		$10,000 <$	122	60	85	10,000
7	3	Ran.		$10,000 <$	3,330	1,670	3,330	10,000
	5	Ran.		$10,000 <$	1,110	769	1,000	10,000
		Real					3,470	5
	7	Ran.		$10,000 <$	5,000	714	1,430	10,000
	10	Ran.		$10,000 <$	385	222	294	10,000

Table 4.2: This table shows the results of tests on the effectiveness of indexing. Each column shows the speedup produced by indexing under various circumstances, where speedup is defined to be the total number of possible matches between image and model groups, divided by the number of matches produced by that type of indexing. Column one gives the number of points per group in the experiment. Column two gives the amount of error allowed in matching. Column three indicates whether the model and image were randomly generated, or came from real objects and images. The "Ideal" methods indicate that we explicitly compared each group of image and model points, used a least-squares method to optimally align them, and then determined whether this matched the points to within ϵ. The fourth column shows this for a scaled orthographic projection, the fifth for a linear transformation. Column six shows the result of analytically comparing the lines that represent a model's images to the rectanguloids that represent an image with bounded error. Columns seven and eight show the results of comparing these objects via a lookup table, where each dimension of the table is divided into d discrete ranges. Column nine shows the number of possible matches between image and model groups for that set of experiments. When different values in the same row were based on different numbers of matches, we use footnotes to identify which values belong together.

form experiments with randomly generated models and images in which we excluded particularly unstable images, and also with real models and images in which image and model groups were formed by hand. In table 4.2 we compare speedups that were derived in several different ways. We first show experiments performed by Clemens and Jacobs to bound the maximum possible speedup for scaled orthographic projection. In this experiment, instead of finding models by indexing, we matched each group of image points to permutations of each model group, and tested each match to determine if the model could appear like the image. Newton's method was used to find the model pose that minimized the distance from the image points to the projected model points (see Lowe[73] for a discussion of the application of Newton's method to this problem). If in this pose, each model point projects to within error bounds of each image point, then a correct indexing system would have to produce this match. This allowed us to determine a lower bound on the number of correct false positive matches, that is, matches that are geometrically consistent, although the matched model points did not actually produce the image points. We did the same thing with the linear transformation, using the method described above to find a least squares fit between image and model points. In the next experiment we analytically compared the affine lines that describe a model with the rectanguloids that describe an image. This is just like our table lookup, except we avoid the effects of discretization by explicitly comparing each line and rectanguloid to see if they are compatible. Finally, we compared table lookup with different values of d.

We also need to check that our indexing system finds the correct matches. The mathematics guarantees that we will find the right answers if our assumptions about error are met, but we need to check that these assumptions hold in real images. In chapter 7 we will present experiments with the entire recognition system, but we have also tested the indexing system using automatically located point features that we grouped by hand. Out of fourteen such groups, two produced rectanguloids outside the bounds of our lookup table, but the other twelve groups were in each case matched to the correct model groups, allowing for five pixels of error when indexing, and using a table in which $d = 100$. Figure 4.6 shows two groups of image features, the groups of model features to which the indexing system matched them, and the resulting hypotheses about the location of the object. Table 4.2 also shows the speedups achieved with these groups.

There are several conclusions that we can draw from these experiments. Most importantly, they show that our indexing system can produce significant reductions in search, especially when we use groups of seven or more points. This demonstrates the tremendous potential of indexing, provided that we can produce a reasonably small number of candidate groups in both the model and the image without sacrificing reliability. We also see the speedups that we give up by making various approximations. It does appear that using a linear transformation is giving up at most the constraint

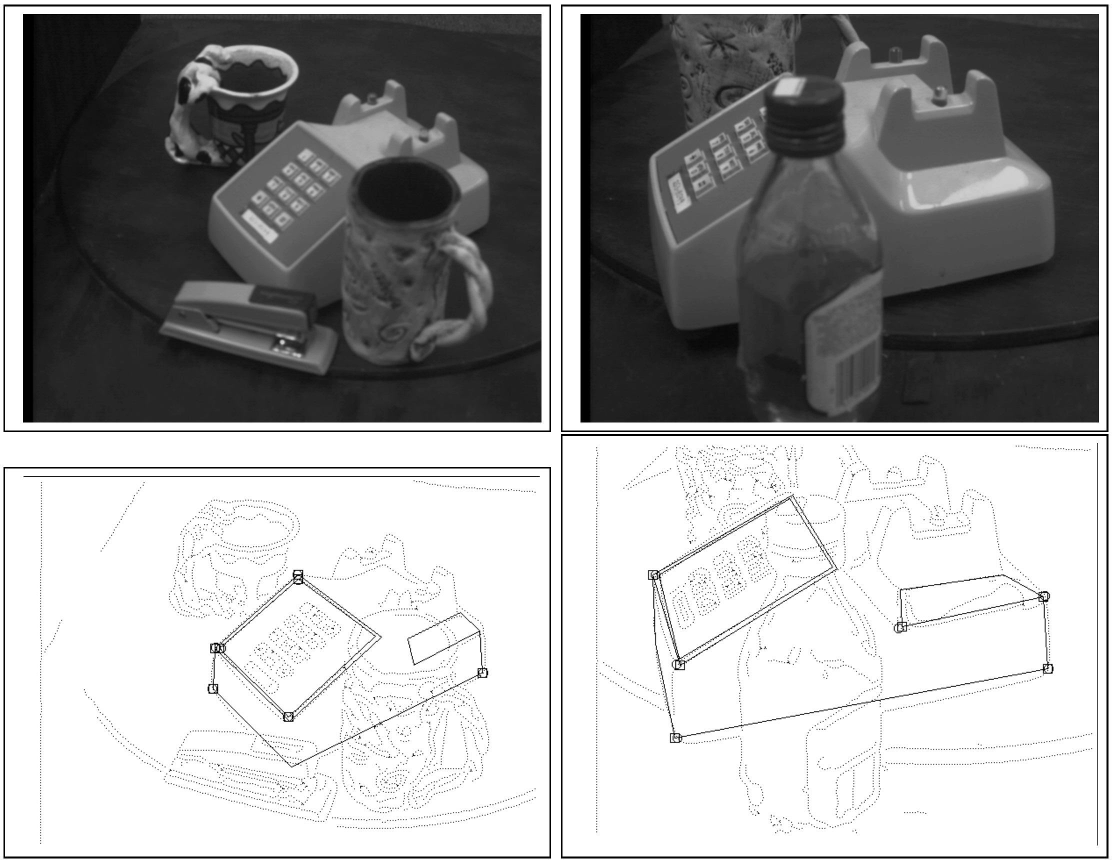


Figure 4.6: On the top are two scenes containing the phone. Underneath each scene is a hypothetical projection of the model considered by the recognition system. Both are correct. In the hypotheses, edges are shown as dotted lines, projected line segments appear as lines, circles represent the image corners in the match, and squares show the location of the projected model corners.

Num. Pts.	ϵ		Ideal Bound Linear	Analytic Comparison	Table $d = 50$	Table $d = 100$	Table $d = 200$	Table $d = 400$	Num. Possible Matches
5	3	Random	$2,500 <$	52.1	14.5	27.2	37.9	43.9	2,500
	5	Random	357	28.4	13.5	19.5	23.8	26.6	2,500
	7	Random	833	21.7	9.8	14.5	17.7	20.2	2,500
6	3	Random	$2,500 <$	833	625	833	833	833	2,500
	5	Random	$2,500 <$	278	132	208	227	227	2,500
	7	Random	$2,500 <$	167	80.6	119	147	156	2,500

Table 4.3: This table shows further results of tests on the effectiveness of indexing, similar to those shown in table 4.2.

that is available in one image point; our least squares bound on the speedups of linear transformation indexing for five points provides a speedup that is a bit larger than the speedup we get for four points with scaled orthographic projection. We also see that bounding error with a rectanguloid results in a significant loss of power in the system, and we would expect this loss to grow with the size of the group. So if we are concerned with increasing the speedups provided by indexing, it might be useful to attempt to use a tighter bound on the error regions.

Our results do not show much of a loss in speedup due to table discretization. It is time-consuming to run experiments with a fine discretization, because many table cells must be accessed in lookup in those cases. However, table 4.3 shows some additional results. In these experiments, the same models and images were compared with $d = 50, 100, 200, 400$. We can see that with $d = 400$ there is almost as good performance with the lookup table as with the analytic matching. But given the run-time and space costs of a higher value for d, it seems reasonable to choose $d = 50$ or $d = 100$.

The significance of our results will depend on how we intend to use indexing. If we expect that grouping can provide a small number of groups with many point features, then we do not need to be too concerned with increasing the speedups provided by simple hash table lookup. If we want to try to take advantage of groups with only four, five or six points for indexing, we can see the importance of taking care with issues such as the projection model used, table discretization, and error. This is pointed out at greater length in Grimson, Huttenlocher, and Jacobs[49]. This lesson may also be relevant to invariant based indexing systems that use projective transformations and perform indexing with the smallest sized groups possible, such as Weiss[111], Forsyth[44], and Rothwell[92].

4.5 Comparison to Other Indexing Methods

Previous systems have also performed feature matching using hash table lookup. This has been done for the recognition of 2-D objects in 2-D images by Wallace[107], Schwartz and Sharir[94], Kalvin et al.[63], Jacobs[60], and Breuel[17], and for the recognition of 3-D objects from 3-D data by Schwartz and Sharir[94] and by Stein and Medioni[98]. However, in these domains, invariant descriptions of the models are available, and so the issues involved in indexing are very different. In this thesis we have focused on the problems of representing a 3-D model's 2-D images, and in this chapter, on the problem of accounting for the effects of error in this domain.

There has been past work in this domain, which is more directly relevant to our current work. Previous authors have noticed that one could represent all of a model's images under scaled orthographic projection by sampling the viewing sphere, and representing in a lookup table each image that the model produces from each point on the viewing sphere. By representing these images in a way that is invariant under scale changes and 2-D translation and rotation all possible images of the model are represented. For example, if a group of image features includes two points, then one can assume that these points have coordinates $(0, 0)$ and $(1, 0)$, and then describe all remaining features in terms of the coordinate system that this gives us. With such a representation, one automatically factors out the effects of four degrees of freedom in the viewing transformation, and need only be concerned with the two degrees of freedom captured in the viewing direction. The set of all images produced by different viewing directions will therefore form a 2-D manifold. By sampling the set of possible viewing directions, one is therefore implicitly sampling the 2-D manifold of images that a model can produce.

Thompson and Mundy[100] use this approach to represent model groups consisting of pairs of vertices. For each pair of 3-D vertices in a model, they sample the set of possible images that the vertices may produce, and store these images in a lookup table, along with the viewing direction. Then, given a pair of image vertices at run time, table lookup is used to determine the viewpoint from which each pair of model vertices might produce a particular pair of image vertices. Thompson and Mundy therefore use lookup primarily to quickly determine the viewpoint implied by a match between the model and the image, not to select valid matches.

Lamdan and Wolfson[71] similarly describe a system that samples the viewing sphere and then creates a separate model for each view. Again, this implicitly samples the 2-D set of images that a 3-D model can produce. Then, a 2D indexing scheme based on invariants is used.

Breuel[17] also proposes an indexing scheme based on sampling the viewing sphere. Breuel's system uses vertex features, making use of the angles of the lines that form the vertices. In this work, the potential effect of changes of viewpoint on the angle of a vertex is determined. This is used to bound the number of different viewpoints needed to find all views of the model that will be represented in different table buckets. Therefore, the number of table entries needed to describe the images that a group of vertices may produce can be bounded.

Clemens and Jacobs[32] also implement an indexing system based on sampling the viewing sphere in order to test the potential speedups provided by indexing. This system represents images of point features in a lookup table.

These approaches provide some of the inspiration for our method, in which we represent all images of a model created from all points on the viewing sphere, using a representation that is invariant under affine transformations. One of the main advantages of our approach, as a method of hash table lookup is that we are able to represent a model's images with two 1-D lines, while previous approaches determined the models' potential images by sampling the viewing sphere, and implicitly were representing a 2-D manifold of images corresponding to the 2-D surface of the viewing sphere[3].

We can see some of the advantages of our approach from some of the reported results. Thompson and Mundy's system required 2,500 table entries to represent the images of a pair of model vertices. Clemens and Jacobs' system required over 5,000 table entries to represent a group of five model points. Lamdan and Wolfson report sampling the viewing sphere at a rate of every ten degrees. The space required by our system is one or two orders of magnitude less. This significantly increases the number of groups of model features that we can hope to represent in a lookup table. Moreover, our approach extends gracefully to handle larger groups of model features, with modest additional space requirements. It is not clear how growth in group size will affect the space required by other approaches; as groups grow larger, more images must be sampled because the chances are greater that some of the model features will be significantly effected by small changes in viewpoint.

Also, our approach allows us to analytically construct a lookup table that is guaranteed to be correct. Most of the systems described above uniformly sampled the viewing sphere, with no guarantees about how much inaccuracy this might introduce into the lookup table. Breuel was able to bound this inaccuracy in the case where only the angle between two lines was used as a model feature.

Finally, we have presented a method of accounting for image error at lookup time that guarantees that we will find all matches that fit a bounded error assumption. Most of the above systems rely on the discretization of the index space to account for error.

On the other hand, we have paid for these advantages by using a more general projection transformation that introduces two more degrees of freedom into the pro-

[3] Breuel[18] mentioned explicitly that a 2-D manifold is represented by this method.

jection. This may be a liability if we do not require the added capability of recognizing photographs of objects viewed from new positions.

We should also stress that while there are considerable practical advantages to our approach to indexing, the greatest difference between our approach and previous ones is more conceptual. We have rephrased indexing as a geometric matching problem between simple objects that we can analytically compute.

4.6 Conclusions

Aside from some smaller implementation points, there are two main conclusions of this chapter. First, we show that we can carefully understand the effects of error on point feature matching. We have shown the precise effects of error on matching four planar points with alignment or geometric hashing, and then shown how this can also bound the effects of matching 3-D objects under linear transformations. These results therefore have relevance to the implementation and analysis of a wide range of approaches to recognition.

We can also see that understanding the effects of error matters. In some indexing systems (Lamdan, Schwartz and Wolfson[70], Thompson and Mundy[100], Lamdan and Wolfson[71], Forsyth et al.[44]) ad-hoc methods are used to handle error, such as counting on the use of discrete cells to match images and models that are a little different. In the case of point features we can see how inaccurate that can be. Using discretization to handle error means effectively putting a fairly arbitrary rectangle about the images in index space, and matching them to all models that map to somewhere in that rectangle. And the rectangles are all the same size. In the case of four points, we can see that the true error regions are usually elliptical, and that their size and orientation can vary quite a bit. When there are more than four points, the variation in affine coordinates of different points can also be great. This means that an arbitrary treatment of error is likely to miss many matches, or to be so sloppy as to greatly reduce the effectiveness of indexing.

We also see experimentally that indexing can be of great value when grouping provides us with large collections of image features. But we see that indexing is of quite limited value when small groups of image features are used. It is especially in those cases that a careful treatment of error is needed to squeeze all the power we can from the indexing system.

Chapter 5

Inferring 3-D Structure

5.1 Introduction

In the introduction to this thesis we described two possible approaches to indexing. In the approach that we have pursued so far, we characterize the set of 2-D images that a 3-D model can produce, and then match between 2-D images. A second approach is to derive 3-D information from a 2-D image and perform a comparison in 3-D. The advantage of such an approach is that since the 3-D structure of a model does not depend on the viewpoint, only a single representation of the model is needed. In this chapter we examine the extent to which we might hope to recover 3-D information about an image of point features.

We need not derive explicit 3-D information about the scene to gain the advantages of a 3-D comparison. If we can derive some viewpoint invariant property of the scene from an image, we have implicitly determined something about its 3-D structure, because we have determined something that depends only on this structure, and not on the viewpoint. Therefore, invariants can be viewed as a representation of the 3-D scene information. So when we discuss invariants in this chapter, we are at the same time discussing 3-D scene reconstruction.

In chapter 2 we showed that there are no complete invariant functions for general 3-D objects. Recall that a complete invariant function is a function of a single image that captures all the information about the model that could effect the images that it can produce. This tells us that indexing by recovering 3-D information, implicitly or explicitly, can never make use of all available information, and can never be as complete as indexing by characterizing a model's images.

However, the advantages of performing indexing using 3-D information are potentially great, because of the space savings and conceptual simplicity gained by using only a single 3-D model. So it is worth considering whether we can do any useful indexing in this way. There are several ways in which we might try to get around the results of chapter 2. First, we might consider allowing our invariant functions, which implicitly contain 3-D information, to introduce errors. We first show that allowing false positive errors is of no help. That is, we show that there are no invariant functions at all, even ones which might throw away some of the useful information in the image. Then we consider whether invariant functions may exist if we allow false negative errors, that is, if we allow functions that occasionally match an image to a model that could not have produced it. We show that when small numbers of false negative errors are allowed, there are still no invariant functions. These results tell us that we may not infer 3-D information from point features even if we allow an occasional mistake.

We then consider whether we might find invariant functions for limited libraries of models. We show under what circumstances a particular set of models might give rise to an invariant. Finally, we turn to non-accidental properties. These are individual properties of an image that are unlikely to occur by chance. It has been thought that these special properties offer a method of inferring 3-D structure when they occur. We show that in the case of point features, however, these properties are of limited value.

A number of researchers have considered methods of inferring 3-D structure from a 2-D image that contains richer features than the simple points that we analyze. Of course there is an extensive literature on stereo and motion understanding, situations where more than one image of the scene is available. There has also been work on deriving 3-D information from the shading or texture of a single image. And there has been extensive work on determining 3-D structure from line drawings. These drawings usually contain either connected line segments or closed curves, the idealized output of an edge detector that would locate all scene discontinuities without gaps or errors. Some of this work on line drawings has derived qualitative 3-D information from images, for example by parsing lines into objects, or determining the meaning of each line (is a line an occluding contour or an orientation discontinuity? does a line indicate a convexity or a concavity in 3-D?). Early work along these lines was done by Guzman[51], Huffman[55], Clowes[33], and Waltz[108]. More recent work is described in Koenderink and Van Doorn[66], Malik[75] and Nalwa[86]. We will discuss in more detail work that makes inferences using non-accidental properties, including that of Binford[11], Kanade[64], and Lowe[73]. And while some of this work makes definite inferences from an image, other work, realizing that many different 3-D scenes are compatible with certain images, attempts to find methods of preferring some interpretations over others. This is done, for example, in Brady and Yuille[16], Marill[80], and Sinha[97].

When so much work has been expended in examining the problem of scene reconstruction in a more complex image, we must explain why we consider this problem for images of just point features. One reason is that much of the above work assumes idealized input, especially in assuming curves without gaps. When working with broken curves, it may be more useful to see what can be inferred from more isolated features, as we do. Second, the above work has achieved only partial success. It may be useful to more thoroughly examine a simpler kind of image. And some of our results may be extended to more complex domains. In particular, our analysis of non-accidental properties is easily applied to properties of lines such as parallelism and symmetry.

The primary conclusions of our work is that there are strong limitations on the 3-D inferences that we can make from a single image of point features. These results strengthen the sense that the representations of a model's images derived in chapter 2 are indeed optimal. Whenever we show that we cannot derive viewpoint-invariant information about a model, we have shown that it is not possible to represent that model with a single point in some image space, which would capture that viewpoint invariant data. Our results also provide greater insight into non-accidental properties as an approach to recovering 3-D information about a scene. It remains to be seen, however, to what extent stronger 3-D inferences can be made when we have richer image information available.

5.2 Scaled Orthographic and Perspective Projection

5.2.1 There Are No Invariants

In chapter 2 we showed that there are no non-trivial, complete invariant functions of images. In this section, we show that there are no non-trivial invariant functions at all. This is equivalent to showing that there is no mapping from images to image space for which every model's manifold is a point unless image space consists of a single point to which all images are mapped. We first present a proof from Clemens and Jacobs[32] which applies only to scaled orthographic projection, and then a proof discovered by Burns, Weiss and Riseman[24] and by Moses and Ullman[84] which applies to all projection models. The results of Clemens and Jacobs[31] and Burns, Weiss and Riseman[22] appeared simultaneously. The work of Moses and Ullman[83] appeared later, but was performed independently.

Following Clemens and Jacobs[32] we proceed by first considering the case in which models have four points. If an invariant function, f, exists, we can use it to partition this set of models into equivalence classes. If f is an invariant function, then it assigns the same value to all images of a single model. We may therefore speak unambiguously of f assigning a value to the model itself. We then say that two models are equivalent if and only if f assigns them the same value. Clearly two models will be equivalent if they produce a common image. If f partitions the models into a trivial equivalence class where all models are equivalent, this means that f is a trivial invariant function which assigns the same value to all images.

We now show that any two four-point models, m_1 and m_2 are equivalent. We proceed by showing that they are each equivalent to a third model. Let $m_{(1,1)}$ denote the planar model with affine coordinates $(1,1)$. Recall that the affine coordinates of point in a set of planar points are its coordinates when the first three planar points are used as a basis, and that these coordinates are invariant under affine transformations. Theorem 2.3 tells us that the planar model $m_{(1,1)}$ can produce any image of four points that has affine coordinates $(1,1)$. We know from lemma 2.5 that m_1 can produce an image with affine coordinates $(1,1)$. Therefore this image will be produced by both $m_{(1,1)}$ and m_1, and the two models are equivalent. Similarly, $m_{(1,1)}$ is equivalent to m_2, because, again by lemma 2.5, it may produce an image with affine coordinates $(1,1)$. So m_1 and m_2 are equivalent.

Suppose now that models have more than four points. Let m_1 be a model with n points. We can use a similar technique to show that m_1 is equivalent to any planar model with n points. Pick some such planar model, with affine coordinates $(\alpha_4, \beta_4, ...\alpha_n, \beta_n)$. We'll call this model p_n. Then we know that there is some viewpoint from which m_1 produces an image with affine coordinates α_4, β_4, and some other affine coordinates we call $\alpha'_5, \beta'_5, ...\alpha'_n, \beta'_n$. Call a planar model with these affine coordinates p_4. m_1 is equivalent to p_4 because because m_1 can produce an image with the same affine coordinates as p_4, and p_4 can produce all such images, by theorem 2.3. We now create another model, m_5, which is the same as p_4 except in its fifth point, which is not coplanar with the others. m_5 is equivalent to p_4 because we can view it to produce affine coordinates α'_5, β'_5 in its fourth point, while all its other affine coordinates always match those of p_4. Similarly, m_5 is also equivalent to a planar model p_5 that has affine coordinates $(\alpha_4, \beta_4, \alpha_5, \beta_5, \alpha'_6, \beta'_6, ...\alpha'_n, \beta'_n)$. So m_1 is equivalent to p_4 which is equivalent to m_4 which is equivalent to p_5. We can continue this process until we have a chain of equivalent models which connects m_1 to p_n. Then, since we have shown that any 3-D model is equivalent to any planar model, we may conclude that any two 3-D models are equivalent to each other. Figure 5.1 illustrates this argument. This means that any invariant function that applies to all possible models must be trivial. By the way, the above proof assumes that points of m_1 are not coplanar, but removing this assumption introduces no special difficulty.

We now present the related proofs of Burns, Weiss and Riseman[24] and Moses and Ullman[84]. because they provide a somewhat different way of thinking about the problem than the one described above, and because they are more general, applying to perspective projection as well as scaled orthographic projection. As the two proofs are similar, we describe them together here.

Given two models of n points each, m_1 and m_2, we construct a chain of intermediate models, m'_1, ... m'_n such that each adjacent pair of models in the sequence

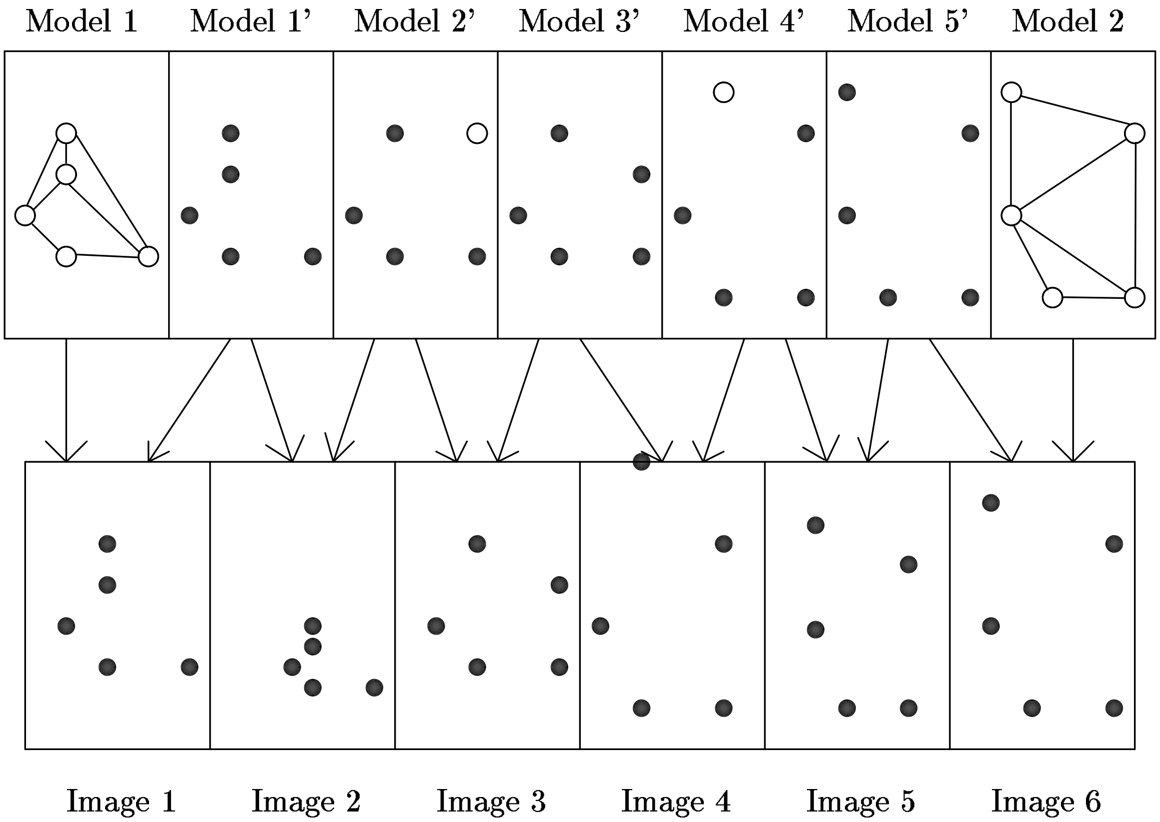


Figure 5.1: The two models on the left and right are connected by a set of intermediate models. We form an arbitrary planar model, model $3'$, and show that each model is equivalent to it. We begin by taking an image (image 1) of model 1 whose fourth point has the same affine coordinates as the fourth point of model $3'$. Then we create model $1'$, a planar model identical to image 1. Model $2'$ is identical to model $1'$ except for its fifth point, which is any point not coplanar with the others. Then both models $1'$ and $2'$ can create image 2, which is the image of model $2'$ that has the same affine coordinates as model $1'$, since model $2'$'s fifth point may appear with any affine coordinates. For this reason, model $2'$ can also create an image in common with model $3'$. We connect model 2 to model $3'$ in a similar fashion.

$m_1, m'_1, \ldots\ m'_n, m_2$ can produce a common image. Then all the models are equivalent, and any invariant function is trivial.

To construct this sequence we first show a simple lemma.

Lemma 5.1 *If two models are identical except for a single point, then they produce a common image from some viewpoint.*

Proof: To find this viewpoint, we just orient the two models so that all their common points are in the same place. A line connecting their two different points then describes a possible viewpoint. When seen from that viewpoint, the points that are different in the two models line up in the same place in the image. All the other model points coincide in 3-D space, so they appear in the same places in the image regardless of viewpoint. This proof applies equally to orthographic or perspective projection, because in either case, given a line through two points, we can view the points along that line so that they create the same image point. □

Now we can create a sequence of intermediate models easily. Let m'_i be identical to m_1 in the first $n - i$ points, and identical to m_2 in the last i points. Then each model in the sequence differs from the previous one in only a single point, because m'_1 differs from m_1 in only the last point, m_i differs from m_{i-1} in only the i'th point, and m'_n is identical to m_2.

These results tell us that allowing false positives in our representation will not allow us to produce invariants, for in these proofs we allow a model to match an image that it could not produce. So we cannot create an image space in which each model's manifold is 0-D. We now ask what is the lowest-dimensional representation possible when we allow false positive errors? Since a model's images are continuously connected, then any continuous mapping from images to image space must map these images to a continuously connected manifold of image space. If this manifold is not a single point, then it must be at least one-dimensional. We have already seen such a one-dimensional representation in chapter 2. If we just consider the set of α coordinates of the images that a model can produce, we know that each model corresponds to a 1-D line in this α space. Before we spoke of decomposing an image space into an α subspace and a β subspace, but we may also consider α space as the entire image space. To do this introduces false positives because two models might produce images with the same set of α coordinates but with different β coordinates. But we have a non-trivial representation in which each model's manifold is 1-D, and this is the lowest dimensional representation possible when we allow false positives.

5.2.2 No Invariants with Minimal Errors

The purpose of this section is to strengthen the results of the previous section. The proofs that there are no invariants depend on the assumption that any invariant applies to all models, and produces no false negative errors. This is a compelling result because there has been a great deal of work done in mathematics on invariants that meet these conditions. We now know that this work will not apply directly to the problem of recognizing a 3-D object from a single 2-D image. However, this result does not prove that there can be no useful functions that are in some sense almost invariant. To address this question we ask what happens when we allow all kinds of errors, but such a small number of errors that they are of no practical significance? We alert the reader to the fact that this section my be somewhat slow going, and that it can be skimmed or skipped without loss of continuity.

We ask two questions: will accepting some errors allow us to find an invariant function? and will it allow us to represent each model at a single point in some image space? When no errors are allowed, we showed that these two questions were identical. This is no longer obvious when we allow errors, so we treat the two questions separately. We must also explain what we mean by a small number of errors. So we begin by defining these two questions more carefully. Then we show that when we allow errors, the existence of 0-D representations of models still implies the existence of an invariant function. Then we show that even with a small number of errors allowed, there is no such invariant function, and hence no such 0-D representation.

First of all, we make the following definitions.

Definition 5.1

- *As before, let $\mathcal{M}$ be the set of all models, let $\mathcal{I}$ be the set of all images, and let f be a function from images to an image space, $\mathcal{S}$. And let $\mathcal{T}$ be the set of all legal projections from $\mathcal{M}$ to $\mathcal{I}$. Our proofs will apply equally to perspective or scaled orthographic projections.*

- *Let g be a function from $\mathcal{M}$ to $\mathcal{S}$ that we will specify later.*

- *Let $gf(i) = \{m | g(m) = f(i)\}$. That is, $gf(i)$ is the set of all models such that g maps these models to the same point in $\mathcal{S}$ to which f maps i.*

- *Let $p(i) = \{m | \exists t \in \mathcal{T}$ such that $t(m) = i\}$. That is, $p(i)$ is the set of all models that can produce image i.*

- *Let $h(m)$ be a function from $\mathcal{M}$ to all subsets of $\mathcal{T}$. That is, the function h describes a particular set of views for each model.*

- *Let $\mathcal{TM}$ stand for the set of all pairs of transforms and models. That is, (t, m) is a typical member of $\mathcal{TM}$. Let $\mathcal{TM}_i$ indicate the set of all $(t, m) \in \mathcal{TM}$ such that $t(m) = i$. Therefore, $\mathcal{TM} = \bigcup_{i \in \mathcal{I}} \mathcal{TM}_i$.*

We will ask whether allowing an infinitesimal number of errors might improve our indexing system. This notion can be made more formal using basic concepts from measure theory, but it should be briefer and clearer to present these proofs using simple intuitive notions. For example, we might allow a function, instead of being invariant for all models, to be invariant for all but an infinitesimal number of models. If $\mathcal{M}'$ is a subset of the set of all models, $\mathcal{M}$, we will say that $\mathcal{M}'$ is *infinitesimal* with respect to $\mathcal{M}$ if, when selecting a model, m, from $\mathcal{M}$ at random, there is a 0 probability that $m \in \mathcal{M}'$. This definition implies that we have some probability distribution for the set of models in $\mathcal{M}$. We will assume that the points of each model are chosen independently and uniformly from a cube of fixed size. We can assume similar distributions for images and transformations that allow us to define infinitesimal subsets of them. As an example, let $\mathcal{M}$ be the set of models with 5 points, and let $\mathcal{M}'$ be the set of planar models in $\mathcal{M}$. Then $\mathcal{M}'$ is infinitesimal in $\mathcal{M}$. On the other hand, if $\mathcal{M}'$ contains all but an infinitesimal set of $\mathcal{M}$'s members, we will say that $\mathcal{M}'$ is *almost all of* $\mathcal{M}$.

In practice, our choice of a probability distribution on the models, images and transformations is not important to the proofs that follow. We only rely on the following properties of the distribution that we have chosen.

1. If $\mathcal{I}'$ is almost all of $\mathcal{I}$, then $\mathcal{M}'$ is almost all of $\mathcal{M}$, if we define $\mathcal{M}' = \{m \in \mathcal{M} | \exists t \in \mathcal{T} \text{ such that } t(m) \in \mathcal{I}'\}$. That is, if we have a set of almost all the images, then the set of models that could produce these images is almost all the models.

2. Similarly, if $\mathcal{M}'$ is almost all of $\mathcal{M}$, and h is defined so that $\forall m \in \mathcal{M}', h(m)$ is almost all of $\mathcal{T}$, then $\mathcal{I} = \{i | \exists m \in \mathcal{M}', t \in h(m) \text{ such that } i = t(m)\}$ is almost all of $\mathcal{I}$. That is, if we have a set of almost all the models, and for each model we consider almost all the viewpoints of that model, we will produce almost all the images.

3. If $\mathcal{M}'$ is almost all of $\mathcal{M}$ and $\mathcal{T}'$ is almost all of $\mathcal{T}$ then $\mathcal{T}'\mathcal{M}'$ is almost all of $\mathcal{T}\mathcal{M}$.

4. Suppose $\mathcal{T}\mathcal{M}'$ is non-infinitesimal in $\mathcal{T}\mathcal{M}$, and let $\mathcal{I}'$ be the set of all images such that $i \in \mathcal{I}'$ if and only if $\mathcal{T}\mathcal{M}' \cap \mathcal{T}\mathcal{M}_i$ is non-infinitesimal in $\mathcal{T}\mathcal{M}_i$. Then $\mathcal{I}'$ is non-infinitesimal in $\mathcal{I}$. This is really fairly simple, in spite of the obscure definitions. It just means that if on the one hand there is a non-zero probability that a randomly chosen model-transformation pair is in $\mathcal{T}\mathcal{M}'$, then if we randomly select an image there will be a non-zero chance that a randomly chosen model-transformation pair that can produce that image will also be in $\mathcal{T}\mathcal{M}'$.

All of these conditions essentially just depend on the fact that the probability distribution we have chosen assigns a 0 probability to selecting what intuitively seems like an infinitesimal set of images or models or transformations. For example, we do not want any one image to have a finite probability of occurring.

We will now define our two questions for the case where we allow errors to occur with only infinitesimal probability. If a result leads to an indexing method in which the set of situations that gives rise to an error is infinitesimal, this means that in practice where we have only a finite number of models and images, such errors will never occur.

One might ask if this condition is too strong. We might be satisfied with a system in which errors occur only rarely, instead of never. However, there are two reasons for examining the strong requirements described here. First, it is easier to show negative results about these requirements than about looser requirements, while these results still help to strengthen our understanding of the difficulty of fulfilling even looser requirements. Secondly, we are still considering the idealized case where there is no sensing error. One may have the intuition that any approach that allows for real errors in this case is likely to produce a great many errors when we account for sensing uncertainties.

Question 5.1 *Does there exists some $\mathcal{X} \subset \mathcal{I}$, where $\mathcal{X}$ is infinitesimal in $\mathcal{I}$, and some g and some f, such that: $\forall i \in \mathcal{I}, i \notin \mathcal{X}$, the following two conditions hold:*

1. *$gf(i)$ is infinitesimal in $\mathcal{M}$*
2. *$gf(i) \cap p(i)$ is nearly all of $p(i)$?*

This question asks whether we can make one entry in image space for each model without having problems on a greater than infinitesimal set of images. The function g describes these entries by mapping each model to a point in image space, while f maps the images to image space. $gf(i)$ tells us which models are mapped to the same place in image space as is the image i. There are two ways we can have problems with an image, given as the two conditions above. First, if the image is matched to all the right models, but this matching is unhelpful because it produces too many false positive matches. In the absence of error an image is geometrically compatible with an infinitesimal subset of models; that is, the probability that an arbitrary model could produce an arbitrary image is zero (see Clemens and Jacobs[32] for further discussion of this). So condition one states that our indexing system should reflect this by matching an image to an infinitesimal set of models. Second, a problem arises if the image is not matched via image space to almost all of the models that could have produced the image. This seems like a reasonable way of defining a map to image space that introduces few errors. If such an f and g existed, we could use them for indexing, and for a randomly selected image of a randomly selected model, with probability 1 our indexing system would match the image to the right model, and to no other models in any finite, randomly chosen data base of models.

We now define question two:

Question 5.2 *Does $\exists Y \subset \mathcal{M}$, where Y is infinitesimal in $\mathcal{M}$, and $\exists f, h$ such that the following two conditions hold:*

1. *$\forall m \in \mathcal{M}, m \notin Y$ then $h(m)$ is almost all of $\mathcal{T}$, and $\forall t, t' \in h(m)$ then $f(t(m)) = f(t'(m))$*
2. *f is non-trivial in the following sense. There exist I' and I'', two non-infinitesimal subsets of $\mathcal{I}$, such that, $\forall i' \in I'$ and $\forall i'' \in I''$, $f(i') \neq f(i'')$?*

This question asks whether there is a "nearly invariant" function, f. So we say that the function might not be invariant for an infinitesimal subset of models. Then the first condition above says that for each remaining model, the function must be invariant for almost all images of the model. That is, when we view the model from almost all possible viewpoints, the function doesn't vary over all the images that are produced. The second condition requires that the function be non-trivial in the sense that it cannot be constant over almost all images.

We prove that both of these questions must be answered negatively. To do this, we first show that if question 5.1 is true, question 5.2 is true. Then we show that question 5.2 is false.

We begin by making a couple of definitions based on the premises of question 5.1. First, we will say that f is either *correct* or *incorrect* for a particular element of $\mathcal{TM}$. We say f is correct for $(t, m) \in \mathcal{TM}$ if $f(t(m)) = g(m)$, and incorrect otherwise. That is, given f and g, we can tell for a particular model, and a particular transformation, whether f and g will map the model and its image to the same place in image space, resulting in correct indexing. We also define $\mathcal{TM}'$ to be the subset of $\mathcal{TM}$ for which f is incorrect. We define $\mathcal{TM}'_i$ to be the subset of $\mathcal{TM}_i$ for which f is incorrect.

Given the assumption that X, f, and g satisfy the conditions of question 5.1, we show that we can satisfy the conditions of question 5.2. Let h be defined so that $h(m)$ is the set of transformations for which, together with m, f is correct. That is, $t \in h(m)$ if and only if $f(t(m)) = g(m)$. Also, let f in question 5.2 just be the same f that worked in question 5.1. Finally, let Y be the set of all models for which f is not constant over almost all transformations. That is, $m \in Y$ if and only if $h(m)$ is not almost all of $\mathcal{T}$. Then we must show three things to establish question 5.2.

First, we must show that for any $m \notin Y$, $h(m)$ is almost all of $\mathcal{T}$. This follows immediately from the definition of Y. Second, that f is non-trivial, in the sense given in question 5.2. If this were not true, f would be constant over almost all images, and, for any of these images, $gf(i)$ would have to include almost all models.

Third, we must show that Y is infinitesimal. If Y were not infinitesimal, this would mean that for a non-infinitesimal set of models, there are a non-infinitesimal set of transformations, such that f is incorrect on these model-transformation pairs. This means that $\mathcal{TM}'$, the set of incorrect model-transformation pairs would be non-infinitesimal as well. Since $\mathcal{TM}'$ is the union of all $\mathcal{TM}'_i$, if $\mathcal{TM}'$ is non-infinitesimal in $\mathcal{TM}$, then there must be a non-infinitesimal set of $\mathcal{TM}'_i$ that are all non-infinitesimal in the corresponding $\mathcal{TM}_i$. Therefore, since X is infinitesimal, there must be some image, $i \notin X$ such that $\mathcal{TM}'_i$ is not infinitesimal in $\mathcal{TM}_i$. That is, there must be some image beyond the ones we expect wrong answers from, for which we still get wrong answers on a non-infinitesimal subset of the set of model-transformation pairs that produce that image. This will mean that we will not match the image to almost all of the models that could produce the image.

We now prove that question 5.2 is false.

We make two definitions. If two models have the same number of points, and are identical except in one point, we will call them *neighbors*. A *neighborhood* is the set of points that all differ from each other in only one point. That is, if we fix all but one point in a model, and let the last point appear anywhere, this defines a neighborhood. It should be clear that a model with n points is in n different neighborhoods.

Our strategy now is to extend the set of excluded models, Y, while keeping it infinitesimal. We will extend it to the set Y', then we will extend Y' to be Y''. For any model not in Y'', we know that f is constant as the model is viewed from almost any viewpoint. We will then show that f has the same constant value for any two models not in Y'', which means that f is constant over almost all images.

Let N stand for the set of all neighborhoods. Let N' stand for the set of all neighborhoods in which a non-infinitesimal portion of the models are in Y. That is, if $n' \in N'$ this means that $n' \cap Y$ is non-infinitesimal in n'. We now want to show that the set of models in all the neighborhoods in N' is infinitesimal in M. Each model in Y can only appear in n different neighborhoods. Since each neighborhood contains an infinitesimal portion of the models, it is possible for some neighborhoods to be dominated by models in Y. But overall, there can only be an infinitesimal set of neighborhoods, out of the set of all neighborhoods, for which a non-infinitesimal portion of the models come from Y. To make this more concrete, suppose we randomly select a neighborhood (by randomly selecting a model with $n-1$ points), and then randomly select a model in that neighborhood (by randomly selecting the n'th point). We know that the probability of this model belonging to Y is zero. That means that the probability must be 1 that once we have selected the initial neighborhood, there is 0 probability that we will select a model in that neighborhood that belongs to the set Y. This is just another way of saying that N' is an infinitesimal subset of N, and so all the neighborhoods in N' can together contain only an infinitesimal set of models. We now define Y' to be the set of models that can be found in N'.

We would now like to show for any two models that are not in Y' and that are in the same neighborhood, call the models m_1 and m_2 and the neighborhood n^*, that f has the same value both for the images created when m_1 is viewed with any transformation in $h(m_1)$ or when m_2 is viewed with the transformations of $h(m_2)$. We can not do this by just claiming that the two models have an image in common. The two models differ in one point, so there are only an infinitesimal set of viewpoints from which m_1 produces an image that m_2 can also produce, and it could be that none of these viewpoints belong to $h(m_1)$. However, since $h(m_1)$ contains almost all viewpoints, we know that m_1 will have an image in common with almost all the models in n^*. So one of these models must have an image in common with m_2, unless it happens that the allowable viewpoints of almost all the models in n^* are constructed so that none of them can produce an image in common with m_2. This can happen. But only for an infinitesimal subset of n^*. Therefore, we create the new set, Y'' which contains all models in Y', and all models that belong to a neighborhood in which they do not share an image with almost all of their neighbors. Y'' will still be infinitesimal. And now, if we assume that m_1 and m_2 do not belong to Y'', then we know that they must each share a legal image with almost all their neighbors. In particular there must be a neighbor with which each shares an image.

Finally, if we take two models that do not belong to the same neighborhood, we can create an intermediate sequence of models, in which all the models in the sequence differ by one point, and so they do share a neighborhood. This tells us that f will be constant over all images of all models, m, as long as $m \notin Y''$ and the image is formed by a transformation in $h(m)$. Since f is constant for almost all images of almost all models, it must be constant for almost all images.

We have therefore shown that even if we allow an infinitesimal number of errors to be made, there can be no invariant functions and no indexing performed using 0-D manifolds for each model. Moses and Ullman[84] have addressed this question from a different perspective, and, under a stronger set of assumptions they show that any invariant function must produce a much larger set of mistakes than we have considered. Collectively, this work makes it seem unlikely that we can produce invariant functions by excluding a small set of uninteresting situations from our domain.

5.3 Linear Transformations

We now focus on linear transformations. In chapter 2 we showed that when this transformation is used, the images that a model produces correspond to a plane in affine space, which is decomposed into lines in α and β space. This result will make it much simpler to determine under what circumstances an invariant function exists, because we can tell whether two models produce a common image by seeing whether their planes in affine space intersect. We begin by allowing false positive errors, but no false negative errors. In that case there are no invariants for general models, but we consider whether a less general library of models might produce an invariant function. We show that if there is an invariant function, the set of allowable objects must be very restricted. We then consider the case in which small number of false negative errors can be made.

5.3.1 False Positive Errors

When we consider a linear transformation, and allow only false positive errors, then there can be no invariants for general 3-D objects. Our proof of this for scaled orthographic projection applies equally well to linear transformations. However, we begin by introducing a simpler proof for this case. This proof also illustrates some general principles that we can use to determine, for any specific library of models, whether there is an invariant function.

Previous proofs that there are no invariants relied on connecting any two models with a sequence of intermediate ones which all share an image. We can present a proof that requires only two intermediate models. Suppose model m_1 corresponds to the two lines, a_1 and b_1 in α-space and β-space respectively. Similarly, suppose model m_2 corresponds to a_2 and b_2. Then there are an infinite number of lines that intersect both a_1 and a_2. Choose one of these, a'_1. Choose b'_1 as any line that is parallel to a'_1, and intersects b_1. then there is a model, m'_1 that corresponds to the lines (a'_1, b'_1). m'_1 has an image in common with m_1, since a_1 intersects a'_1, and b_1 intersects b'_1. We may then construct m'_2 and its lines, (a'_2, b'_2), so that b'_2 intersects b'_1 and b_2, and so that a'_2 passes through the point where a'_1 and a_2 intersect. So m'_2 will have an image in common with m'_1 and m_2. This is illustrated in Figure 5.2. Therefore, any invariant function must have the same value for any image of any of the four models.

This proof shows us in general how to tell whether a particular library of objects has a possible invariant function. As Moses and Ullman[84] point out, there will be invariant functions for a particular set of 3-D models if the models can be divided into non-trivial equivalence classes, where two models are equivalent if they have an image in common, or are both equivalent to another model. From our previous work, it becomes easy to form these equivalence classes for a particular set of models, because we can tell that two models produce a common image if their corresponding lines in α-space and in β-space intersect, that is, if their 2-D planes in affine space intersect. Therefore, there will be a non-trivial invariant function if and only if the planes that represent our models in affine space do not form a completely connected set of images.

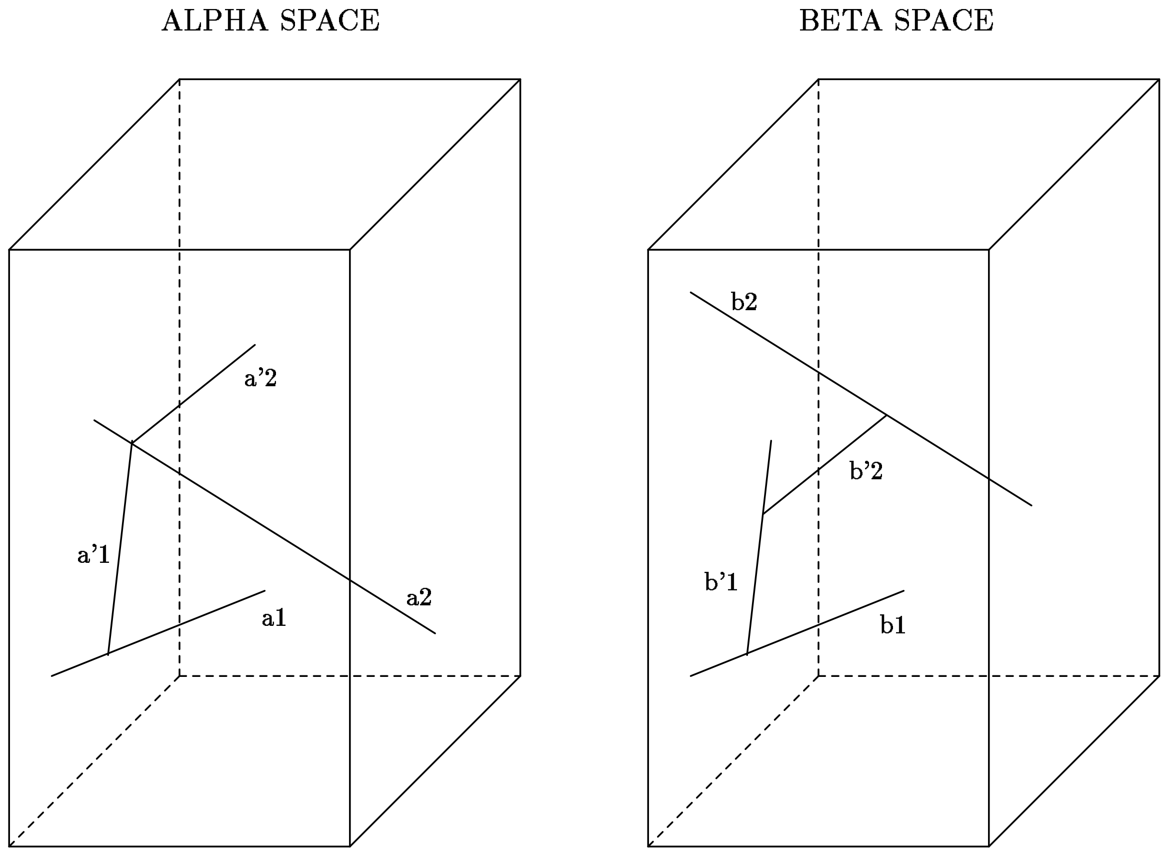


Figure 5.2: Two models, m_1 and m_2 correspond to the pairs of lines (a_1, b_1) and (a_2, b_2) respectively. We construct the models m_1' and m_2', which correspond to the lines (a_1', b_1') and (a_2', b_2') respectively. Whenever the α and β lines of a two models both intersect the models produce a common image. So we can see that the two constructed models link the two original models with a series of common images.

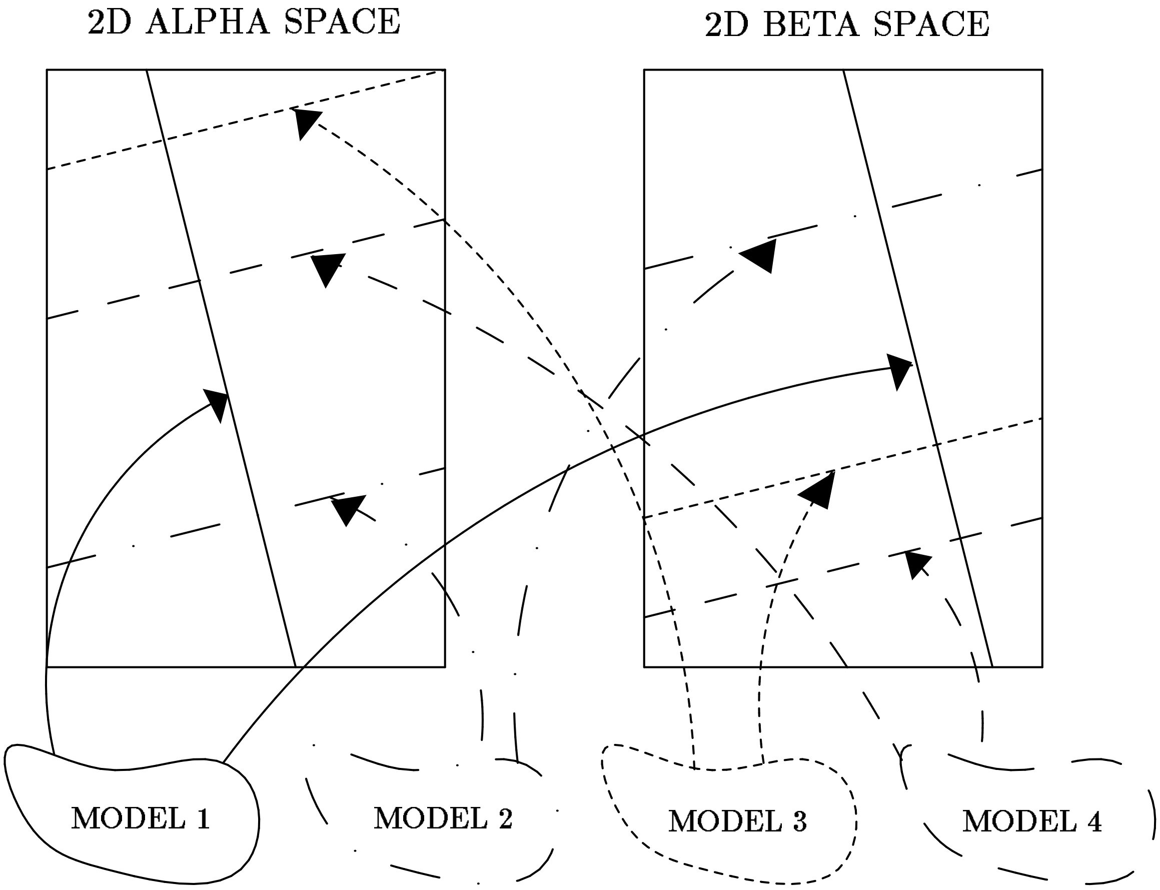


Figure 5.3: This figure illustrates the fact that when models consist of five points, the images they produce are linked into a continuous set unless all the height ratios are the same, and hence all the lines to which they correspond are parallel.

An interesting special case in which to see this is that of models containing five points. This is the smallest group of points that can produce invariants, because in general any four model points can appear as any four image points under a linear transformation. Most systems based on invariants have used the smallest possible model groups, in order to limit the combinatoric or grouping problem involved in considering all possible image groups (this is true of Lamdan, Schwartz and Wolfson[70], Forsyth et al.[44], and Van Gool, Kenpenaers and Oosterlinck [106], for example). A set of model groups of five points will each produce a pair of lines in 2-D α-space and β-space (see figure 5.3). Furthermore, recall that these two lines will have the same slope, which means that two different models will produce lines that are either parallel in both spaces, or that intersect in both spaces. Therefore, an invariant function for groups of five points is possible only when all lines produced by all models are parallel. For example, if one model produces lines not parallel to the others, it will have an image in common with each of them, implying that the invariant function must be constant over all images produced by all models. The lines produced by all models will be parallel only when r_5, (see equation 2.1), is the same for all models, that is, when the ratio of the height above the model plane of the fourth point to the height of the fifth point is always the same.

When models consist of five points, only an infinitesimal subset of the set of all possible models will give rise to an invariant function. This is not true for models with more points. Here we give an example of a function that will be invariant for a greater than infinitesimal set of possible models in the case where models have six points. This function will be defined by an hourglass shaped region of α space. That is, the function will give one value for any image with α coordinates in this region, and a different value for any other image. (Since we are allowing false positive errors, we may consider a function that ignores the β coordinates of the images. Such a function cannot distinguish between two models that produce the same α coordinates but different β coordinates). By an hourglass shaped region, we mean, for example, the section of α-space formed by all lines that intercept the $\alpha_4 = 0$ plane over some disc, and which are within a few degrees of being orthogonal to this plane. Figure 5.4 illustrates this. If we consider the set of possible models that intersect a bounded portion of α-space, then the set of models which correspond to lines completely inside this hourglass region is non-infinitesimal. There is also a non-infinitesimal set of models that do not intersect this region. So we can pick two non-infinitesimal sets of models for which this hourglass function is an invariant.

At this point we briefly return to the question of complete invariant functions, that is, functions without false positives, considering now restricted libraries of objects. We consult our representation of a model's images as planes in affine space rather than as lines in α space and β space. There is a complete invariant function for a specific set of models if and only if no two planes that correspond to two models

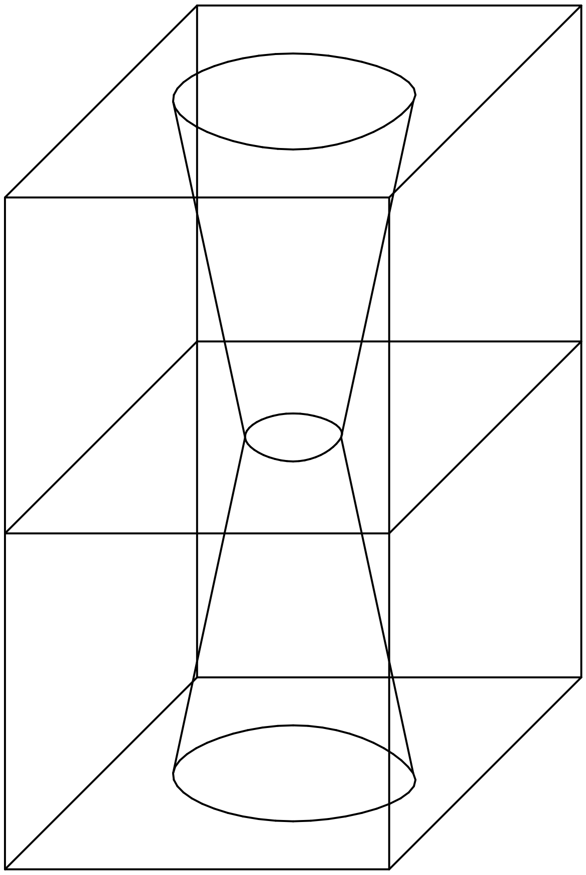


Figure 5.4: An hourglass shaped region of α space.

intersect without completely coinciding. If this condition is met, we may construct an invariant function that assigns a common value to all the images that fall on a model's plane of images, and assigns a different value to any two images that lie on the planes of different models. Then, any model that can produce one image with a particular value of the invariant function will be able to produce exactly the set of images that have that value of the invariant function. If, on the other hand, two planes intersect and do not coincide, then all images that either plane contains must have the same value for any invariant function, but neither object can produce all these images, so false positives will occur.

We can now see that a specific set of models can have an invariant function with no false positives only if it is an infinitesimal subset of the set of all possible models, because any set of non-intersecting planes is an infinitesimal subset of the set of all planes that correspond to models. For example, if we have a restricted set of models corresponding to a set of non-intersecting planes in affine space, this means that any point in affine space can belong to only one of these planes, although it belongs to uncountably many planes that correspond to some model.

5.3.2 False Negative Errors: Non-Accidental Properties

We now turn to a topic closely related to invariants: non-accidental properties (NAPs). When used for recognition this is a property of an image that is true of all images of some objects, and is false of all or almost all images of the remaining objects. Thus an NAP can be thought of as an invariant, in which some false negative conclusions are drawn, because images are not matched to models that rarely produce them. With NAPs, the connection between invariants and 3-D scene reconstruction is clear. NAPs in an image are used to infer 3-D properties of the scene that produced the image.

The theoretical results that we have built up will allow us to draw simple and general conclusions about the capabilities of NAPs, although the reader must always keep in mind that our results will only directly apply to the case of a linear transformation applied to models consisting of point features. After reviewing some of the past work on NAPs, we will show how to characterize the set of all possible NAPs as particular subsets of image space. This will make clear that there are an infinite set of possible NAPs, and that any one NAP is only valuable if we make specific assumptions about the particular class of object models that we are interested in detecting. We then discuss limitations that exist in attempting to make use of an ensemble of different NAPs.

The importance of some now commonly used NAPs was first discussed by some of the gestalt psychologists, who noted that properties such as proximity or symmetry tend to make a group of image features salient. We perceive a set of symmetric points in an image as a single whole, for example. Kohler[67] and Wertheimer[113] summarize some of this work. Lowe[73] also provides a useful discussion of gestalt and other early work on perceptual organization.

In computer vision, much work on perceptual organization has focused on NAPs. In general, an NAP is taken to be some property of the image that seems very unlikely to occur by accident, and so reveals the presence of some underlying non-random process. This idea is discussed and applied to a variety of vision problems by Witkin and Tenenbaum[115]. It is suggested that NAPs be used to infer 3-D structure from a 2-D image by Binford[11] and Kanade[64]. Kanade states the NAP criteria as: "Regularities in the picture are not by accident, but are some projection of real regularities". He suggests, for example, that parallel lines in the image come from parallel lines in the scene because, when scaled orthographic projection is assumed, parallel scene lines always project to parallel image lines, while non-parallel scene lines can project to parallel image lines from only an infinitesimal set of possible views.

Lowe[73] takes a more explicitly probabilistic approach to applying NAPs to object recognition. He also selects as NAPs properties of a 2-D image that some 3-D scenes will produce from a large range of viewpoints. We have mentioned that parallel scene lines always produce parallel image lines, with scaled orthographic projection. Similarly, if 3-D lines are connected, they will always be connected in the image. Since we can not expect noisy images to produce perfect examples of these properties, however, we must also make use of approximate instances of parallelism or group together lines that are nearby rather than connected. Lowe computes the likelihood of such approximate features occurring by chance, assuming some random distributions of image features. Then a property is useful when its presence in the scene guarantees its presence in the image; and is more useful the less likely it is to arise by chance. Lowe is using NAPs to infer scene properties from image properties, and the probabilistic machinery allows his system to determine the relative strength of each possible inference, which in turn orders his recognition system's search for objects.

As described in more detail in chapter 1, Biederman[9] takes Lowe's work as a starting point, and uses NAPs as the basis for a recognition system that attempts to cope with large libraries of objects. Burns and Riseman[23] also incorporate NAPs into a recognition system. NAPs have also been explored by Denasi, Quaglia, and Renaudi[40], and they have been used in a stereo system by Mohan and Nevatia[81].

This past work on NAPs has left a number of open questions. First, although past work has pointed out a number of examples of NAPs, they do not provide us with an exhaustive list of possible NAPs. We can not tell, for example, whether there are a small number of NAPs, or an infinite set of possible NAPs. We also lack a basic geometric insight into NAPs. What is it about certain geometric configurations that make them an NAP? We will provide a precise answer to this question. Second, it is not completely clear whether the value of an NAP depends on the particular

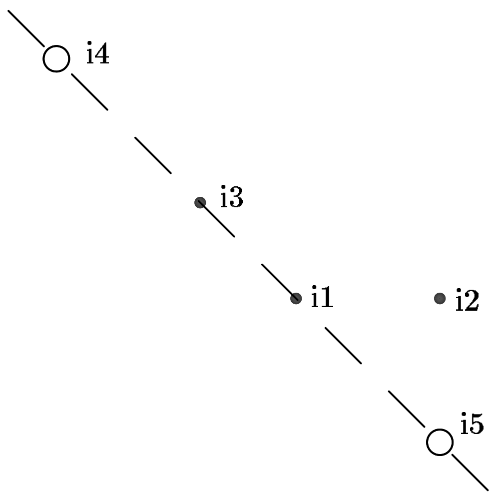


Figure 5.5: If an image corresponds to the point $(0, 0)$ in α-space, that means that the last two points in the image are collinear with the first and the third points. This is shown above, where the first three points are shown as dots, and the second two points, shown as open circles, must lie somewhere on the dashed line.

distribution of object models that we assume. Witkin and Tenenbaum seem to suggest that NAPs are useful only if we assume that processes in the world tend to produce certain types of scenes. On the other hand, Lowe seems to suggest that the usefulness of parallelism can be justified without assuming that models are particularly likely to contain parallel lines. The view that NAPs are inherently important cues is bolstered by the fact that the NAPs that have been used in vision are particularly salient, perceptually. Parallelism, collinearity, and symmetry, for example, can jump out of an image at the viewer. However, we show that this perceptual salience cannot come from the non-accidentalness of these properties. An infinite set of NAPs exist, and they are mostly not perceptually salient. We then show that NAPs are only useful if we make special assumptions about the models that produce images. Finally, as only a small set of NAPs have been used for recognition, one wonders whether one could achieve greater coverage of a wider set of images and models by using a large collection of NAPs. We will show that this strategy has significant limitations.

We begin by showing how NAPs may be considered within our geometric framework. A feature of any sort may be thought of as a subset of image space. Image space is just our representation of images, and the set of all images that contain a particular feature will map to some subset of image space. To illustrate this fact, we will consider the feature collinearity. Suppose that an image has five points, and the fourth and fifth points are collinear with the first and third points, as shown in figure 5.5. Such an image will have (α_4, α_5) coordinates $(0, 0)$, because the vector from the first image point to the fourth and fifth image points is entirely in the direction of the vector from the first to the third image point. The image might have any values for (β_4, β_5). Therefore, we can describe this kind of collinearity by pointing out that all images with this collinearity will map to a single point at the origin of α space. As another example, if the fourth image point is collinear with the first and second points, then $\beta_4 = 0$, while the other affine coordinates may have any values. Therefore this collinearity is described by a line in β space. More generally, any image feature is described by some region in an image space. If we consider features that are invariant under affine transformations, then these features may be described by regions in affine space. We have given examples of features that may be even more simply described by regions in α and β space. It seems reasonable to focus on affine-invariant features. This amounts to assuming that the features we wish to use in describing a photograph will be the same when we view the photograph from different directions.

A non-accidental property may be defined as a feature that some objects always produce, and that other objects produce from only an infinitesimal portion of possible viewpoints. The collinearity feature that we described with the point $(0, 0)$ in α space is one example of such a feature. Using the type of reasoning that Kanade or Lowe did, we can show this by pointing out that an image with four collinear points can occur in two ways. A model with four points that are collinear in 3-D would always produce an image with this collinearity. Or, a model with four coplanar points could produce this collinearity when it is viewed from a point within this plane, which is only an infinitesimal set of viewpoints. From our perspective, we can see the non-accidentalness of collinearity by noting that an image described by coordinates $(0, 0)$ in α space could be produced in two ways. A planar model with these α coordinates would always produce an image with these coordinates. Or, a non-planar model that corresponded to a line in α space that went through the origin could produce such an image. But in this latter case, the image with collinearity would be only an infinitesimal point on the line that describes all the α coordinates of all the images that the model can produce. These two analyses are equivalent. Note that if four model points are collinear, then the five model points will be coplanar, and have α coordinates $(0, 0)$. And there is a simple mapping from the set of all viewpoints of a model to the set of all images that the model produces.

However, our new view of NAPs as portions of affine space makes clear some questions that were previously obscure. For example, what is the set of NAPs? Any subset of affine space is an NAP if a plane that corresponds to any model is either entirely contained in this subset of affine space, or intersects it in only an infinitesimal part. For example, any point in α space corresponds to an NAP. Or, if our models have five points, then any 1-D curve in the 2-D α space will also be an NAP, while any 2-D subset of α space will not be an NAP.

Let's consider an example of a new NAP that we can discover in this way. Suppose

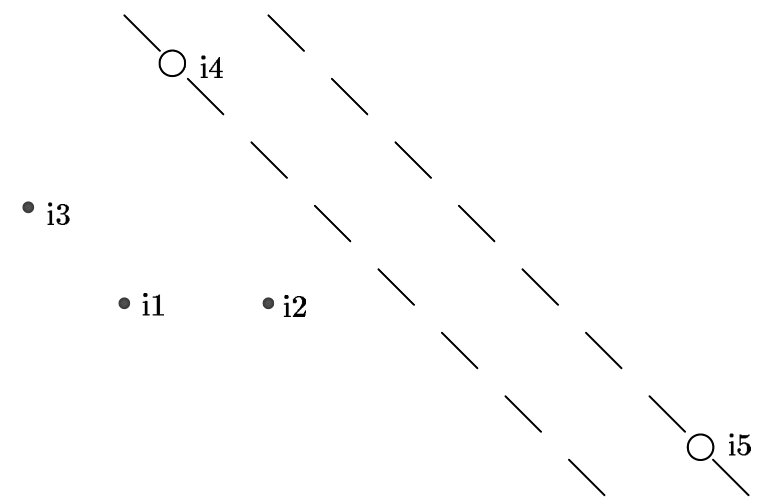


Figure 5.6: If the image corresponds to $(2,3)$ in α-space, the points must fall on the two lines shown.

models have five points. The point $(2,3)$ in α space corresponds to a new NAP. If an image has this NAP, it means that the fourth image point falls somewhere on a line parallel to the line connecting the first and third points. The distance to this line is twice the distance to the second image point. The fifth point is somewhere on another parallel line, that is three times as far away as the second image point. Figure 5.6 illustrates this.

This new property is as much an NAP as collinearity: both can be produced either by a planar model with the right α coordinates, which always produces an image with this property, or by a non-planar model that passes through a particular point in α space, and only rarely produces an image with this property. However, our new NAP appears to have no special perceptual saliency. In fact there is an NAP that corresponds to every possible image, the NAP formed by that image's affine coordinates. Not all these NAPs could be perceptually salient. This addresses the second question we raised above. The salience of properties such as collinearity or symmetry can not be explained by their non-accidentalness as we have defined it. It is clear that an NAP that is equivalent to a point in α or β space is limited. Using such an NAP amounts to inferring that the scene is planar, and such an NAP can only apply to an infinitesimal set of images, and hence to only an infinitesimal set of models. Such an NAP may be useful, but only if we make assumptions about the particular domain in which we are trying to recognize objects. Such an NAP is useful if we know that some such image property really does arise a lot, for example, if we know that real objects often do have parallel lines. That is, an NAP is useful when we assume a special distribution on possible models in which sets of models that produce an infinitesimal set of images actually occur with finite probability.

NAPs need not be restricted to a single point in α or β space, however. If we defined an NAP with a line in α space, this NAP would always be produced not only by a set of planar models, but also by the 3-D models that corresponded to that line. We might also hope to define a collection of NAPs that might together be more useful than they would be individually. So the question that still remains is whether a more complex NAP, or an ensemble of NAPs might have some useful properties. The limitations of either of these approaches is best understood by considering an NAP as a commitment to infer 3-D structure from a 2-D image.

An image, in our domain, corresponds to a single point in affine space. Without NAPs we can readily characterize the models that could produce this image: it could be a planar model with the right affine coordinates or it could be a 3-D model that corresponds to a plane that goes through the right place in affine space. When we use an NAP, we are choosing to accept some of these models as plausible matches to the image, while rejecting other models. If we have no prior knowledge about the likelihood of different models being correct, then there is only one meaningful distinction that we may make between the different models that could produce this image. That is the distinction between the planar model that always produces the same affine coordinates, and the 3-D model that rarely does [1]. This tells us that there is no criteria we can use to infer that the image came from one 3-D object rather than another. And it tells us that when we generalize the use of NAPs, we have a strategy that says: in the absence of other clues, assume that the model is planar, and then use an affine-invariant indexing strategy appropriate for planar models. This can be a useful strategy for locating planar parts of 3-D objects. Lowe used it successfully. However, it will never allow us to use information that comes from non-planar parts of an object to do indexing.

Let us return to the three questions we asked earlier in this section. First, we have a simple criteria that describes the infinite set of possible NAPs. Second, we can see that if we have no prior knowledge about which model properties are likely to occur, the only inference that we can reasonably make from an image is that a planar model is more likely than a non-planar model to have produced it. This is a generalization of such NAPs as parallelism or symmetry. But in generalizing these NAPs, we see how weak they are to begin with. This does not mean that past work on perceptual organization is not useful. Certain image properties are perceptually salient, and past work has pointed out the value to recognition of understanding this salience and using it for object recognition or for other visual tasks. However, we can not understand this salience from its non-accidentalness alone. We must understand what 3-D structures

[1] This is a bit of a simplification. We could also show that the closer a model is to planar, the more likely it is to produce affine coordinates like the ones in the image. This nuance does not affect the basic structure of our argument, however.

tend to occur in the world in order to explain why certain inferences about 3-D structure from a 2-D image are more valid than others. If we do not want to be stuck just inferring planarity we must look beyond the geometry of our features for regularities in the world that make certain features especially useful. Finally, with regard to our third question, we have shown that as we generalize NAPs it becomes clear that we do not have a viable strategy for recognizing 3-D objects, but only for handling planar portions of these objects.

5.4 Conclusion

In order to draw together the work in this chapter and in chapter 2 we must return to the themes that were laid out in the introduction to this thesis. We stressed the value of grouping and indexing, and suggested that there were two approaches that we can take to indexing. We can either try to characterize all the 2-D images that a 3-D model can produce, or we can try to infer 3-D structure from a single 2-D image. By understanding the best ways of characterizing a model's images, we have provided ourselves with the tools needed to show the limitations of trying to infer 3-D structure from a single 2-D image.

In chapter 2 we derived an optimal, error-free representation of a model's images as two lines in two high-dimensional spaces. In this chapter we have strengthened that result, by showing that when we consider representations that introduce small amounts of error we do not gain much. Small numbers of false positive and false negative errors will not give us invariant representations. And non-accidental properties, which allow small numbers of false negative errors, although they may be used effectively to index planar models, do not provide us with coverage over a wide range of 3-D model libraries. This shows us that if we want to design an indexing method capable of handling any collection of object models without introducing errors, then the best representation of a model's images to use is the one developed in chapter 2.

At the same time, this representation has provided us with a simple geometric interpretation of indexing that shows the limitations of attempts to infer 3-D structure from a 2-D image. Such inferences have been attempted with invariant descriptions in special domains, or with NAPs, which are properties that are invariant as long as one ignores models which can produce them from only a small set of views. However, in the absence of special assumptions about our model library, there is a symmetry to this problem that precludes such inferences. An image is a point in some space, a model may correspond to any line, so there is no reason to prefer any one line over another. The only sensible inference about 3-D structure from a 2-D image in our domain is the inference that the model must correspond to a point or a plane in affine space that includes the point that corresponds to the image.

We must keep in mind two important limitations of this work, however. Our goal has been to thoroughly explore a simple indexing problem. Therefore, we have primarily assumed that models consist merely of points, and that we have no domain-dependent knowledge. Our results might be taken to mean that no inference about 3-D structure is possible, and that no method of perceptual organization is to be preferred over another. Neither of these conclusions seem true. We have meant to show the limitations to using simple domain-independent geometric knowledge when performing perceptual organization or structural inference.

It is clear that we can infer more about 3-D structure if we assume that models consist of surfaces, or if we have access to shading or texture clues. More importantly, as in much of Artificial Intelligence, the limitations of domain-independent knowledge provide an impetus for understanding the structure of our domain. Surely symmetry is a valuable clue in perceptual organization because the world contains so much symmetry. We believe that in general our work supports the view that to understand images we must understand the regularities of the world that produces them, and not just the geometry and physics of the image formation process. Researchers have usually avoided this task, because it is difficult to formalize or to prove anything about the regularities that happen to exist in our world. It seems much easier to understand the regularities that must exist due to the laws of physics and the truths of geometry. Our goal in this chapter has been to show that while it is important to reason about images, there is a limit to what we can conclude about images without also incorporating a knowledge of the world.

Chapter 6

Convex Grouping

In the introduction we presented an overall strategy for recognition that combined grouping and indexing. But until now, we have focused only on indexing. We have seen that to be effective our indexing system requires groups of six, seven or more point features. It is not practical to consider all possible groups of image features of that size. So to be useful, our indexing system requires a grouping system that will select and order collections of point features that are particularly likely to come from a single object.

It has long been recognized that grouping is a difficult problem, which perhaps explains its relative neglect. Marr[77] said:

> The figure-ground "problem" may not be a single problem, being instead a mixture of several subproblems which combine to achieve figural separation, just as the different molecular interactions combine to cause a protein to fold. There is in fact no reason why a solution to the figure-ground problem should be derivable from a single underlying theory.

Marr recommended focusing on problems that have a "clean underlying theory", instead. These simpler problems included shape-from-shading, edge detection, and object representation. More recently, Huang[54] has stated:

> Everyone in computer vision knows that segmentation is of the utmost importance. We do not see many results published not because we do not work on it but because it is such a difficult problem that it is hard to get any good results worthy of publication.

However, in addition to its importance for indexing and for other visual processes, grouping deserves attention because even partial progress can be quite valuable. A grouping system need not fully decompose the scene into figure and background to be useful. Far from it. Without grouping, we must perform an exhaustive search. If grouping collects together features that are more likely to come from a single object than a random collection of features, than it will improve our search by focusing it. Many recognition systems now work on simple images by using very simple grouping techniques. Any extension in our understanding of grouping will extend the range of images in which a grouping system can make recognition feasible.

The grouping work described in this chapter serves two purposes. First, grouping and indexing are interdependent; we cannot demonstrate indexing as part of a fully functioning recognition system without automatically locating groups of point features in an image. Second, we describe some concrete progress on the grouping problem. We show how to efficiently locate salient convex groups in an image, and we show that these groups may be used to recognize objects in realistic scenes.

There are many different types of clues available for finding portions of an image likely to come from a single object, such as the distance between features, the relative orientation of lines, and the color, texture and shading of regions in an image. Which of these clues is most useful varies considerably from image to image, so ideally they should be integrated together, to build a grouping system that can make use of whichever clues are appropriate. In this chapter, however, we explore just one clue. We show how we can efficiently form salient collections of convex lines in an image. While in the worst case this is an inherently exponential problem, we show both theoretically and experimentally that we can efficiently find subgroups in practice. And we show that these groups are sufficiently powerful to support recognition in some realistic scenes.

We have given less attention to some of the steps that are intermediate between finding these convex groups and indexing with them. We present a simple method for robustly finding point features, starting with convex groups of line segments. We also present a simple method for selecting the most salient convex groups for use in indexing. We need to combine pairs of these groups in order to have enough information for indexing, but we have not developed a method of doing this, and so we simply perform indexing using all pairs of salient convex groups. Our research strategy has been to thoroughly explore one important aspect of the grouping problem, finding convex groups, and then to hook this together with our indexing system in a fairly simple way so that we may explore the interaction between the two processes.

After our exploration of indexing, we are in a much better position to explain the interrelationship of grouping and indexing than we could in chapter 1, and so we begin by showing why grouping is necessary for any recognition system based on indexing. We then discuss the value of convex groups for recognition. We present our method of finding convex groups, with a theoretical and experimental analysis of its performance. Finally, we fill in the remaining pieces needed to connect this grouping system to our indexing system, and demonstrate the performance of the complete grouping program.

6.1 Why Indexing needs Grouping[1]

We can make the relationship between grouping and indexing clear by returning to the combinatorics of recognition with point features, reviewing and extending the analysis given in chapter 1. In that chapter we showed how a simple recognition strategy, alignment, developed problems of computational complexity in challenging domains. We presented a formal analysis for recognition using point features. In that case, for 3-D recognition from 2-D images, alignment involves determining the object's pose using matches between triples of image features and triples of model features. We will now show that in this domain, grouping and indexing together can dramatically reduce our search, while either one individually is of only limited value.

Suppose that we have N image features, M model features, and n points in a group. Let $s(n)$ again stand for this average speedup, that is, the total number of model groups that could possibly match an image group of size n, divided by the average number of model groups matched to an image group through indexing. $s(n)$ is the reduction in the amount of search required when we compare an exhaustive search to one guided by indexing. We want to determine the number of hypothetical matches between image and model features that we must consider when we take various approaches to recognition.

Recall that with alignment we consider all possible matches between three image points and three model points. Each such match allows us to determine two possible poses of the model in the scene. The total number of such matches is approximately $\frac{(N^3M^3)}{3!}$, because there are $\binom{N}{3}$ image groups, $\binom{M}{3}$ model groups, and 3! ways of matching three image points to three model points.

The main implication of both the experimentally and theoretically determined speedup factors that we found for indexing in chapter 4 is that a recognition system based on indexing alone is not a practical alternative to alignment. First, indexing using no grouping and large values of n is clearly not feasible, because $\binom{N}{n}$ image groups will have to be explored, no matter what speedup is produced by indexing. Second, for smaller values of n, it is not possible to attain significant speedup, in comparison to the combinatoric explosion of matching all groups. If we try all combinations of image and model features, the overall number of matches will be $\approx \frac{M^nN^n}{n!s(n)}$, for $n \ll M, N$, since there are $\binom{M}{n}\binom{N}{n}n!$ possible matches between image and model groups containing n points, and indexing will weed out $\frac{1}{s(n)}$ of these matches. Therefore, to determine the effect of incrementing n, we should compare $\frac{MN}{n}$, the increase in the number of possible matches, to $\frac{s(n)}{s(n-1)}$, the increase in speedup. As we will see,

[1]This section is a modified version of material appearing in Clemens and Jacobs[32], and should be considered joint work between the author and David Clemens.

this comparison is unfavorable, and so it is more practical to use the minimum size of $n = 3$, which does not make use of indexing.

For example, with an image containing 100 points and a model with 50 points, there would be 1250 times as many matches of four points as there would be matches of three points, and 1000 times as many matches of five points as four points. First, from theoretical arguments, we found in chapter 4 that $s(n) < j^{n-3}$, where j is the size of a region describing the error in sensing a point divided by the size of the image. This implies that $\frac{s(n)}{s(n-1)} < j$. For $\epsilon = 5$, and a 500 by 500 pixel image, j is about 3200. Thus, even using this loose upper bound, increasing n from three to four, or from four to five will only decrease the number of matches found by indexing by approximately a factor of three. Furthermore, we did experiments to bound the possibilities of indexing by explicitly comparing image and model groups with a least squares method, to find matches that any correct indexing system must produce. From these experiments, we found that for scaled orthographic projection, $\frac{s(4)}{s(3)}$ ranges from 46 to 270, and $\frac{s(5)}{s(4)}$ ranges between roughly 46 and 54. With a linear transformation, we found $\frac{s(5)}{s(4)}$ to be 1,100. These experiments do not contain enough data to draw firm conclusions about the exact speedups possible with indexing, and we might produce better results by assuming smaller amounts of image error. But we can see that the speedups provided by indexing will provide little if any overall speedup in a recognition system that does not use grouping. Essentially, as the size of our groups grow, the number of geometrically consistent matches between groups of image and model points will either rise, or fall by only a small amount.

In this analysis, we are comparing alignment and an indexing approach that generates hypotheses that we must then evaluate. We should note two important exceptions to this analysis. First, one might use more complex features than simple points, such as vertices, curves, or collections of points. Indexing with such complex features should produce greater speedups. In our view, such complex features are the output of a grouping process, and in arguing for the value of grouping, we are equivalently arguing for the value of complex, rather than simple features. Second, we note that in the geometric hashing approach of Lamdan, Schwartz and Wolfson[70], indexing with quadruples of points can produce a more efficient system than alignment, because essentially the verification process is performed at the same time as indexing. The efficiency of the system comes because it does not need to separately verify each possible hypothesis, as alignment does.

Our experiments do, however, also indicate that indexing can result in greatly reduced recognition time when combined with some grouping. Grouping alleviates the need to form all combinations of features. Instead, groups of features that are likely to come from a single object are found. Let $P(n)$ be the number of image groups produced by a grouping method, and $Q(n)$ be the number of model groups. In order for grouping to be useful, $P(n)Q(n)$, the number of all matches between groups, must be much less than $N^n M^n$, the number of possible combinations of features.

With grouping alone, and no indexing, recognition must consider $P(n)Q(n)n!$ matches if grouping provides us with no information about the ordering of the points in a group. This quantity increases as n increases, partly because we expect that $P(n)$ and $Q(n)$ will increase with n. So it is still more desirable to generate hypothetical matches using small groups, as we see in Lowe's system[73]. However, when we combine grouping with indexing, the number of matches is $\frac{P(n)Q(n)n!}{s(n)}$. Since $s(n)$ increases exponentially in n, this quantity will decrease as n increases, as long as $n! \ll s(n)$. Thus, with indexing and grouping used together, larger groups may be used to significantly speed up recognition.

Furthermore, these figures assume that even with grouping, we must consider matching all permutations of an image group to a model group, because they assume that a group is just an unstructured collection of points. As we will see, grouping can also provide information on the order of points in a group that we can use to rule out most permutations. As a simple example, if we group lines into parallelograms, as Lowe did, and use the corners as point features, we know the order of the points around the parallelogram. There are only four different ways to match the points of two parallelograms, compared to twenty-four different ways of matching two general collections of four points each. This still does not mean that larger groups will be useful to a system that uses grouping without indexing, but it means that grouping and indexing combined can achieve even better performance.

6.2 Convex Grouping

Much work on grouping has focused on Non-Accidental Properties, and we have discussed that work in chapter 5. We mention here that Lowe first stressed the importance of using grouping when recognizing objects. His system estimated the likelihood that image lines approximating a parallelogram came from a model parallelogram using the proximity of the endpoints of the lines and the degree of parallelism in the image group. There has also been a good deal of work on image segmentation that focuses on dividing up an image into regions based on clues such as color and texture. Most segmentation work focuses on using regional intensity patterns to divide on image into its component parts. Recently Clemens[30] has considered methods of using regional intensity to form groups in the image for use in recognition, and Syeda-Mahmood[99] has considered using color-based segmentation to guide a recognition system.

Jacobs[60, 59] proposes using the relative orientation of edges to guide a grouping system that is combined with indexing. In this system, the proximity and relative orientation of two convex collections of line segments are used to estimate the likelihood that these two groups came from a single object. If one assumes that objects are made of convex parts, then one can show that the mutual convexity of collections of lines is a strong clue that they came from a single object, and one can estimate the strength of this clue from the overall shape of the lines. These grouping clues guide a search for an object in which the groups are tried in the order provided by the likelihood that they come from a single object. This system explicitly combined grouping and indexing, and it demonstrated that by building larger and larger groups, grouping combined with indexing could produce significant reductions in search. This system provides some of the justification for our current method of finding salient convex groups.

Convexity has also played an important role in a number of recognition systems that rely implicitly on some grouping process. Acronym[21] modeled objects using generalized cylinders that were convex. These convex parts projected to convex collections of edges in the image. A bottom-up grouping system then located ribbons and ellipses, the convex groups generated by the types of generalized cylinders used. Kalvin et al.[63] describes a two-dimensional recognition system that indexed into a library of objects. It began by finding unoccluded convex curves in the image, which it used for indexing. A variety of authors have proposed more general recognition systems that rely on finding the convex parts of objects. Hoffman and Richards[52], for example, suggest dividing objects into parts at concave discontinuities. Pentland[88] also suggests recognizing objects by recovering their underlying part structure using superquadrics. And Biederman[9] suggests performing recognition using the invariant qualities of an object's parts and their relations. While these parts need not be convex, in all examples the edges of a part are either convex, or are formed by joining two convex curves (as in the curved tail of a cat or the outline of part of a doughnut). In fact, in implementing a version of Biederman's work, Bergevin and Levine[8] rely on convexity to find the parts of an object.

Convexity may be useful for other types of matching problems, such as motion analysis or stereo. Mohan and Nevatia[81], for example, perform stereo matching between groups of line segments that form partial rectangles in each image.

In sum, the use of convexity for bottom-up aggregation is quite pervasive in recognition systems. It clearly can contribute to the perceptual saliency of a group of lines. Past work has shown that convexity can be a valuable grouping clue to combine with our indexing system. It also can provide information about the order of points within a group. At the same time, a thorough treatment of convexity can provide a middle-level vision module that would be useful to many other systems. For although many systems rely on finding convex edges in images, they find these convex groups through ad-hoc methods that either rely on the particular kind of convex shape being sought, or that rely on having data with clear, connected or nearly connected edges and few distracting edges. In this chapter we present an algorithm that finds all collections of convex line segments in an image such that the length of the line segments accounts for some minimum proportion of the length of their convex hull. This promotes robustness because all convex curves are found provided that a sufficient percentage of the curve is visible in the image. The algorithm is not stumped when there are gaps in a curve due to occlusion, due to a curve blending into its background, or due to failures in edge detection. And spurious line segments will not distract the algorithm from finding any convex groups that are present. We also show that this algorithm is efficient, finding the most salient convex groups in approximately $4n^2 \log(2n)$ time on real images.

While many vision systems that use convex groups have developed simple ad-hoc methods for finding them, there has also been some work that specifically addresses this grouping problem. Our own previous work on grouping, which focused on combining pairs of convex connected, or nearly connected, collections of edges, used a simple method to produce the convex groups that formed the initial input to our primary grouping system. Lines were connected if they were mutually convex, and if their end points were closer to each other than to any other lines. Because this initial grouping step was based on a purely local decision, it was sensitive to noise, and could combine lines that seemed good locally, but poor when viewed from a more global perspective.

Huttenlocher and Wayner[58] also present a local method for finding convex groups. They begin with each side of each line segment as a convex group, and then extend a group if its nearest neighbor will preserve its convexity. By only making the best local extension to each group, they guarantee that the output will be linear in the size of the input. And by using a Delauney triangulation of the line segments, they guarantee that nearest neighbors are found efficiently, and that total run time is $O(n \log(n))$. They also can efficiently form groups in which the best extension is judged differently, for example, extending groups by adding a nearby line that minimizes the change in angle of the group. However, convex groups are still formed using purely local judgements that do not optimize any global grouping criteria.

In both systems, a line may be the best local addition to a neighboring line, but may lead nowhere, causing each system to miss a better overall group. Figure 6.1 illustrates the sensitivity of these local methods of finding convexity. In the top left picture, the side of the box that says "ICE" is a closed rectangle. The local and global methods will all identify this rectangle as a convex group. On the left, we see two pictures in which the local methods for finding convexity will fail to find this rectangle. The top picture in the leftmost set shows that local methods may be sensitive to small gaps between edges if there are nearby distracting edges, while the bottom picture illustrates the fact that neither local method is resistant to occlusion. On the right, and on the bottom, we see some of the groups that a local method

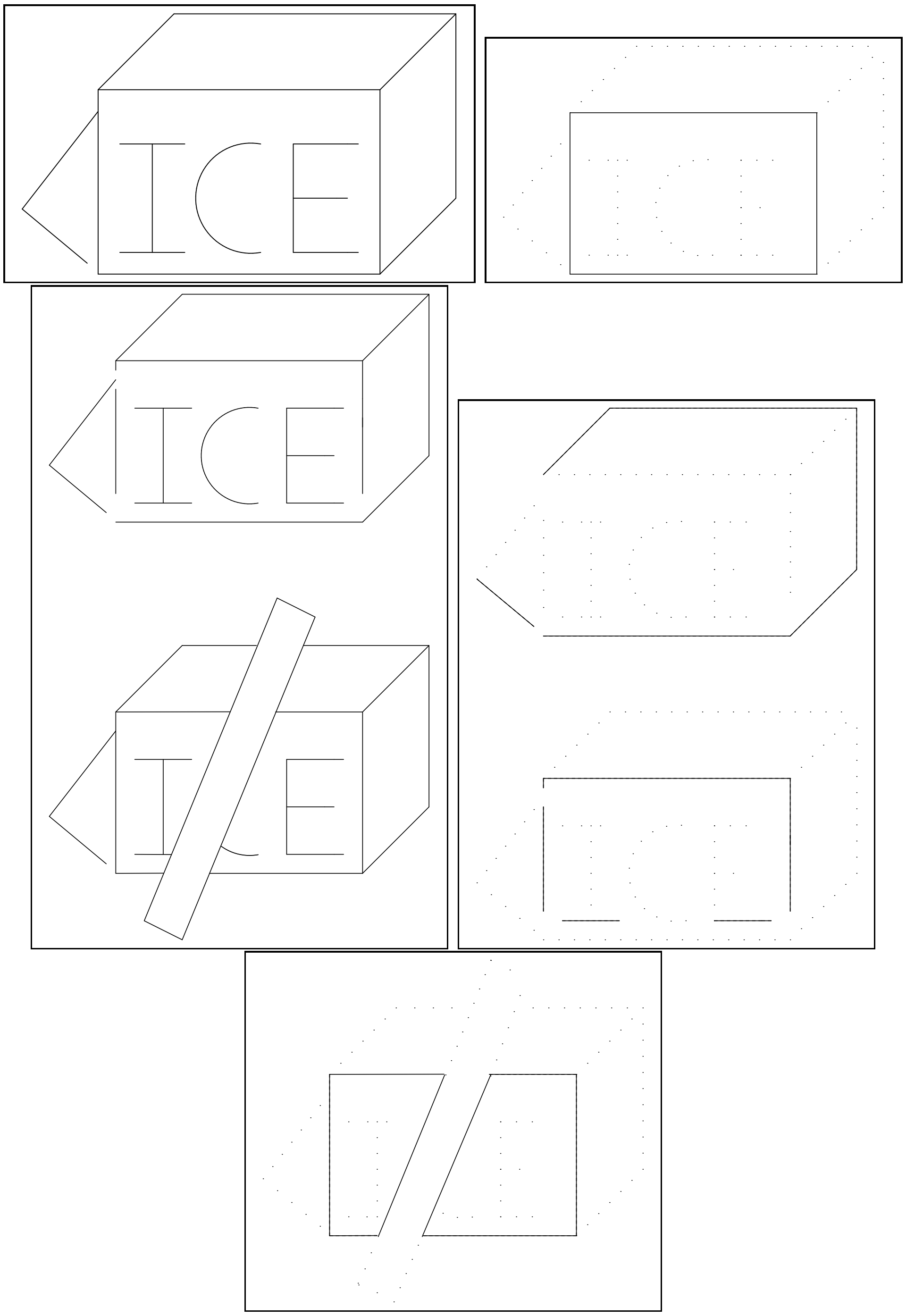


Figure 6.1: In the top left, we show a simple line drawing. Local methods can find the rectangle containing the word "ICE", as shown on the top right. In the two pictures on the left, these methods will fail. The three pictures on the right and on the bottom shows some of the groups that a local method might find in place of the central rectangle.

might find in place of the rectangle surrounding the word "ICE". In the case of our previous work on grouping, the failure that would occur in these images in the initial local grouping phase would have led to failure in the subsequent global grouping step. However, the convex grouping system presented in this chapter will succeed on all the images shown, because the gaps and occlusions in the figures will only slightly reduce the salience fraction of the foremost rectangle; it will still stand out as quite salient.

Shashua and Ullman[96] present a different kind of grouping system that is in some sense local and global. Their system finds curves that globally minimize a weighted sum of the total curvature of the curve and the total length of gaps in the curve. Their system optimizes a global criteria efficiently and finds perceptually salient curves. But it is only able to do this because their global criteria is reducible to a local one. The effect that adding a curve segment has on a group only depends on the previous curve segment, and is independent of the overall structure of the curve.

Along these lines, a number of other systems attempt to extract meaningful curve segments from an image. Mahoney[74] describes an algorithm for extracting smooth curves. The focus of this work is on developing an efficient parallel algorithm, and on deciding between competing possibilities when two curves overlap. Cox, Rehg, and Hingorani[36] describe a system that will partition the edges of an image into collections of curves. These curves will tend to be smooth, and may contain gaps. A Bayesian approach is used to find the curves that are likeliest to be the noisy images of smooth, connected curves in the scene. Zucker[116] and Dolan and Riseman[41]) also group together smooth image curves with small gaps. Other systems have found curves in the image that may be grouped together based on collinearity, (Boldt, Weiss and Riseman[24]), or cocircularity (Saund[93]).

These grouping systems all apply local criteria for grouping because it seems necessary for efficiency. In this chapter we show that a global criteria can be enforced in an efficient system.

6.3 Precise Statement of the Problem

Here we describe precisely the convex sequences of line segments the algorithm will produce, and introduce some useful notation.

The system begins with line segments that we obtain by running a Canny edge detector[25] (in the experiments shown, $\sigma = 2$), and then using a split-and-merge algorithm based on Horowitz and Pavlidis[87] to approximate these edges with straight lines. This system approximates curves with lines whose end points are on the curves, such that the curves are no more than three pixels from the line segments.

We call a line segment "oriented" when one endpoint is distinguished as the first endpoint. If l_i is an oriented line segment, then $l_{i,1}$ is its first endpoint, and $l_{i,2}$

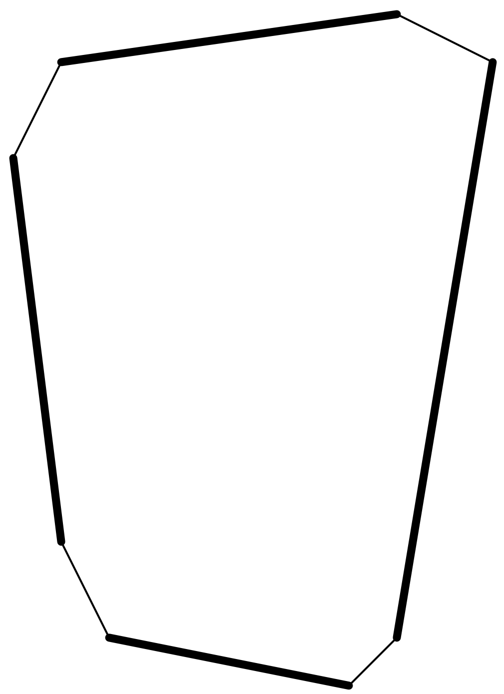


Figure 6.2: The thick lines represent the lines in a group. The thin lines show the gaps between them. The salience fraction is the sum of the length of the thick lines divided by the sum of the length of all the lines.

is its second. The image contains n line segments, and so it has $2n$ oriented line segments. The convexity of a set of oriented line segments depends on their having the appropriate orientation. That is, a set of oriented line segments is convex if for each oriented line segment, all the other line segments are on the same side of the oriented line segment as its normal, where we define the normal as pointing to the right when we face in the direction from the line segment's first endpoint to its second endpoint.

Let S_n be the cyclic sequence of oriented line segments: $(l_1, l_2, ...l_n)$, that is, l_1 follows l_n. We define L_i to be the length of l_i, and G_i to be the distance between $l_{i,2}$ and $l_{i+1,1}$, where G_n is the gap between $l_{n,2}$ and $l_{1,1}$. We then let $L_{1,n} = \sum_{i=1}^{n} L_i$ and $G_{1,n} = \sum_{i=1}^{n} G_i$. We say that S_n is *valid* if and only if connecting the line segments in sequence would create a convex polygon, and if $\frac{L_{1,n}}{L_{1,n}+G_{1,n}} > k$ for some fixed k. We call the fraction $\frac{L_{1,n}}{L_{1,n}+G_{1,n}}$ the *salience fraction* of the convex group (see figure 6.2).

Often, many of the subsets or supersets of a valid sequence will also be valid. To avoid excessive search that merely produces many similar sequences, the algorithm will not produce some sequences when valid subsets or supersets of those sequences are produced for which the salience fraction is higher. With this caveat, the algorithm produces all valid sequences of oriented line segments.

The important thing about the output of this algorithm is that it is guaranteed to satisfy a simple, global criteria. One way to think of the salience fraction is that if the lines forming a group originally came from a closed convex curve, the salience fraction tells us the maximum fraction of the curve's boundary which has shown up in the image. Use of a global salience criteria means that if a group is salient, adding more distracting lines to an image cannot prevent the algorithm from finding it, although a new line could be added to a salient group to make it even more salient. Occlusion can affect the salience of a group by covering up some of its edges, but will not otherwise interfere with the detection of the group.

6.4 The Grouping Algorithm

In this section we present an algorithm for finding these salient convex groups. We begin by presenting a basic back-tracking algorithm. We are able also to analyze this algorithm theoretically in order to predict its expected run time and the expected size of the output. We show that the actual results of running the algorithm match our theoretical predictions. We then make some modifications to the basic algorithm, which make it more robust, but which would make a complexity analysis more complicated. So we use experiments to show that these modifications do not significantly affect the algorithm's performance.

6.4.1 The Basic Algorithm

In order to find all sequences of line segments that meet our criteria, we perform a backtracking search through the space of all sequences. While such a search in the worst case has an execution time that is exponential in the number of line segments, this search is efficient in practice for two reasons. First of all, we are able to formulate constraints that allow us to prune away much of the potential search space. Second, much of the work needed to check these constraints can be computed once, at the beginning of the algorithm in $O(n^2 \log(n))$ time, and stored in tables. This makes the work required to explore each node of the search tree small.

Constraints for a backtracking search

Our problem definition is in terms of global constraints on the groups of line segments we seek. To perform an effective backtracking search, we must convert these into local constraints as much as possible. That is, we need constraints that determine whether a sequence of line segments could possibly lead to a valid sequence, so that we may prune our search.

First, we will list the constraints we use to decide whether to add an oriented line segment to an existing sequence of oriented line segments. We call a sequence *acceptable* if it passes these constraints. Then we will show that searching the space of acceptable sequences will lead to all valid sequences. These constraints may, however, produce some duplicate sequences or some sequences that are subsets of others. These may be eliminated in a post-processing phase. We provide the following recursive definition of an acceptable sequence, assuming that S_i is acceptable.

1. Any sequence of a singleton oriented line segment is acceptable.

2. S_{i+1} is acceptable only if $l_{i+1} \notin S_i$.

3. S_{i+1} is acceptable only if the oriented line segments in it are mutually convex. This will be the case if the sum of the angles turned is less than 2π when one travels from the first endpoint of the first line to each additional endpoint in turn, returning finally to the first endpoint.

4. S_{i+1} is acceptable only if: $G_i < \frac{L_{1,i}(1-k)}{k} - G_{1,i-1}$. This is equivalent to stating that $\frac{L_{1,i}}{L_{1,i}+G_{1,i}} > k$.

It is obvious that (1) and (2) will not eliminate any valid convex sequences. (3) guarantees that all the sequences we find are convex without eliminating any convex sequences. (4) states that if we are seeking a sequence of line segments with a favorable ratio of length to gaps between them, we need only consider subsequences that have the same favorable ratio. This trivially ensures that the final sequence will not have gaps that are too large, since the ratio of length to gaps is checked for the final sequence.

We must show that constraint (4) does not eliminate any valid sequences, however. To show this, we first notice that our backtracking search will try to build a valid sequence starting with each of the line segments in the sequence. In some cases this will cause our search to reproduce the same sequence of line segments, as we start at each line segment, tracing out the remaining segments of the convex sequence. To ensure that a valid sequence is always found, we must show that using at least one of the sequence's line segments as a starting point, constraint (4) will not apply to any of the subsequences we produce on the way to finding the entire sequence.

We will talk about subsequences, $S_{i,j}$ consisting of $(l_i...l_j)$. If $j < i$, we will mean the subsequence $l_i...l_n l_1...l_j$. We say for $S_{i,j}$ that its "ratio is acceptable" iff $\frac{L_{i,j}}{L_{i,j}+G_{i,j}} > k$. We call $(l_i...l_j)$ *continuable* if for all r, such that l_r is between l_i and l_j

in the sequence, S_{ir}'s ratio is acceptable. Then constraint (4) will not eliminate any valid sequence, S_n, if for some i, $S_{i,i-1}$ is continuable.

First we show that if two continuable sequences of line segments, $S_{i,j}$ and $S_{i',j'}$ overlap or if $S_{i',j'}$ begins with the line segment that follows the last line segment of $S_{i,j}$, then their union, $S_{i,j'}$ is continuable. To show this we must show that $S_{i,r}$ has an acceptable ratio when l_r is between l_i and $l_{j'}$. Clearly this is true when l_r is between l_i and l_j. If l_r is between l_j and $l_{j'}$, then $S_{i,r} = S_{i,j} \cup S_{i',r}$. $S_{i,j}$ and $S_{i',r}$ have acceptable ratios, and since $L_{i,r} = L_{i,j} + L_{i',r}$ and $G_{i,r} = G_{i,j} + G_{i',r}$, $S_{i,r}$ also has an acceptable ratio.

We can form the set of maximal continuable subsequences, i.e. no sequence in the set is a proper subsequence of another continuable subsequence. If this set has just one subsequence that covers the entire sequence of line segments, we are done. Otherwise, suppose $S_{i,j}$ is a maximal continuable subsequence. Then l_{j+1} must not belong to any continuable subsequence or this sequence and $S_{i,j}$ would together form a larger continuable subsequence. $S_{i,j+1}$ must not have an acceptable ratio or it would be a continuable subsequence. So we may divide the sequence into disjoint subsequences consisting of each maximal continuable subsequence and the line segment that follows it. Each of these subsequences has an unacceptable ratio, so the sequence as a whole must, a contradiction.

So we can find all valid collections of oriented line segments by searching through all sets of acceptable line segments. The constraints that define an acceptable sequence prove sufficient to greatly limit our search.

Pre-processing for the search

To further reduce the run time of our algorithm we notice that some computations are re-used many times in the course of such a search, so we precompute the results of these computations and save them in tables. In particular, we often wish to know whether two oriented line segments are mutually convex, and if they are we want to know the distance from the end of one segment to the beginning of the other. It is also convenient to keep, for each oriented line segment, a list of all other line segments that are mutually convex with it, sorted by the distance that separates them. Finally, we precompute the angle that is turned when going from one oriented line segment to another. Calculating this information takes $O(n^2 \log(n))$, because we must sort $2n$ lists that can each contain up to n items.

We may now describe the backtracking search in more detail, noting how these results are used. The search begins by trying all oriented line segments in turn as singleton sequences. Given an S_i, we calculate $\frac{L_{1,i}(1-k)}{k} - G_{1,i-1}$. From constraint (4), we know that we only want to consider adding a line, l_{i+1}, when the distance from l_{i2} to $l_{i+1,1}$ is less than or equal to this quantity. Clearly we only want to add l_{i+1} if it

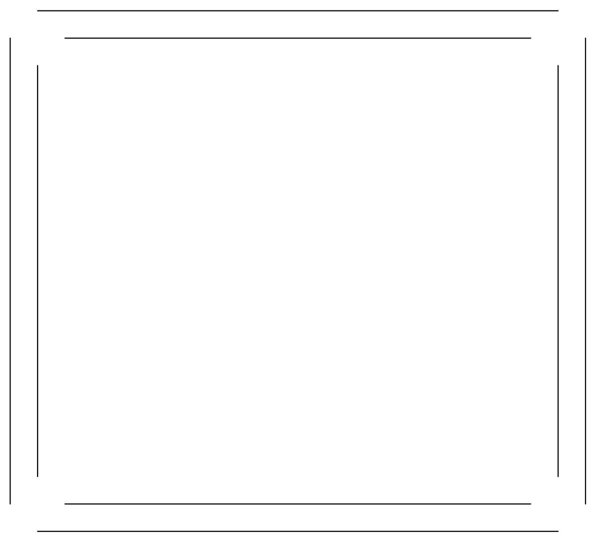


Figure 6.3: A salient convex group may be formed by choosing any line from each of the four sides.

is mutually convex with l_i. So we can find all candidates to add to the sequence by referencing our precomputed list of line segments that are convex with l_i. Since these lines are sorted by their distance from l_i we may loop through them, stopping once we reach line segments that are too far to consider. By limiting ourselves to these candidates, we have enforced constraint (4). In addition, we check that l_{i+1} is convex with l_1 using our precomputed results.

We can then enforce constraint (3) by keeping a running count of the angles turned as we traverse the line segments in S_i. A table lookup will tell us the angles added to go from l_i to l_{i+1} and from l_{i+1} to l_1. Therefore, we can ensure that the entire sequence is mutually convex by checking that the angles turned in traversing it sum to 2π. And constraint (2) is simply checked explicitly.

6.4.2 Complexity Analysis of the Basic Algorithm

In the worst case, this search will be exponential in both run time and in the size of its output. As a simple example of this, in figure 6.3 we show eight lines formed into a squarish shape. Even for fairly high values of k, we may form a salient convex group using either of the two lines on each side of the square. This gives us a total of 2^4 different square groups. If instead of a square we formed n-tuples of lines around an m sided convex polygon, we could easily construct an image with an output of at least m^n groups. By making the sides' endpoints close together, we can ensure that these groups are judged salient for any value of k less than 1. And the work required by the system is at least equal to the size of the output, so this would also be exponential.

However, we have found our algorithm to be fast in practice, and we can understand this by making an expected time analysis instead of a worst-case one. To do this, we need some model of the kinds of images on which the system will run. We model our image simply as randomly distributed line segments, and then compare the results of this analysis to its performance on real images.

There are many different random worlds that we could use to describe the image formation process. Our goal is to choose something simple and realistic. These goals are often in conflict, however, so we will often sacrifice realism for simplicity. We will also have to choose between several alternatives which may seem equally plausible. In making these choices and trade-offs, we will try to ensure that we provide a conservative estimate of our algorithm's performance.

We assume that an image consists of n line segments whose length is uniformly distributed from 0 to M, the maximum allowed length. This distribution is conservative because real images seem to produce shorter line segments more often than longer ones, while the presence of longer lines causes our algorithm to perform worse, since longer lines contribute to more salient groups.

Our overall strategy for this analysis is to derive two saliency constraints which may be considered separately. We use these to determine the likelihood that a randomly located line segment can be legitimately added to an existing group. We will assume that all line segments have angles of orientation drawn from a uniform distribution, and that the beginning point of a new oriented line segment is uniformly distributed within a circle of fixed radius, R, centered at the second endpoint of the last oriented line segment in the current sequence. A circle is the worst shape for our algorithm, because randomly distributed lines are closest together when distributed in a circle, again leading to more saliency. This likelihood will vary with the size of the group. By determining these likelihoods, we compute the number of partial groups that our search will have to explore at each level, which tells us how much work the search must perform overall. In the course of this analysis, we will make several more simplifications that will serve to make the analysis further overestimate the algorithm's run time.

First, let us illustrate the two constraints assuming that a group contains just one line, as shown in figures 6.4 and 6.5. We will call the length of this line m_1. We assume that the connection between the first line, l_1, and the second line, l_2 is made between the second point in the first line, l_{12} and the first point in the second line, l_{21}. Then from condition (4), given above, we know that the distance from l_{12} to l_{21} must be less than $m_1 k'$, where we let $k' = \frac{1-k}{k}$. This determines a circle around l_{12} where l_{21} must appear to preserve our saliency constraint.

Next, we have a constraint on the angle and location of l_2. Let us use a_i to

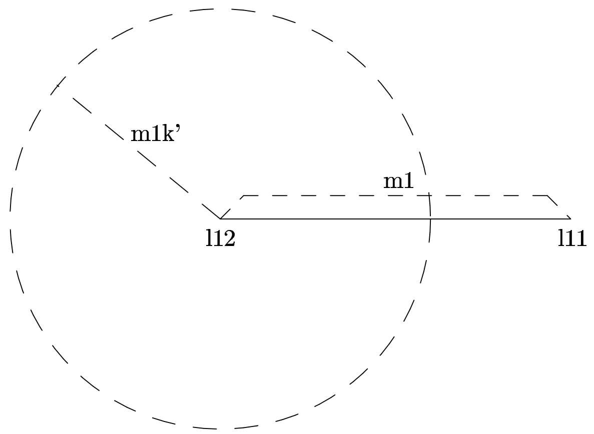


Figure 6.4: l_{21} must appear within the dashed circle to satisfy the distance constraint derived from the requirement that groups be salient.

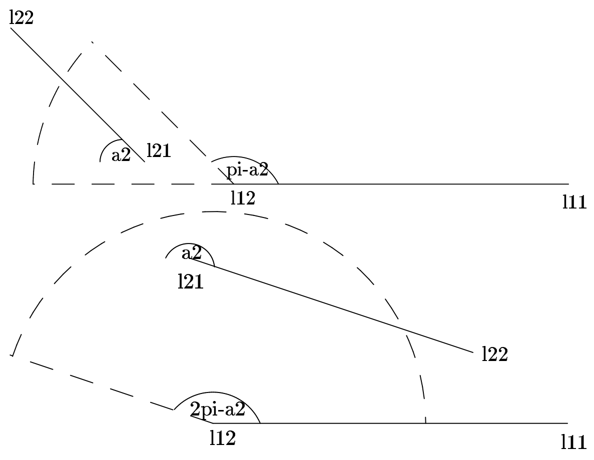


Figure 6.5: The dashed circular wedges show where l_{21} must lie in order to satisfy the distance and convexity constraints.

denote the angle of the vector from l_{i1} to l_{i2}, where the angle of $(-1, 0)$ is taken as 0, angles increase as we rotate clockwise, and without loss of generality we assume that l_1 has an angle of 0. Then the angle of l_2 restricts the location at which l_{21} can appear while maintaining convexity with l_1. For example, if l_2 has an angle of $\frac{\pi}{2}$ it must appear in the upper left quarter of the circle determined by the distance constraint. In general, as shown in figure 6.5, for $a_2 < \pi$, the angle formed by l_1 and the connecting line from l_{12} to l_{21} must be between π and $\pi - a_2$. If $a_2 > \pi$ a convex connection may still be made, but only if this angle is less than $2\pi - a_2$. In either event, l_{21} must be above l_1 to maintain convexity when connecting the lines. This derivation is a bit conservative; it does not capture all the possible constraint in our definition of convexity. In particular, it does not ensure that the connection from l_{22} to l_{11} preserves convexity.

If we consider groups with more than one line, these constraints change only slightly. The distance constraint reverts to the more general formulation: $G_i < L_{1,i}k' - G_{1,i-1}$. The angle constraint remains unchanged, because it reflects the constraint that the two lines be mutually convex. But we must add to it a constraint that all the lines be convex together. We can require that as we go from one line to the next in the group, we do not rotate by more than 2π, which we can express as: $\sum_{i=2}^{n} a_i - a_{i-1} \leq 2\pi$.

We now outline our strategy for computing the probability that an ordered group of n line segments will form a string that satisfy our salient convexity constraints. First we determine the probability that the distance constraint is met each step of the way, and take the product of these probabilities. Then given that the distance constraint is met at one step, and so l_{i1} falls in an appropriate circle about $l_{(i-1)2}$, we compute the probability that it will fall in the right part of the circle to produce convexity. We also take the product of these probabilities over all steps. Finally, we find a probability distribution for the sum of the angles our lines have turned, and use this to find the probability that the n lines will not have together turned by more than 2π. One thing that makes these computations much simpler is that they are essentially independent. Once we assume that the distance constraint is met, this does not effect the probability distribution of the slope of a line, or the angle to it from the previous line. Therefore, the angle constraints may be treated in the same manner every step of the way.

The Distance Constraint

As we have noted, the distance constraint requires l_{i1} to fall somewhere in a circle of some radius, call it r_i, so that the gap created is not too large. We also assume that the point is located somewhere in a circle of radius R according to a uniform distribution. Therefore, the probability of the distance constraint being met at any one step is the ratio of the area of these two circles. To find this, we need to compute the distribution of the radius of the first circle. As mentioned above, $r_2 = k'm_1$. Let $h(r_i)$ denote a probability density function on r_i.

It is useful to think of r_i in the following way. The ratio of the lengths of the lines in a group to the gaps must always be held above some threshold. r_i is the gap that is currently allowed between $l_{(i-1)2}$ and l_{i1}. However, if the distance from $l_{(i-1)2}$ to l_{i1} is less than r_i, than this gap isn't all used up. The excess gap may be carried over to the next line that we add on. In addition, the length of l_i (m_i) will make $k'm_i$ more gap available to us. Therefore, we can recursively compute $h(r_i)$ by alternating two steps. When we add the first line to the group, this increases the allowed gap by $k'm_1$. So we compute $h(r_2)$ using the distribution of m_1. Then when we attach another line to the group, we use up some of this allowed gap. Let s_i stand for the gap that is left over, that is $s_i = r_i - ||l_{(i-1)2}l_{i1}||$. Let $g(s_i)$ stand for its probability distribution. Then we can readily compute $g(s_i)$ from $h(r_i)$. We then compute $h(r_{i+1})$ from $g(s_i)$ by taking account of the fact that the length, m_i will allow some more gap.

To begin, we have assumed that the lengths of the line segments are uniformly distributed between 0 and M. Therefore the distribution function of m_i is $\frac{1}{M}$, and the distribution: $h(r_2) = k'm_1 = \frac{1}{Mk'}$.

Given a distribution for $h(r_i)$ we will want to compute two quantities. First, we will compute the probability that l_{i1} falls inside a circle of radius r_i. We indicate this as:

$$Pr(||l_{(i-1)2}l_{i1}|| \leq r_i) = \int_0^R h(r_i)\frac{r_i^2}{R^2}dr_i$$

Second, we want to use $h(r_i)$ to determine $g(s_i)$, given that l_{i1} does fall in a circle of radius r_i. For a fixed value of r_i, and for some value $\overline{s_i} \leq r_i$:

$$\begin{aligned} Pr(s_i \leq \overline{s_i}|r_i) &= 1 - \frac{(r_i - \overline{s_i})^2}{r_i^2} \\ &= \frac{2\overline{s_i}}{r_i} - \frac{\overline{s_i}^2}{r_i^2} \end{aligned}$$

Taking the derivative, we find:

$$g(s_i|r_i) = \frac{2}{r_i} - \frac{2s_i}{r_i^2}$$

Note that this equation holds only when $s_i \leq r_i$, since the probability that $s_i > r_i$ is always 0.

To compute $g(s_i)$ we must consider all values of r_i greater than s_i, and for each, determine the likelihood of such an s_i occurring. That is:

$$g(s_i) = \int_{s_i}^{\max(r_i)} h(r_i)\left(\frac{2}{r_i} - \frac{2s_i}{r_i^2}\right)dr_i$$

where $\max(r_i)$ is the largest possible value of r_i. r_i is bounded by $(i-1)k'M$, which is the maximum length of a group of $(i-1)$ lines, times k'. r_i is also bounded by R.

Next, given $g(s_i)$, we want to determine $h(r_{i+1})$. For a given s_i, $r_{i+1} = s_i + k'm_i$. The distribution of the sum of two values is found by convolving the distributions of each value. In this case, since $k'm_i$ is at most $k'M$ and no less than 0 we only consider values of s_i between $r_{i+1} - k'M$ and r_{i+1}, so:

$$h(r_{i+1}) = \int_{\max(0,(r_{i+1}-k'M))}^{r_{i+1}} \frac{g(s_i)}{Mk'} ds_i$$

In fact, this integral is a bit of a simplification, because r_i may never be bigger than R. If we ignore this effect, we are only exaggerating the likelihood of an additional line meeting the distance constraint, and hence overestimating the work that the system performs.

Given these relationships, we may compute any $h(r_i)$. This means that given that we have a group of i lines, we may compute the probability that another line will fulfill our distance constraints. While we could in principal find these values analytically, the integrals quickly become complicated, and so it is more convenient to compute these values numerically.

The Angle Constraint

There are two parts to our treatment of the angle constraint. First, we consider the probability that a line that passes the distance constraint will be locally convex with just the previous line. When a line, l_i, passes the distance constraint, we know that l_{i1} will be uniformly distributed in a circle about $l_{(i-1)2}$. As we mentioned above, the location of l_{i1} in this circle is constrained to lie in a wedge, and so the probability of this occurring depends only on the angle of the wedge, and is independent of the radius of the circle. The wedge's angle is $a_i - a_{i-1}$, the angle of l_i relative to l_{i-1}, provided that $a_i - a_{i-1} \leq \pi$. Otherwise, the wedge's angle is $2\pi - (a_i - a_{i-1})$. For a given angle of l_i, the likelihood of l_i being compatible with l_{i-1} is just the angle of this wedge divided by 2π. So integrating over all angles, which we've assumed are uniformly distributed, we find that there is a probability of $\frac{1}{4}$ that the lines will be compatible.

We must also consider the probability that a sequence of i lines will be mutually convex. We derive a distribution on the sum of the angles that must be turned as we go from one line to the next, and use this to determine the probability that this sum is less than 2π. This is a necessary, though not a sufficient condition on convexity. The distribution on each such angle is independent of the others and of the distance between the lines. So we need only consider the distribution on one of these changes of angle, and then convolve this distribution with itself i times to find the distribution on the sum of i such angles.

We know that the angle of one line relative to the previous one is uniformly distributed between 0 and 2π. We also know the relationship between the angle of the line and the probability that it is compatible with the previous line. So this probability of compatibility gives us a distribution on the relative angle of a new line, once we normalize it. If we let f be a probability density function on the change in angle of a new line, we have:

$$f(a_i - a_{i-1}) = \begin{cases} \frac{a_i - a_{i-1}}{\pi^2} & \text{if } a_i - a_{i-1} \leq \pi \\ \frac{2\pi - (a_i - a_{i-1})}{\pi^2} & \text{if } a_i - a_{i-1} > \pi \end{cases}$$

Convolving this distribution with itself is perhaps facilitated by noticing that this distribution is the convolution of a uniform distribution from $\frac{\pi}{2}$ to $\frac{3\pi}{2}$. This convolution is straightforward to perform analytically, but for our convenience we take it numerically.

Expected Work of the Algorithm

We are now in a position to determine the expected work our algorithm must do. As we have stated, there is a fixed overhead of $O(n^2 \log(n))$ work. In addition, we sum over all i the amount of work that must be done when we consider extending all groups of i lines by an additional line.

Since we are interested in the expected amount of work, we need to know the expected number of groups of length i that we will build. This is just the number of possible groups of that length times the probability that any one of them will pass our salient convexity constraints. If our image contains n line segments, we must consider two possible directional orderings for each line segment, so there are $2n(2n-2)...(2n-2i+2)$ possible ordered sequences of i line segments. Let δ_i be the probability that the $i'th$ line will pass the distance constraint, given that the previous lines have, and let λ_i be the probability that a group of otherwise compatible lines will have angles that sum to less than 2π. Then the expected number of groups of size i that we must consider, which we will call E_i, is:

$$E_i = \lambda_i 2n(2n-2)...(2n-2i+2)\delta_2...\delta_i \left(\frac{1}{4}\right)^{i-1}$$

where $E_1 = 2n$.

For each group we reach with i lines, there are potentially $2n - 2i$ lines that we must consider adding to the group. However, our preprocessing has sorted these lines, so that we only need to explicitly consider the ones that meet the distance constraints and that are convex with the last line in our group. We call the expected number of possible extensions to a group of size i, X_i, and:

$$X_i = \frac{1}{4}(2n - 2i)\delta_{i+1}$$

The total amount of work that we must perform in the course of our search is proportional to the number of times that we must consider extending a group. This work is given by:

$$\sum_{i=1}^{n} E_i X_i$$

This expression allows us to see that asymptotically, even the expected amount of work is exponential in n, because our run time is of the form:

$$c_1 n + c_2 n^2 + c_3 n^3 ...$$

where the c_i values are constants that are independent of n. This type of asymptotic behavior is uninteresting, though. It only tells us that as the image becomes arbitrarily cluttered, the number of salient groups becomes unmanageable. But of course we know that at some point, as an image becomes arbitrarily cluttered, recognition will be impossible. The key question is: when will this happen? An alternative approach is to perform an analysis of the algorithm's asymptotic behavior by assuming that as n grows the size of the image grows so as to maintain a constant density of lines in the image. Instead, we simply compute the expected run time for a variety of realistic situations.

To compute these values, we simplify the distance constraint by assuming that r_i is never larger than R. This is reasonable, because the maximum total length of lines in a group can never exceed $2\pi R$, and so r_i can never exceed $k'2\pi R$. So this limitation will have no effect when $k' < \frac{1}{2\pi}$, and will otherwise only effect rare groups that have very long lines with little gap between them. With this simplification made, R appears only when we use $h(r_i)$ to compute the likelihood that a random line segment end will fall inside a circle of radius r_i; we can ignore R in computing the values of $h(r_i)$ and $g(s_i)$. We can further simplify by choosing all distances in units of $k'M$. By assigning $k'M$ the value 1, we may compute all $h(r_i)$ and $g(s_i)$ once only, without any variables. We only need to make up for this when solving the equation:

$$Pr(||l_{(i-1)2} l_{i1}|| \leq r_i) = \int_0^R h(r_i) \frac{r_i^2}{R^2} dr_i$$

At this point, we replace R with its value written in units of $k'M$, that is, with $\frac{R}{k'M}$.

In table 6.1 we list the numerically computed values for the first 12 λs and δs[3]. The δ values are computed with $k'M = .25$ and $R = 1$. In practice, to find the δ values for a

[3] All data in this section shows the first three significant digits only.

Constant Probabilities of Adding a Line			
λ_1	1.00	δ_1	.021
λ_2	0.500	δ_2	.035
λ_3	0.0805	δ_3	.041
λ_4	0.00614	δ_4	.044
λ_5	0.000278	δ_5	.044
λ_6	0.00000848	δ_6	.045
λ_7	0.000000186	δ_7	.045
λ_8	$3.10\ x10^{-9}$	δ_8	.045
λ_9	$4.04\ x10^{-11}$	δ_9	.046
λ_{10}	$4.28\ x10^{-13}$	δ_{10}	.046
λ_{11}	$3.65\ x10^{-15}$	δ_{11}	.046
λ_{12}	$2.64\ x10^{-17}$	δ_{12}	.046

Table 6.1: The constant probabilities used to determine the likelihood that a random line can be added to a salient convex group.

different value of R, say R', we simply multiply the above values by $(\frac{R}{R'})^2$. In principal, things are not this simple, because while this reflects the change in the value inside the above integral, it does not take account of the change in the limits of the integral. In practice, however, this effect is tiny because the function we are integrating becomes very small before reaching the limits of integration. We have therefore shown that after numerically computing one set of constants, we can then analytically compute all relevant probabilities, making only unimportant approximations.

An important question is whether we can compute the expected work of the system without having to consider every value in the summation, that is, whether E_iX_i becomes negligible as i grows larger. As an example, with $n = 500$ and $R = 1$, the first twelve terms of the summation are:

$$5,180 + 45,200 + 231,000 + 401,000 + 337,000 + 170,000$$
$$+57,600 + 14,200 + 2,640 + 386 + 45.6 + 4.44$$

In this case we can see that the trailing values become small, relative to the total. Intuitively, we can also see why the trailing values of the summation will continue to shrink. When taking the i'th value of the summation, we multiply the previous value by something less than $2n$ times $\frac{1}{4}$, δ_i, and $\frac{\lambda_i}{\lambda_{i-1}}$. While δ_i rises as i increases, it quickly approaches an equilibrium point at which the average gap used up in adding a line to a group equals the average gap allowed by adding a typical line. λ_i is just being repeatedly convolved (twice at each iteration) with a constant function, causing

Expected Work, for M = R							
Number of lines	k						
	.6	.65	.7	.75	.8	.85	.9
200	$1.99\ x10^7$	2,270,000	346,000	65,800	15,100	4,280	1,650
300	$5.82\ x10^8$	$4.14\ x10^7$	4,200,000	561,000	93,200	19,000	5,200
400	$8.73\ x10^9$	$4.23\ x10^8$	$3.04\ x10^7$	3,020,000	386,000	61,000	12,700
500	$8.53\ x10^{10}$	$3.05\ x10^9$	$1.62\ x10^8$	$1.24\ x10^7$	1,260,000	162,000	26,800
700	$3.45\ x10^{12}$	$8.09\ x10^{10}$	$2.65\ x10^9$	$1.29\ x10^8$	8,830,000	791,000	90,700
1000	$2.25\ x10^{14}$	$3.76\ x10^{12}$	$7.54\ x10^{10}$	$2.11\ x10^9$	$8.77\ x10^7$	5,060,000	378,000

Table 6.2: This table shows the expected work required to find convex groups. As the number of lines in the image and the salience fraction, k, varies, we show the number of nodes in the search tree that we expect to explore. The table does not show the number of steps spent in a preprocessing step. By $M = R$ we indicate that the length of lines are distributed according to a uniform distribution, with a maximum length equal to the radius of the image.

Ratio of search to preprocessing, for M = R							
Number of lines	k						
	.6	.65	.7	.75	.8	.85	.9
200	14.4	1.65	.25	.048	.011	.003	.001
300	175	12.5	1.26	.169	.028	.006	.002
400	1,410	68.6	4.92	.489	.063	.010	.002
500	8,560	306	16.3	1.25	.127	.016	.003
700	168,000	3,950	130	6.28	.431	.039	.004
1000	5,140,000	85,800	1,720	48.2	2.00	.115	.009

Table 6.3: This table is an adjunct to Table 6.2. It shows the expected number of nodes explored in the search tree divided by the number of steps in a preprocessing phase, for various image sizes and salience fractions.

it's tail to shrink at a faster than exponential rate. So overall, once $\frac{2n\delta_i\lambda_i}{4\lambda_{i-1}}$ is less than one, the terms in the summation continue to shrink.

We now use these values to compute some sample run times for the system. We compute two sets of examples. First we suppose that $M = R$, that is, that the longest line is the length of the radius of the image. This seems a reasonable upper bound on the length of the lines. Table 6.2 shows the expected work of the system as n and k vary. Table 6.3 shows the amount of work of the system divided by $(2n)^2 \log(2n)$. This tells us roughly the proportion of the system's work is spent in search, as opposed to fixed overhead, although one step of overhead is not directly comparable to one step of search. When this ratio is low, the run time is well-approximated by $(2n)^2 \log(2n)$. In tables 6.4 and 6.5 we show the same values for $M = \frac{R}{2}$.

Expected Work, for M = $\frac{R}{2}$							
Number	k						
of lines	.6	.65	.7	.75	.8	.85	.9
200	65,800	21,200	8,090	3,650	1,970	1,280	983
300	561,000	141,000	42,900	15,400	6,700	3,600	2,420
400	3,020,000	623,000	158,000	47,800	17,400	7,930	4,700
500	1.24 $x10^7$	2,150,000	468,000	123,000	38,700	15,300	8,030
700	1.29 $x10^8$	1.64 $x10^7$	2,750,000	571,000	143,000	44,300	18,700
1000	2.11 $x10^9$	1.83 $x10^8$	2.21 $x10^7$	3,440,000	658,000	155,000	49,400

Table 6.4: This table is similar to Table 6.4, except that the lines are assumed to have lengths drawn from a uniform distribution between zero and half the radius of the image. It shows the expected number of nodes explored in the search tree to find all salient groups.

Ratio of search to preprocessing, for M = $\frac{R}{2}$							
Number	k						
of lines	.6	.65	.7	.75	.8	.85	.9
200	.0476	.0153	.00585	.00264	.00143	.000927	.000711
300	.169	.0426	.0129	.00465	.00202	.00108	.000728
400	.489	.101	.0256	.00775	.00282	.00129	.000762
500	1.25	.216	.0469	.0123	.00388	.00153	.000805
700	6.28	.803	.134	.0279	.00697	.00216	.000915
1000	48.2	4.18	.504	.0785	.0150	.00353	.00113

Table 6.5: The companion to Table 6.4, this shows the expected number of nodes explored in the search tree divided by the number of preprocessing steps. This allows us to see roughly when each component of the algorithm will dominate the overall run time.

Actual work for random lines, M = R							
Number of lines	k						
	.6	.65	.7	.75	.8	.85	.9
200	347,000	112,000	31,300	8,590	2,270	601	123
300	3,120,000	892,000	208,000	45,600	8,760	1,670	306
400	17,400,000	4,970,000	1,130,000	172,000	27,000	4,050	692

Table 6.6: This table shows the actual number of nodes in the search tree that were explored when finding convex groups in randomly generated images. The lengths of the lines were generated from a uniform distribution from zero to half the width of the image. The points were then randomly located in a square image.

Later, we will compare this to the results of the full system on real data. For now, we compare this theoretically derived estimate of the system's work to simulated data, to determine the affects of various approximations that we have made. Our primary approximation has been to assume that the end point of one line will be uniformly distributed in a circle about the end point of a previous line, when even in simulation, lines will be uniformly distributed in a fixed image. Also, we do not apply the full convexity constraints in our analysis, because we do not ensure that a newly added line is convex with the first line in our group. It is possible that connecting these two lines would cause a concavity. We expect that these approximations should make our analysis conservative, overestimating the work required.

In this test we first generate collections of random line segments in a square. To do this, we choose the length of the line segment from a uniform distribution between 0 and half the width of the square, and we choose the angle from a uniform distribution. Then we generate possible locations of the line by picking its first end point from a uniform random distribution over the square. If the entire line segment fits in the square, we keep it, otherwise, we use the same length and angle, but generate a new location for the line until we find one that is inside the square. Table 6.6 shows the results of these experiments for a few different values of k, and of the number of lines. Comparing this table with table 6.2, we see that our analysis does conservatively estimate the work needed for grouping. It overestimates this work by between a factor of six and a factor of twenty, roughly.

We will discuss some additions to the algorithm that make it more suitable for real images, and then describe experiments run on real images before we discuss the significance of the system's run time. But first, we consider the size of its output.

Expected Size of the Output

The size of the output of our algorithm is also important to consider, but its importance depends on how we intend to use our grouping system. For example, if we want to try indexing using all pairs of groups that are produced by grouping, it is important that the number of groups be small, or forming all pairs of them could take too much time. On the other hand, if we intend to select the 20 or 30 most salient groups produced by the system, then the number of groups produced is not important to the efficiency of the system. Given a set of salient groups, we can readily find the most salient ones.

Since we in fact intend to use only the most salient groups for indexing, we want to estimate the size of the output for two reasons other than efficiency. First, if a random image will produce many highly salient groups, this is a sign that convex grouping will not be effective. It means that the "real" groups that reflect the actual structure of the scene are likely to be drowned out by random groups. Second, if we can predict the size of the output ahead of time, we can use this to decide how high to set our salience threshold based on the number of lines in the image. We want to avoid wasting time by picking a low salience value that will produce many random groups which we will only discard when we select the most salient groups. Therefore we can set our threshold to produce an appropriate output, reducing the work that we perform finding less useful groups.

The expressions above for E_i tell us the expected number of groups of any particular length that we will encounter in our search. We could use this as a bound on the size of the output, but this is an oversimplification. As we have noted, the values for E_i are exaggerations because they are not based on all the constraint provided by the convexity requirement. But in addition, just because a group is reached in our search does not mean we will accept it. When we reach a group of length i in our search, we have yet to take account of the length of the i'th line, or the gap between the i'th line and the first one. It is difficult to determine the probability distribution of this final gap, because it is dependent on the combination of i previous processes that built up our group. But we can approximate it very simply by just assuming that the last end point in the group is randomly distributed with respect to the first end point. Using that approximation, the expected number of groups of size i that the system will produce is:

$$\frac{1}{4}\delta_{i+1}E_i$$

To find the total number of groups expected, we just sum all these values for i from 2 up. We ignore groups of size one because such groups are never salient unless the salience fraction is less than or equal to .5.

Tables 6.7 and 6.8 show the number of groups that we expect to find using this

Expected number of groups produced, for M = R							
Number of lines	k						
	.6	.65	.7	.75	.8	.85	.9
200	51,400	5,850	885	166	36.3	8.75	2.09
300	996,000	70,600	7,120	946	154	28.8	5.62
400	1.12 $x10^7$	539,000	38,600	3,820	484	72.8	11.8
500	8.69 $x10^7$	3,100,000	165,000	12,500	1,270	158	21.7
700	2.50 $x10^9$	5.86 $x10^7$	1,920,000	92,800	6,340	561	57.8
1000	1.14 $x10^{11}$	1.90 $x10^9$	3.80 $x10^7$	1,060,000	44,100	2,530	179

Table 6.7: This table shows the expected number of salient convex groups that we expect our algorithm to produce, as the size of the image and the salience fraction vary.

Expected number of groups produced, for M = $\frac{R}{2}$							
Number of lines	k						
	.6	.65	.7	.75	.8	.85	.9
200	166	51.9	18.4	7.16	2.90	1.15	.395
300	946	236	69.2	22.9	8.12	2.91	.934
400	3,820	783	195	56.1	17.7	5.81	1.74
500	12,500	2,170	466	118	33.7	10.1	2.86
700	92,800	11,800	1,970	403	95.2	24.4	6.15
1000	1,060,000	92,300	11,100	1,720	320	67.2	14.4

Table 6.8: This table shows the expected number of salient convex groups that we expect our algorithm to produce, as the size of the image and the salience fraction vary.

Actual number groups for random lines, M = R							
Number of lines	k						
	.6	.65	.7	.75	.8	.85	.9
200	7,500	2,590	782	234	70	15	5
300	45,800	13,800	3,540	867	159	32	5
400	189,000	55,500	13,300	2,300	392	58	6

Table 6.9: The actual number of salient groups that were found in images of randomly generated line segments.

method. Table 6.9 shows the number of groups that we found in experiments with random line segments. There is a close fit between the predictions and the result of simulations, both for the system's run time and the size of the output. In particular, all these results show that the system will begin to break down at about the same time. The key point is that our analysis and experiments all agree on the values of k for which the system will run in a reasonable amount of time and produce a reasonable sized output. We will now discuss how we apply this algorithm to real data, and defer discussion of the implications of our analysis until we have presented experiments on real images.

6.4.3 Additions to the Basic Algorithm

Up until now we have presented our algorithm in its simplest form, to facilitate an analysis of its performance. We now discuss some modifications that make the system more robust by finding groups that are nearly convex, and that reduce the size of the system's output by eliminating groups that are similar or identical. Since these modifications make analysis difficult, we present experiments to show the effect that they have on the algorithm's performance.

Error in sensing or feature detection can cause lines that are convex in the world to appear nearly convex in an image. Two lines may be almost collinear, so as to create a slight concavity when joined in a group. To account for this, we decide that two lines are mutually convex when they are slightly concave, but nearly collinear. Or, one line, when extended might intersect the end of a second line, forcing a concavity when the lines are joined (see figure 6.6). To handle this possibility, we allow a portion of a line to participate in a convex group. This also allows us to find convex regions of edges even when some of these edges are approximated by a long segment that doesn't fit the region. We also may use just a portion of a line in a group even if using the whole line would still produce a convex group, if using just a portion of the line will make the group more salient by reducing the gaps between lines.

Our system will often find a number of similar convex groups. It will produce duplicates of a group if that group can be found starting with different lines. Subsets of a group will often pass our saliency constraint. For example, if four lines form a square, and the saliency fraction, k, is .75, then a group of all four lines will be duplicated four times, and four additional groups will appear containing every combination of three of the lines. These duplications are most easily handled in a post-processing stage. As long as our algorithm produces a reasonably small output, we can quickly sort through the groups it produces. When duplicates occur, we keep only one copy. If one group is a proper subset of another that was produced, we throw away the subset.

The system can also produce groups that are spurious supersets of other groups.

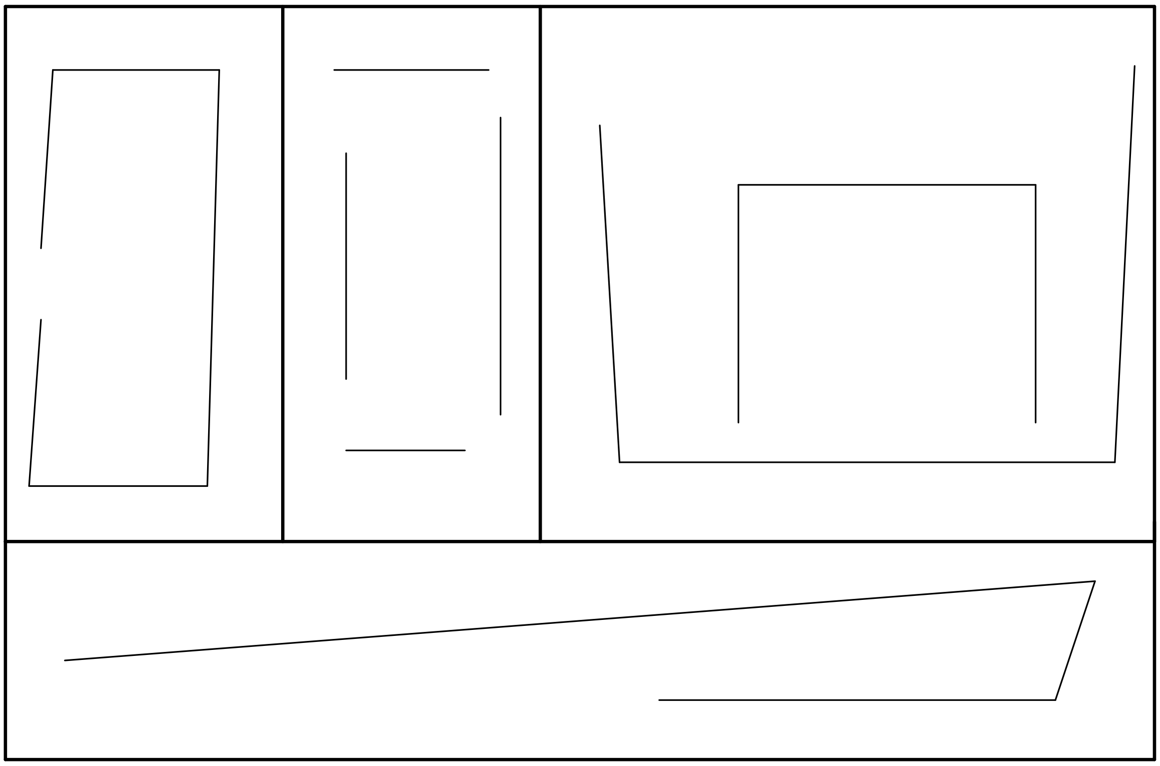


Figure 6.6: Four groups that our basic algorithm would not find. On the left, two of the lines in the group are nearly collinear, but not quite. In the middle, the upper line extends a little too far to the left to maintain convexity. On the right, the bottom line is not convex with the middle ones, but the central part of the bottom line can help form a strong convex group. In the bottom example, we can form a much more salient group by including only part of the top line.

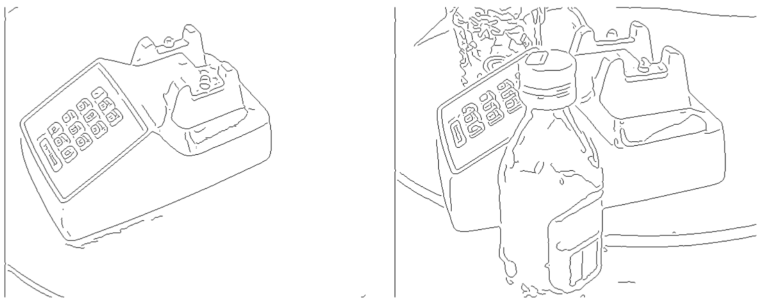


Figure 6.7: Edges from two of the real edges used to test the grouping system.

Often, a strong group will have added to it some combinations of additional small lines that reduce its saliency without changing the basic structure of the group. To avoid this, if our search has produced a string with lines: ABC, we do not consider the string $ABXC$ if the latter string has a lower salience than the former. Since only the salience of a string and its first and last line effect what other lines we may add to it, we are guaranteed that any group that is formed from the sequence $ABXC$ will also be formed from the sequence ABC.

The additions described above allow our algorithm to handle sensing error and to omit some groups that will be redundant in the recognition process. Since we will use these groups to derive stable point features, two groups with nearly identical sets of lines are likely to give rise to the same set of stable features, and we will not want to use both of them in recognizing an object.

After making these additions to our algorithm, we reran our experiments to determine the effect they have on the runtime of the system and on the size of its output. We ran both the basic algorithm and the augmented algorithm on a set of real images, so that in comparing these results to our previous results we can tell how much of the change is due to the use of real images, and how much is due to the additional constraints.

Figure 6.7 shows examples of two of the images we used. Table 6.10 shows the number of nodes explored in the search, for both the basic and full systems on these and similar images. Table 6.11 shows the number of groups produced both both variations of the algorithm on these images. Finally, figure 6.8 graphically compares the previous results of our analysis and tests on random images to these new results.

Comparing these results to previous tables shows a good fit between our theoretical predictions and the actual performance of the algorithm. We expect first of all that our analysis will overestimate the amount of work required by the system. Second, since we are overestimating the constants in an exponential series, we expect to have more and more of an overestimate as the later expressions in the series become more important. That is, we are overestimating the number of pairs of lines that our search

Actual work for real images								
No.	Alg.	k						
lines	Type	.6	.65	.7	.75	.8	.85	.9
183	basic	1,800,000	548,000	133,000	28,900	7,030	2,000	613
	complete	284,000	166,000	75,000	37,000	15,400	6,710	2,470
265	basic	5,630,000	1,660,000	420,000	102,000	27,200	6,370	1,410
	complete	496,000	288,000	136,000	55,300	18,200	7,440	2,800
271	basic			816,000	193,000	47,000	9,740	1,590
	complete			106,000	59,400	24,200	9,810	3,840
296	basic	7,200,000	1,820,000	429,000	93,300	16,300	3,350	946
	complete	273,000	163,000	89,400	41,800	14,800	6,170	2,900
375	basic	689,000	226,000	78,600	27,100	9,620	3,410	1,390
	complete	201,000	104,000	54,600	31,300	18,500	9,610	3,610
450	basic	2,090,000	696,000	227,000	69,000	21,300	6,420	2,160
	complete	295,000	163,000	92,500	48,400	24,800	11,400	4,440
461	basic			72,000	26,700	9,970	3,560	1,250
	complete			105,000	37,500	19,300	8,200	3,130

Table 6.10: This is the number of nodes explored in the search tree for some real images. The second column indicates whether we used the modifications to the algorithm described in the text to make it more robust, or whether we used just the basic algorithm.

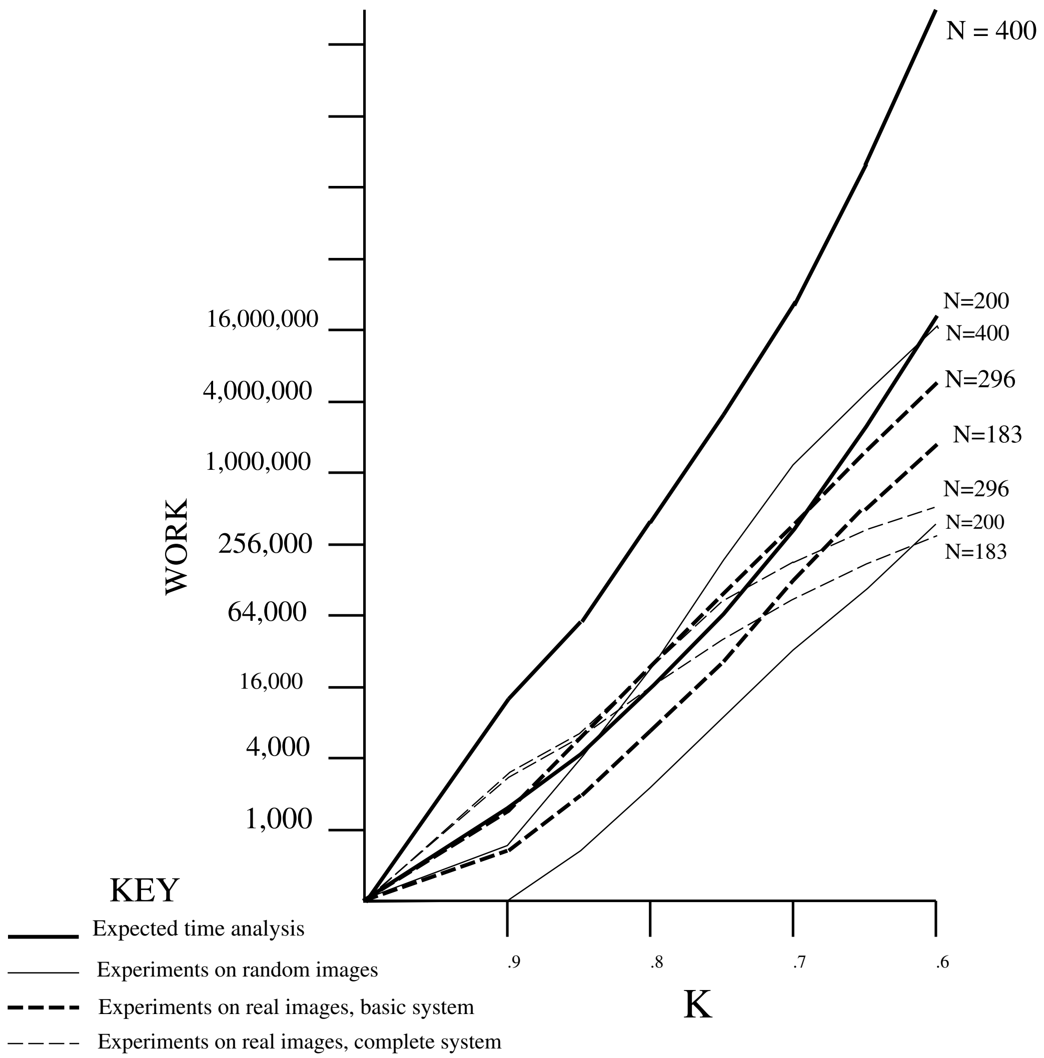


Figure 6.8: This graph compares the expected and actual number of nodes explored in the search for salient convex groups. The number of nodes are graphed on a log scale as k varies, for four cases: our theoretical predictions of expected work when $M = R$; the number of nodes actually searched with randomly generated images; the number of nodes searched on real images with the basic system; and the number of nodes searched on real images by the complete system. Only a sample of the results are graphed here, for clarity. Different cases are graphed with different brushes. On the right, N indicates the number of lines in the image. These are 200 and 400 for the analysis and random images, and 183 and 296 for the real images.

Actual number of groups found for real images								
No. lines	Alg. Type	k						
		.6	.65	.7	.75	.8	.85	.9
183	basic	16,300	4,760	1,170	248	110	60	35
	complete	2,160	1,190	494	315	134	69	31
265	basic	32,300	8,930	1,850	404	182	98	51
	complete	1,750	982	542	276	120	47	27
271	basic			5,600	598	148	64	36
	complete			540	291	157	65	34
296	basic	47,400	12,800	2,670	474	136	73	42
	complete	2,040	1,180	620	312	152	85	42
375	basic	23,100	11,100	5,250	2,390	1,020	406	163
	complete	1,680	1,020	536	331	188	122	48
450	basic	74,100	37,900	18,500	7,840	2,960	965	293
	complete	2,160	1,340	797	430	235	125	52
461	basic			754	376	194	96	49
	complete			863	368	178	84	34

Table 6.11: This table shows the number of salient convex groups produced by the two variations of our algorithm, when applied to a number of real images.

will encounter by underestimating the effects of our constraints. When we consider the number of triples of lines encountered, this overestimate gets compounded. We expect the overestimate to become more extreme in situations in which the higher order terms of our series come into play. This occurs as k shrinks and as N grows, when the number of larger groups considered becomes substantial.

This is in fact what happens. Although there is considerable variation in the size of the search from image to image, our theoretical analysis generally overestimates the amount of work to be done, even when we assume $M = \frac{R}{2}$, that is, a distribution of line segments in which the longest segment is half the radius of the image. And as k shrinks, the gap between our theoretical estimate and actual performance grows. All in all, we see that in real images we may use salience fractions of about .7 without causing the search to dominate our computation. We find that the real images produce somewhat more groups than our analysis predicts, assuming $M = \frac{R}{2}$. The numbers are of the same order of magnitude except when our analysis predicts a very small number of groups; it seems that in these images there is usually a minimum number of groups that will be found for each salience fraction, reflecting the basic structure that is present to a comparable extent in all the images. Looking at the size of the output to be produced, we find that a salience fraction of about .7 will also produce an output of roughly the same size as the input, making the output reasonable to work with.

There are two significant ways in which real images differ from the random images that we have analyzed. First, the real images contain many more short than long line segments; our assumption about the distribution of the lengths of lines was quite conservative. Second, real images contain much more structure than random images. While the structure of real images might cause considerable extra search, this does not seem to happen. One reason for this is that most of the system's search is spent finding that one cannot form salient groups from lines in well separated parts of the image. It is not surprising if, even in real images, collections of lines from separate sections of the image are well modeled as having random relative orientations and locations.

We also measured the actual run time of our implementations of the grouping system. However, because these programs have not been carefully optimized, numbers concerning actual run times should be regarded with some skepticism. The system ran on a Symbolics 3640 Lisp Machine. On an image with 246 lines, the basic algorithm spent 48 seconds on preprocessing overhead. The search tree was explored at a rate of between 450 and 2,300 nodes per second. Our implementation of the complete algorithm was approximately a factor of 20 slower. Preprocessing was particularly slow in the complete algorithm. However, our implementation of the complete algorithm was simple and inefficient, and we believe that most of the additional time it required could be eliminated in a more careful implementation. These numbers indicate that the overall system could be expected to run in a few minutes or less in a practical implementation, and that the difference in cost between a step of preprocessing and a step of search is about one order of magnitude.

6.4.4 Conclusions on Convex Grouping

We begin with a very simple, global definition of what constitutes a salient group. Although finding such groups is intractable in the worst case, we have shown that it is practical in real situations. Our constraints on salient convexity allow us to prune our search tree extensively, efficiently finding the most salient groups.

Although our experiments and analysis are long, they all support the same simple conclusion. In images with several hundred to a thousand lines, our algorithm is efficient if we set the salience fraction between .7 and .85. In these cases, run time will be roughly proportional to $4n^2 \log(n)$, and the size of the output will be roughly n. In fact, we see that the size of the output becomes unreasonable at about the same point at which the run time does. One way to condense all these numbers into a simple form is to ask how much computation is required to find the m most salient groups in any particular image. For although we cannot anticipate exactly how many groups will satisfy a particular salience fraction in a particular image, our analysis

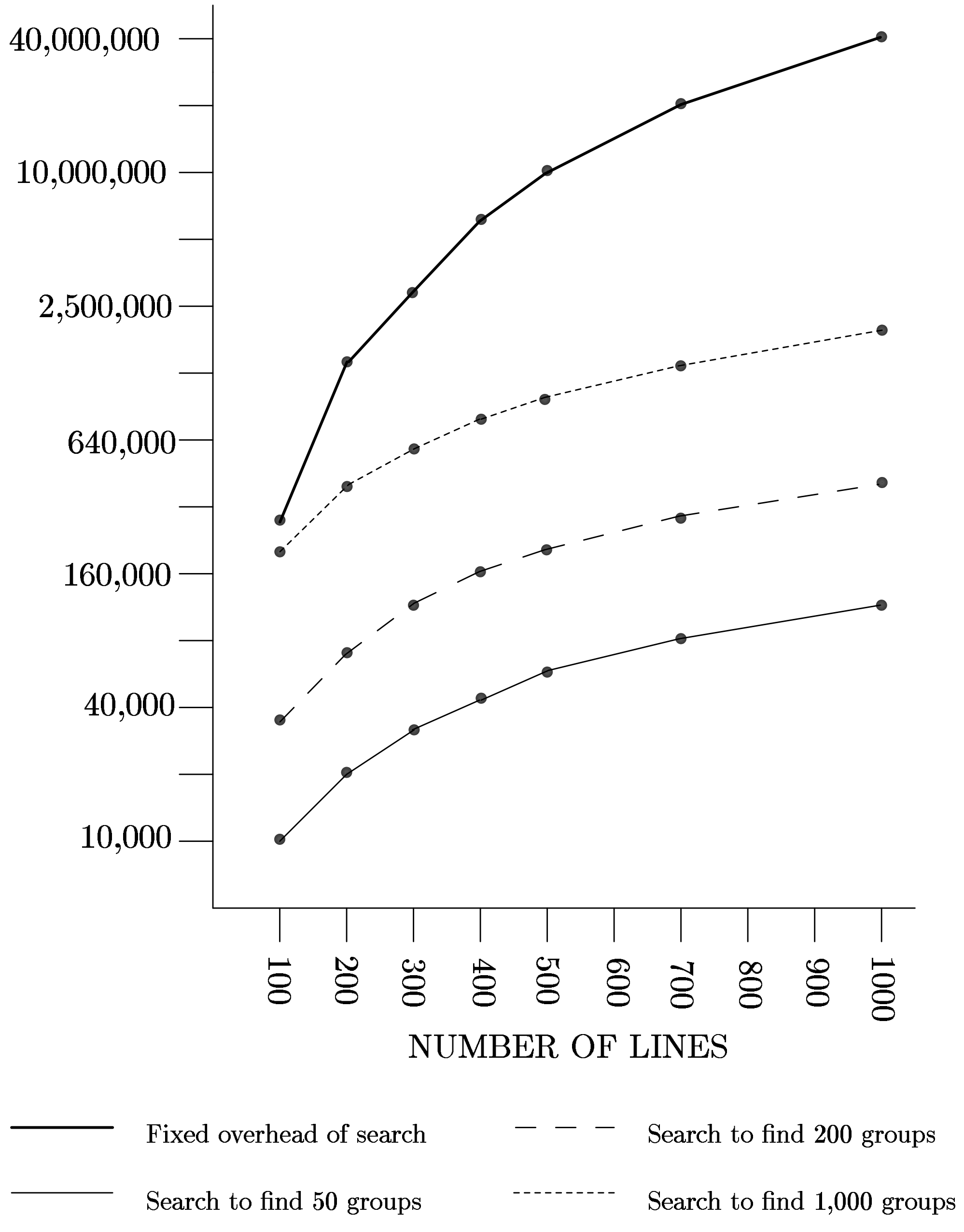


Figure 6.9: This graph shows the amount of work required to find the m most salient groups in an image, when we use our analysis to choose an appropriate salience fraction given the number of lines in the image, and m. The thick line shows the cost of our preprocessing phase, which is the same for any value of m. The other lines show the number of nodes we expect to explore in the search tree to find the desired number of salient groups. Note that the graph is at a log scale.

does predict this to good accuracy. So we may first use our analysis to estimate the salience fraction required to produce a particular number of groups, given the size of the image, and then use it to determine the cost of finding the groups in the image with that salience fraction. This allows us to capture the main points of the cost of our system in one graph. Figure 6.9 shows the variation in the number of steps of computation required to find a fixed size output, as n varies. The graph shows how many preprocessing steps are required, as n varies. Then, for different values of m, we use the above analysis to determine the value of k which will produce m salient groups. k varies with n. Then, we use these values of k and n to determine the expected number of nodes we will explore in our search. This graph shows quite clearly that for any reasonable sized output, the computation of the algorithm will be dominated by our initial preprocessing step.

This tells us that in practice, the limitation of salient convex grouping is that for extremely cluttered images, a very high salience measure must be used or we will not find a small number of salient groups. We have therefore demonstrated a simple salience clue that may be efficiently used in many real situations, while at the same time we can see the need for using stronger and different kinds of evidence of perceptual salience to handle especially large or cluttered images.

Although the algorithm stands or falls with its performance on real images of interest, we believe there is much value to a theoretical analysis of the algorithm's behavior. Our analysis has predictive value, it allows us to see when the system will break down. We can tell that as images become more cluttered, our system will continue to work if we demand greater salience in the groups that it produces. This allows us to set the salience threshold dynamically, if we wish, to ensure that our system will run quickly and produce a reasonably small output.

Hopefully this theoretical analysis also provides insight into the problem which can be used to improve on our algorithm. We know that salient convexity is not the end-all of perceptual grouping. It is too weak a condition, because it will not provide a small number of salient groups in a complex image. It is too strong a condition, because perceptually salient groups are not always convex. But understanding our algorithm well should be helpful in showing us how to extend it. For example, our salience measure does not pay attention to the angle between connected lines. If adjacent lines in a group are nearly collinear, the group will appear more salient than if they are at right angles, all other things being equal. And our analysis shows us exactly how much less likely collinear lines are to occur by accident than are perpendicular lines. So, while our current algorithm will add any line to a group that is within a circle of the end point of the group, we can imagine using an elliptical region that accounts for the fact that the less the angle between two lines, the farther apart they can be while still being equally unlikely to appear convex by chance.

In sum, by carefully analyzing and testing our system, we can determine exactly when it will be useful. We find that it will be of value in many realistic situations. We also can understand where the efficiency of the system originates. This should be especially valuable as we attempt to modify or add to it.

6.5 Stable Point Features

Our indexing system requires point features, not line segments. At compile time we must find groups of 3-D points to represent in our lookup table. At run time we must find the projections of these groups in the image. Our indexing system finds matches between image points and groups of model points that could have produced exactly those image points, no more and no fewer. A model group will not match an image group if some of the model groups' points are occluded, or if they are not found by the feature detector. It also will not produce a match if an image group contains extraneous points. We can make some allowances for occlusion by representing some subsets of model groups in our table. But we need to find 3-D points that consistently project to 2-D points that we can detect. The greatest difficulty in doing this is stability. If small amounts of sensing error or small changes in viewpoint can make a point feature appear or vanish, then the set of points that characterize a convex group will be always changing, and we would need to represent every possible combination of these points in our indexing table in order to recognize an object. If error causes the locations of points to shift drastically, then we cannot enforce reasonable bounds on the error that occurs in locating a point.

Our strategy is based on locating points at the intersections formed by the lines that a group's line segments lie on, as shown in figure 6.10. So we first focus on evaluating the stability of these potential point features, and then show how this stability measure is used to derive points from a group. We ask: Can small amounts of errors in the line segments have a large effect on the location of these intersection points? and: Can small changes in the location of edges obliterate a point altogether by merging the two line segments that form the point into a single line segment?

We handle these questions in two ways, depending on whether or not the two line segments are connected. If the intersection point is formed by an actual connection between the line segments, then we know that the segments are adjacent lines in the approximation of a string of edge pixels. In this case we may base our analysis on some properties of the straight-line approximation. Otherwise, we use a more general model of the error in locating our line segments to determine the error in locating their intersection point.

Suppose first that we have two unconnected line segments, which we call l_1 and l_2, with points numbered $l_{11}, l_{12}, l_{21}, l_{22}$ as usual. Their nominal intersection point is the intersection of two rays, with endpoints at l_{12} and l_{21}, and with directions given

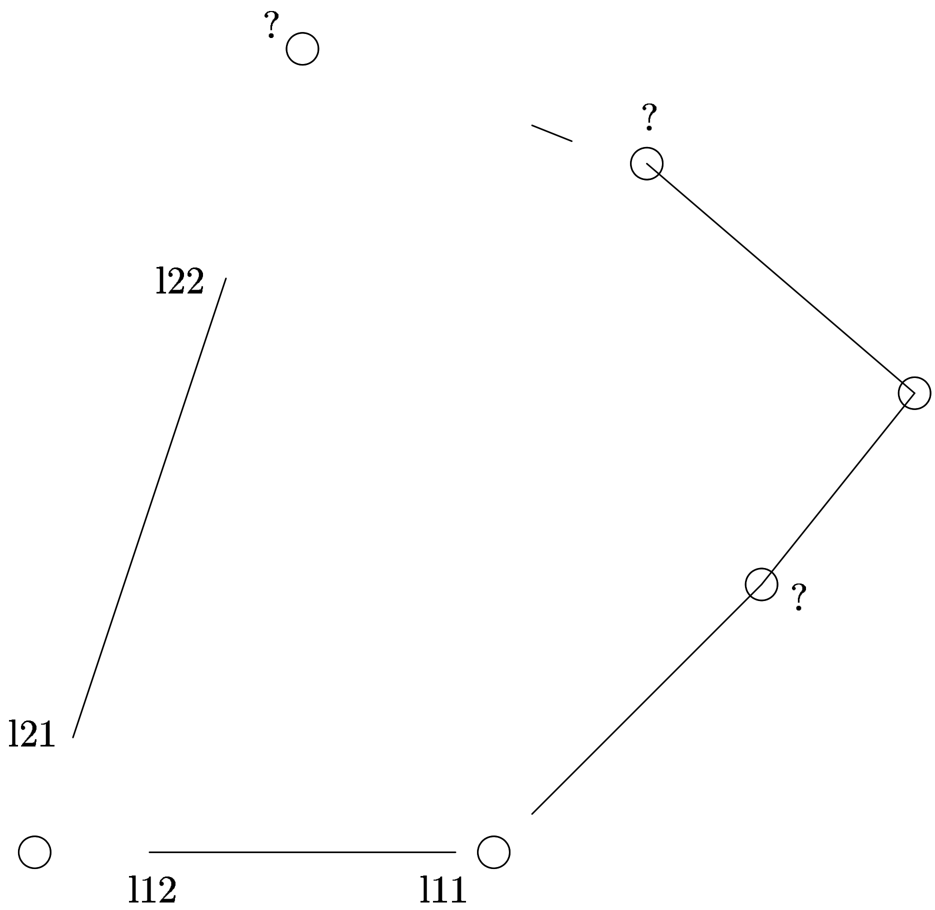


Figure 6.10: We show the lines of a possible group of features. Circles indicate the possible location of corner features. We have placed a question mark next to some corners that are possibly unstable. We label the end points of two of the lines that generate a corner point.

by the vectors from l_{11} to l_{12} from l_{22} to l_{21}. We assume that there is some error in locating the endpoints of each line segment, and see how this translates into error in the location of the intersection point of their corresponding rays.

We assume a fixed bounded error in the two points that are furthest apart, l_{11} and l_{22}. We call the amount of this error ϵ_1, and we have set it to five pixels. While these two points are usually widely separated, the points l_{12} and l_{21} may be quite close together, and allowing fixed errors in their locations may exaggerate the relative error that will occur. For example, if the points are only five pixels apart, it will often be an exaggeration to assume that they might be fifteen pixels apart. This is based on the empirical observation that relative uncertainty in locating features is often correlated when the features are nearby. Therefore, for the error in these points, we use either ϵ_1 or 10% of the distance between the two points, whichever is smaller.

If we think of the intersection point as arising from two rays, then we may separately bound the maximum effect these errors can have on the angle of the rays and on their location. Since the rays are located at l_{12} and l_{21}, the error in these points bounds the error in the rays' locations. The error in their direction is maximized as the points l_{11} and l_{22} are displaced normal to the line, while the other end points are fixed. Thus, the maximum variation in angle is given by $\arctan(\frac{\epsilon_1}{m_1})$ and $\arctan(\frac{\epsilon_1}{m_2})$, where we recall that m_i is the length of l_i[4].

With these bounds on the location and direction of the two rays, we find the maximum distance between their possible intersection point and their nominal intersection point by considering all combinations of extreme values for their angles and locations. If for any possible angles the rays do not intersect, this means that for some error values they are parallel, and the instability in the location of their intersection point is infinite. The maximum variation in the intersection point is then used as an estimate of a point's instability.

Suppose now that the two line segments intersect at a point, that is, that $l_{12} = l_{21}$. Then the problem becomes one of determining how much an anchor point can vary in the split-and-merge algorithm for straight-line approximations. With this algorithm, a curve is approximated by a straight line segment, which is recursively split into two segments that better approximate the curve by locating an intermediate endpoint at the point on the curve that is farthest from the line segment that approximates it. After the curve is sufficiently well approximated, adjacent line segments are merged back together if the resulting single line segment does not differ too much from the underlying curve. The variation of line segment endpoints due to variation in the underlying curve is hard to characterize with this algorithm because it can depend on events far from that anchor point. If distant parts of the edge that we are approximating are occluded or noisy, this can effect the entire approximation. This is easy

[4]Actually we approximate this by ignoring the arctan in the expression

to see when we consider approximating a circle, where the choice of anchor points is essentially arbitrary, and can change greatly with slight deformations in the circle.

However, we approximate this variation simply by assuming that the endpoints l_{11} and l_{22} are held fixed. We then allow the underlying curve to differ from the two line segments that approximate it by as much as ϵ_2. We set ϵ_2 to twice the amount that we allowed our approximation to differ from the underlying curve, that is to six pixels, since we know that even without sensing error, our straight line approximation may have introduced three pixels of error. We then ask how much the location of l_{12} can vary while keeping the original line segments to within ϵ_2 pixels of the new line segments.

We allow l_{12} to shift either forward along l_2, rotating l_1, or backward along l_1, rotating l_2, until the rotated line is ϵ_2 pixels from the original location of l_{12}. As shown in figure 6.11, if the angle formed by the two line segments, call it a' is acute, then the amount that l_{12} can shift along either direction is limited by ϵ_2 pixels. If a' is obtuse and l_{12} is shifted forward, we let $a = \pi - a'$, and we let b equal $\arcsin(\frac{\epsilon_2}{m_1})$ (which we again approximating by ignoring the inverse trigonometric function). Then the distance that l_{12} can shift forward is either infinite if $a < b$, or is $\frac{\epsilon_2}{\sin(a-b)}$. We compute the amount that the corner can shift backward similarly. The instability in the point is then taken to be the maximum of the amount that it could shift forward or backward.

Finally, we also want to take account of the fact that a slightly different approximation of the edges could eliminate a corner altogether, merging l_1 and l_2 into one line. So if a corner could forward more than the length of l_2, or backward more than the length of l_1, we automatically rule out that corner.

This process gives us a single number for each corner which estimates its instability. We must now set a threshold for ruling out some of these corners as unstable. We could set this threshold at the maximum error that our indexing system allows, but our determination of maximum instability is quite conservative, and such a strict threshold would rule out many reasonable corners. Also, in computing the instability of corners, we have used several fairly arbitrary constants, and so the absolute instability that we compute is probably not that reliable, although the relative instability that we compute between corners is useful. So we set a stability threshold of 15 pixels, which is just based on our experience with the system.

There is one more possibility we take account of in computing corners, illustrated in figure 6.12. It may happen that a pair of lines with a rounded corner is approximated by two long straight lines separated by a short line. In that case, the two corners produced by the short line will be unstable. In cases such as that, where a sequence of one or more lines do not contribute to a stable corner, we check whether the neighboring two lines can produce a stable corner. If so, we make use of that.

We now have a method of deriving corner points from a group of line segments.

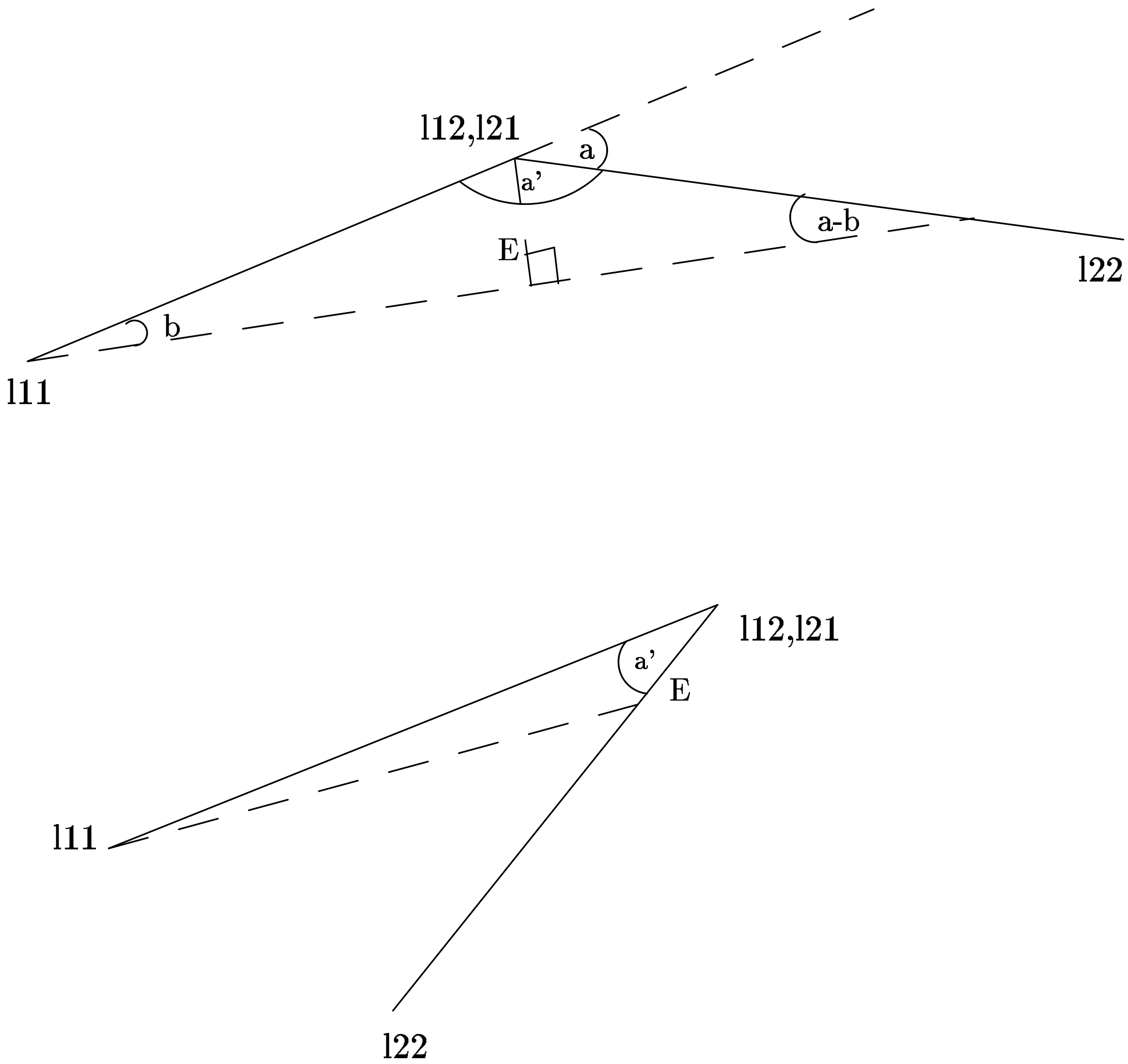


Figure 6.11: Two examples, for acute (below) and obtuse (above) angles, in which l_1 rotates so that it deviates from the original line by E pixels.

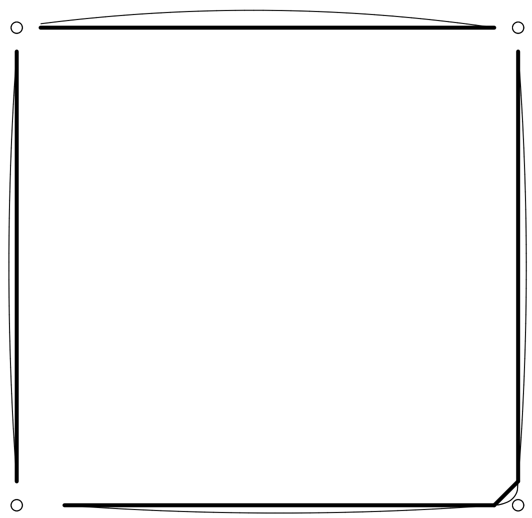


Figure 6.12: In the group shown above, a rounded corner's approximation contains a small line at the corner. If this line does not contribute to a stable corner, we may ignore it and form a corner from its neighboring lines. Edges are shown with a lighter weight, line approximations are shown in heavy weight. Circles show where we would like to find corners.

Since this is the only information about a group that we actually use for indexing, there is no distinction to be made between two groups that produce the same corners, so we once again remove duplicate groups and groups that are subsets of other groups, this time making these judgements based on the groups' points, not their line segments. To facilitate this process, we will also merge together points that are quite close together.

6.6 Ordering Groups by Saliency

After detecting groups in an image, we wish to use their salience to order our search for an object. There are two factors of which we would like to take account, but so far we have addressed only one of these. What we have *not* done is to look at the factors that make two groups seem particularly likely to come from the same object. It is necessary to form pairs of groups because a single group does not typically provide enough points for indexing. There are plenty of clues available to tell us which pairs of groups are especially likely to lead to successful recognition. For example, 3-D objects often produce pairs of groups that share a line. Other clues are explored in Jacobs[60, 59]. But we have not had a chance to explore the use of these clues in this thesis.

Since we have not worked on deciding which pairs of groups go particularly well together, we decide which groups are best individually, and order our search for an object by starting with the best groups. We have already defined the saliency of groups, and we use this salience to order them, with one caveat. In spite of our precautions, it often happens that the same lines participate in several similar salient groups. We loop through all the convex groups, in order of their saliency. If none of the lines in a group appear in a previously selected group, we select that group now. A line has a particular orientation in a group, so we do allow one side of the line to appear in one group, and the other side to appear in another group. On the first pass, then, we collect together the most salient groups that come from different regions of the image. We then repeat this process as many times as we like.

6.7 The Overall System

The most important thing about our grouping system, of course, is not its efficiency but whether it produces useful groups. There are several ways to judge this. For our indexing system, the important thing is that the grouping system finds some groups repeatedly in different pictures of an object. In chapter 7 we will show experiments that measure under what circumstances our grouping system is adequate for recognition. But we would also like to get a sense of how useful our convex grouping system might be as a starting point for further work on grouping. For example, we know that our method of locating point features is rather simple, and we would like to know whether the convex groups that we form might be a good basis for a better point finder. And we would like to know whether adding constraints to our system might winnow out some spurious groups. One way to judge the potential of our current system is to just see some examples of the output that it produces.

Figures 6.13 and 6.14 show the most salient groups found by the grouping system on an isolated telephone. Many of the groups found here show up reliably in other pictures of the phone, taken from slightly different viewpoints or distances.

In figures 6.16 and 6.17 we see some groups found in the scene shown in figure 6.15. Almost all the telephone's convex groups are at least partially occluded in this picture. However, we find unoccluded portions of these groups, many useful groups from the stapler, and some of the salient structure of the mugs. Figures 6.19, 6.20, and 6.21 show groups found in a similar scene, which is shown in figure 6.18.

Figures 6.23 and 6.24 show the results on a different scene, shown in figure 6.22, which was taken at the CMU calibrated image lab. Although the edges are noisy and hard for a person to interpret, we can see that the system finds much of the rectangular structure inherent in the buildings in the scene.

In each of the pictures shown, many of the most salient groups come entirely from

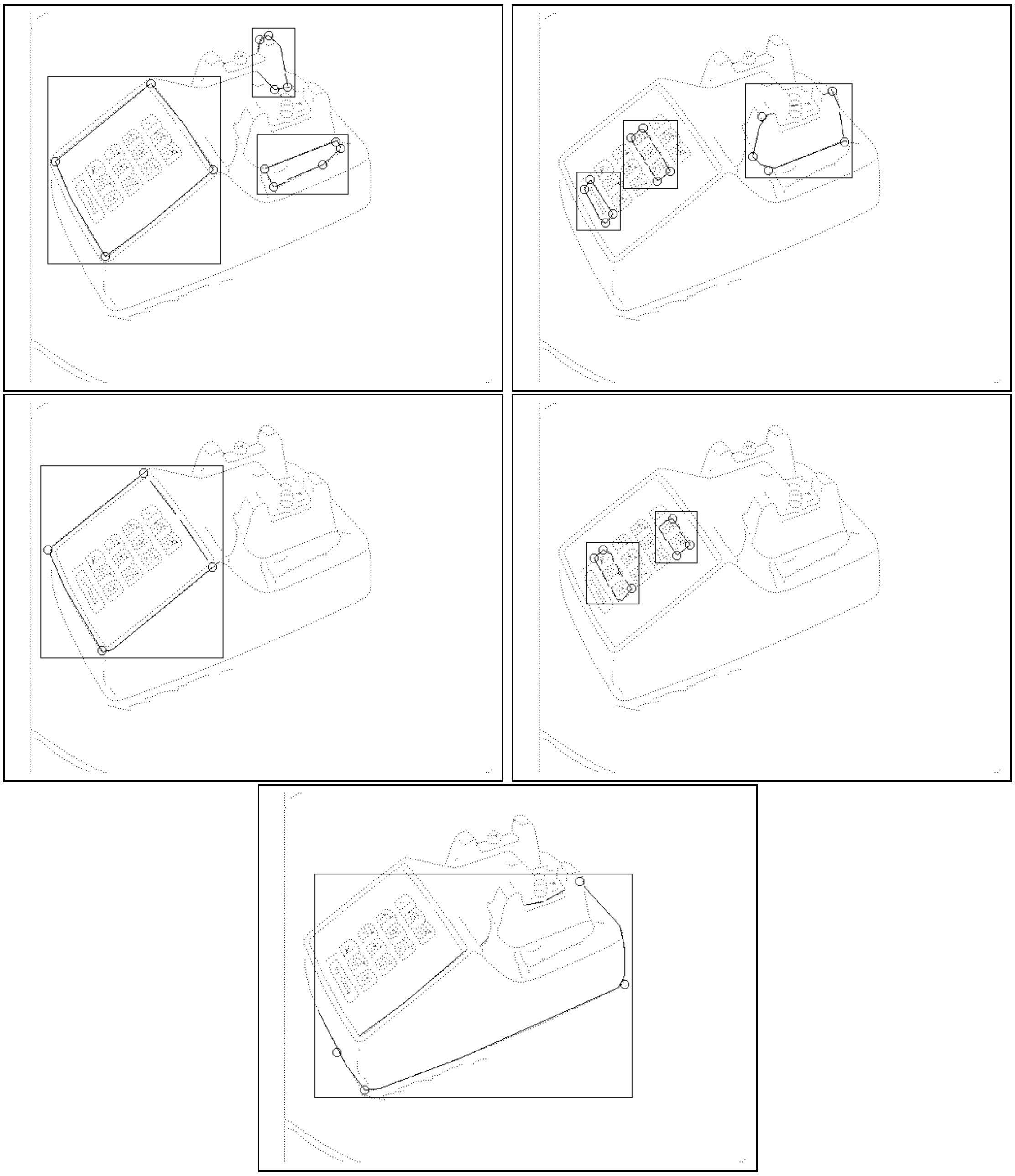


Figure 6.13: This shows the most salient groups in an image of a telephone. These groups have a salience fraction of at least .75, and each group contains line segments that do not appear with the same orientation in any more salient group. The dotted lines show the edges of the image. There is a box around each separate group. Solid lines show the lines that form the group. Circles show the corners found in the group.

Figure 6.14: This shows the second pass through the convex groups. The lines segments in each group may have appeared in one previously selected group, but not in two.

Figure 6.15: A scene with the telephone and some other objects.

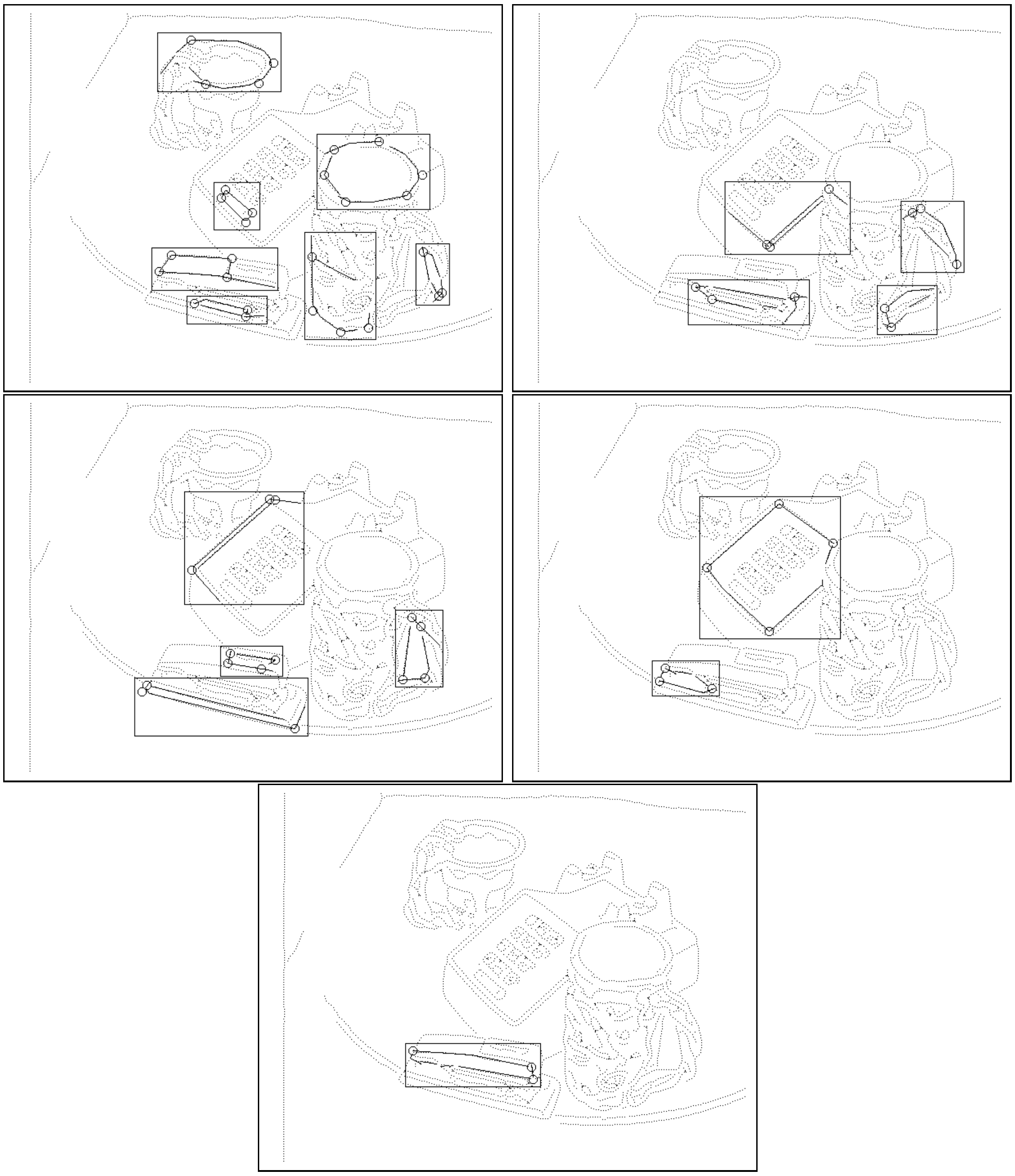


Figure 6.16: Similarly, the most salient groups found in a scene.

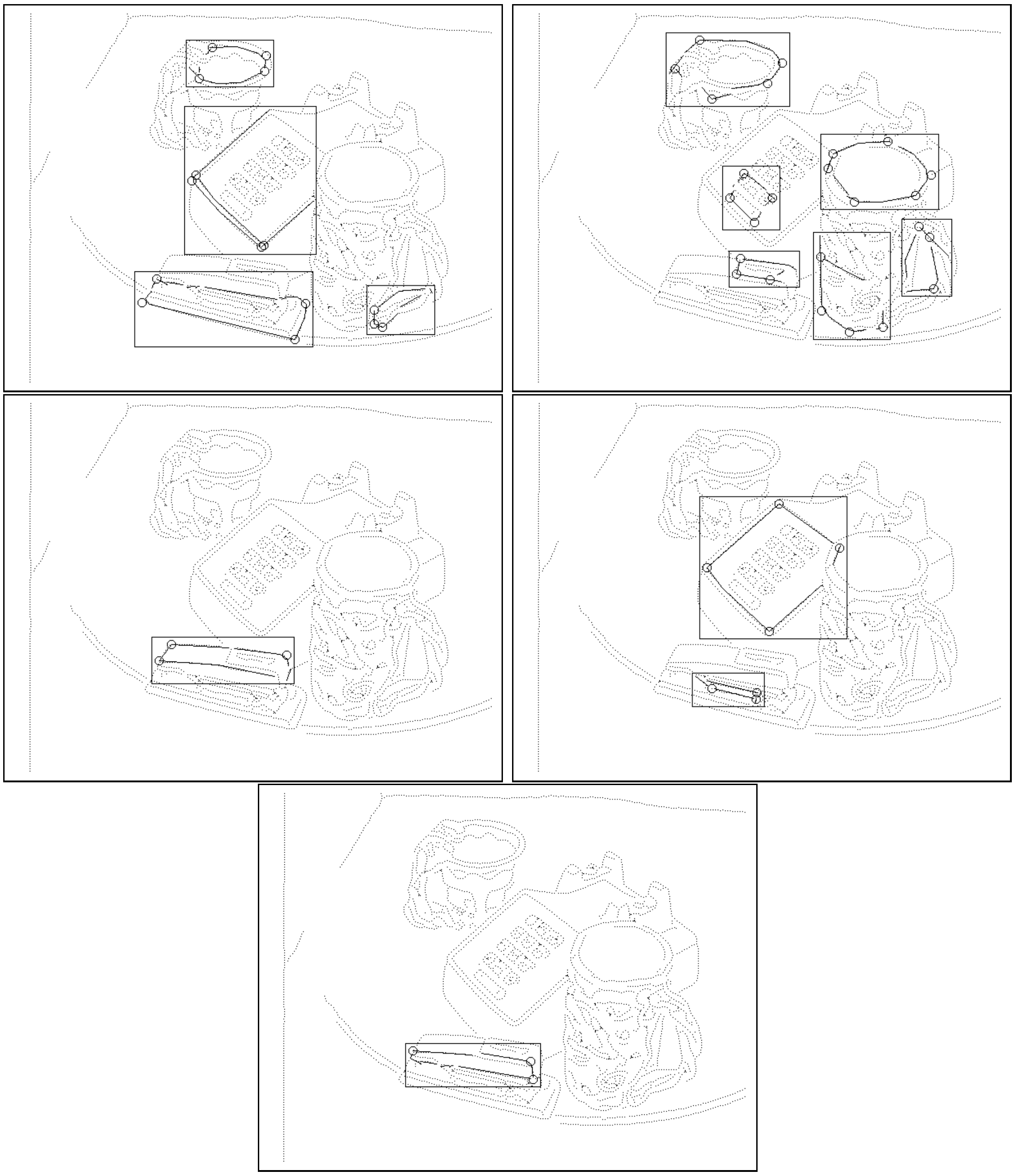


Figure 6.17: The second most salient groups found.

Figure 6.18: Photograph of another scene.

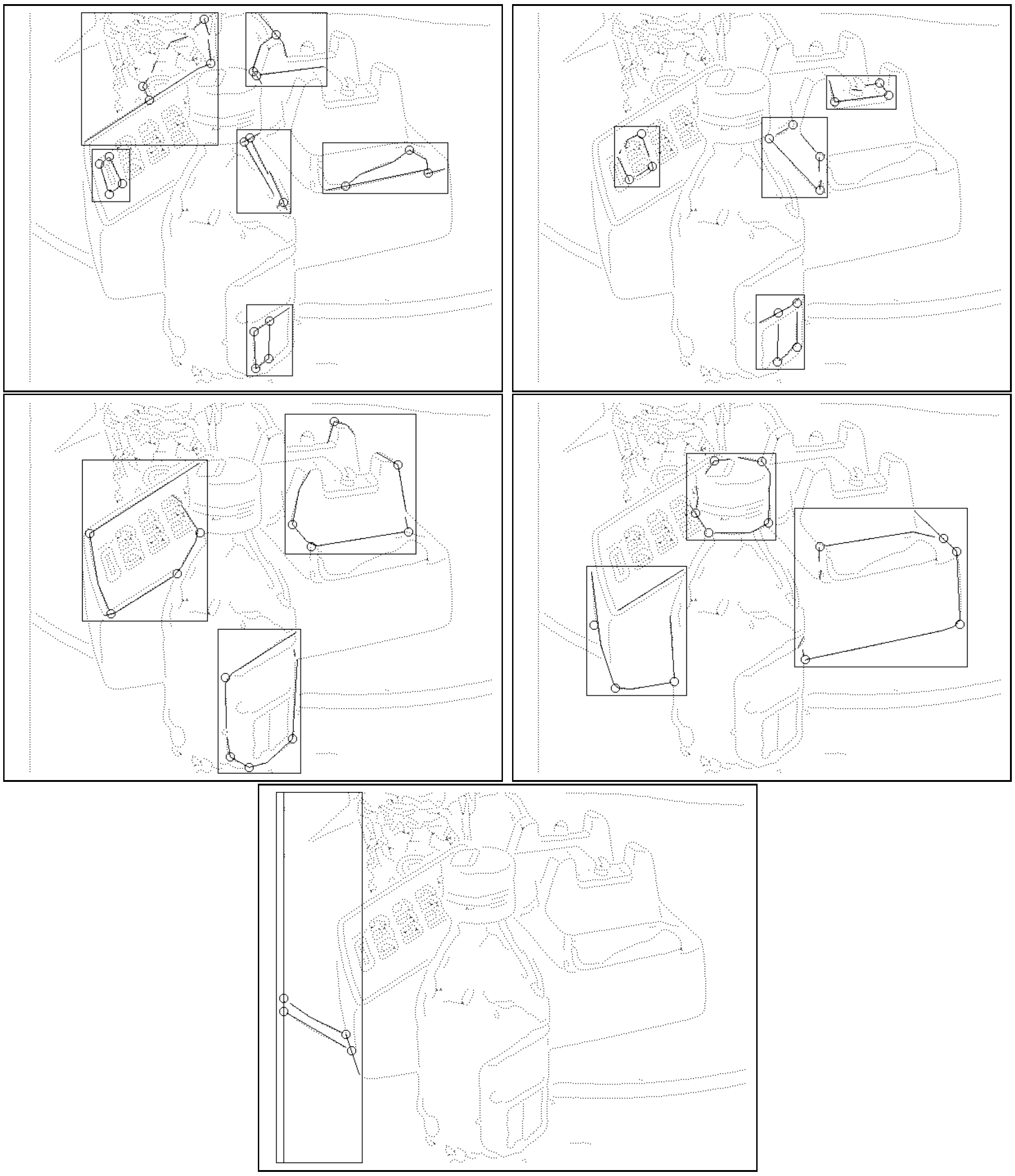


Figure 6.19: This shows the most salient groups in an image of a telephone. These groups have a salience fraction of at least .75, and each group contains line segments that do not appear with the same orientation in any more salient group. The dotted lines show the edges of the image. There is a box around each separate group. Solid lines show the lines that form the group. Circles show the corners found in the group.

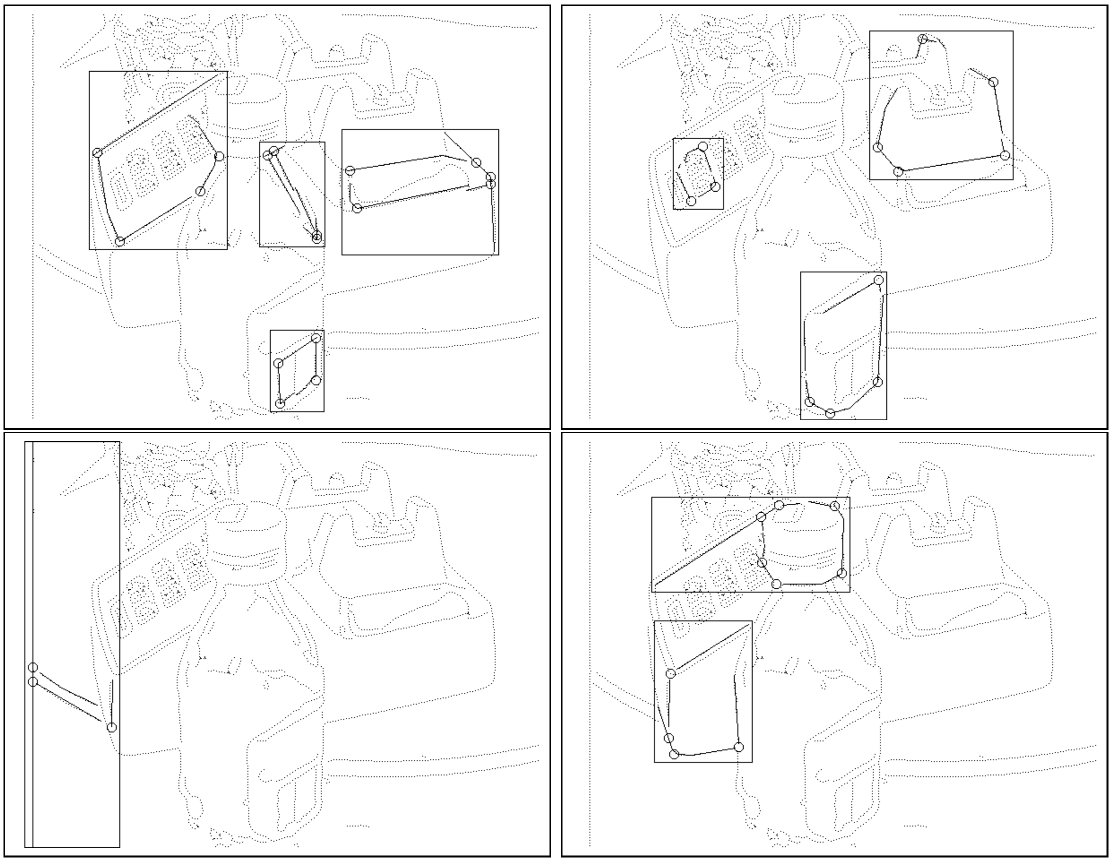


Figure 6.20: This shows the second pass through the convex groups. The lines segments in each group may have appeared in one previously selected group, but not in two.

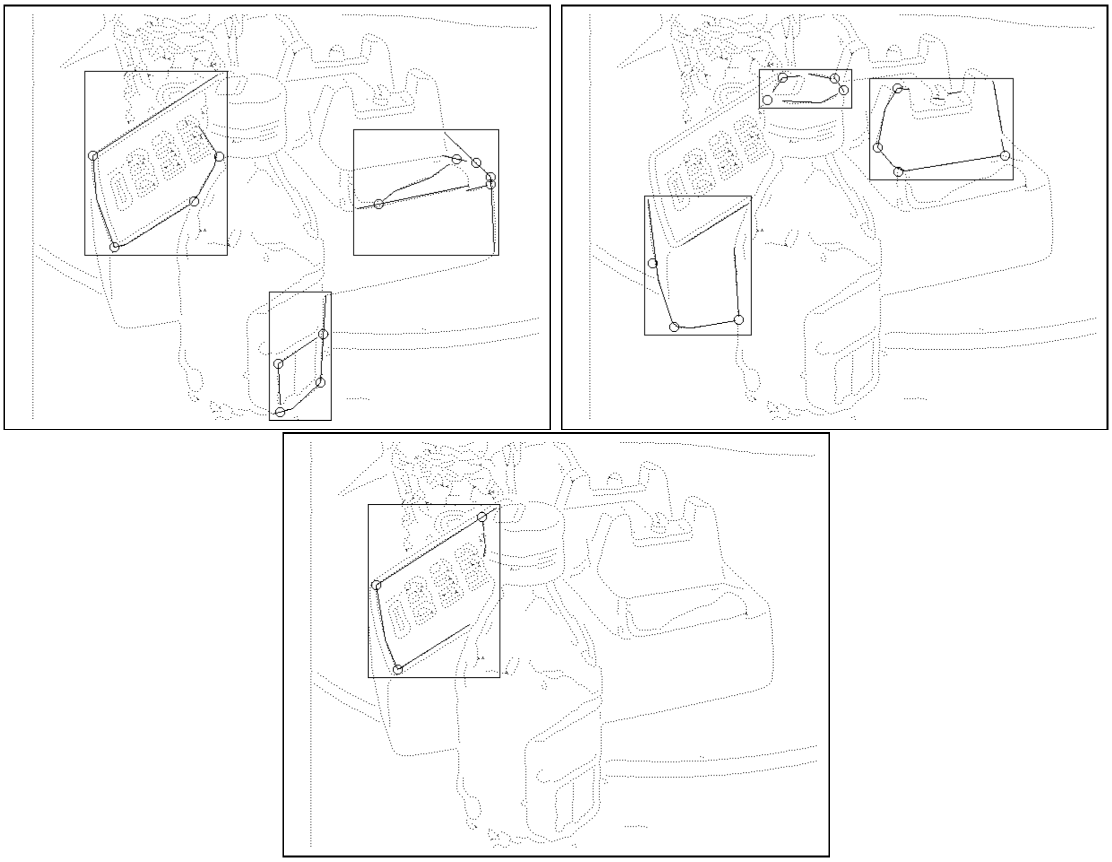


Figure 6.21: This shows the third pass through the convex groups.

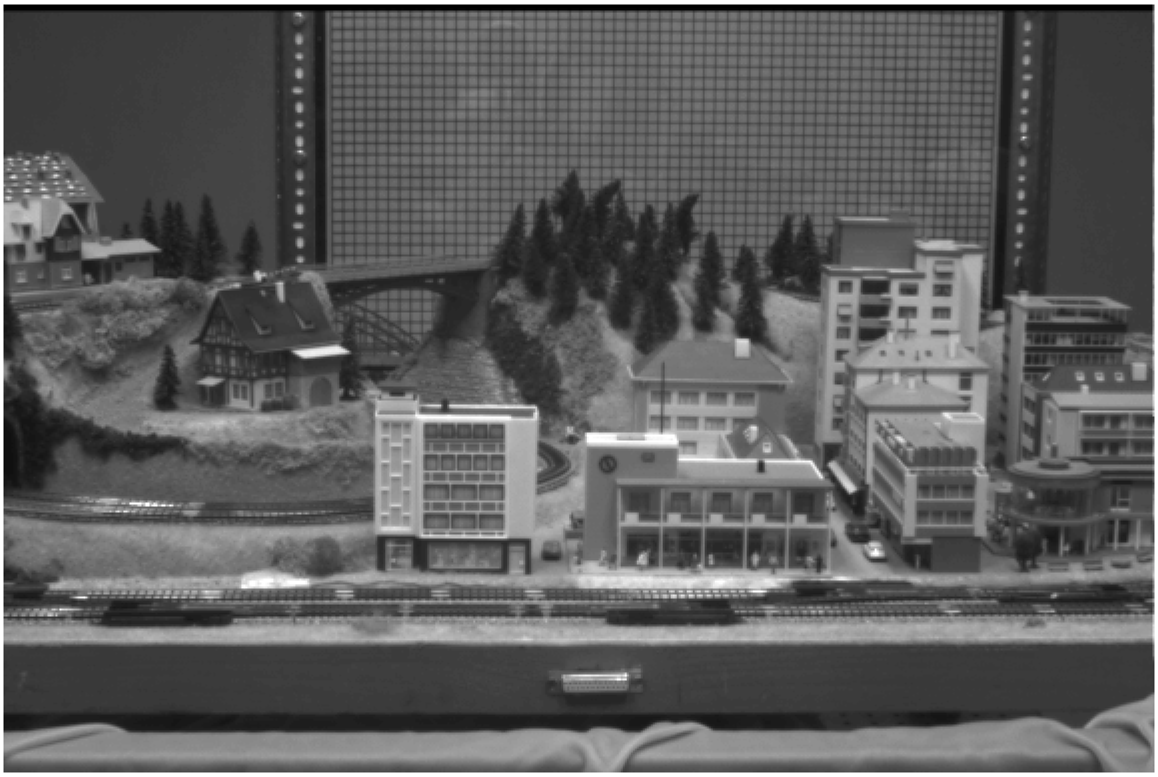


Figure 6.22: A picture from the CMU calibration lab, which was randomly selected from David Michael's directory.

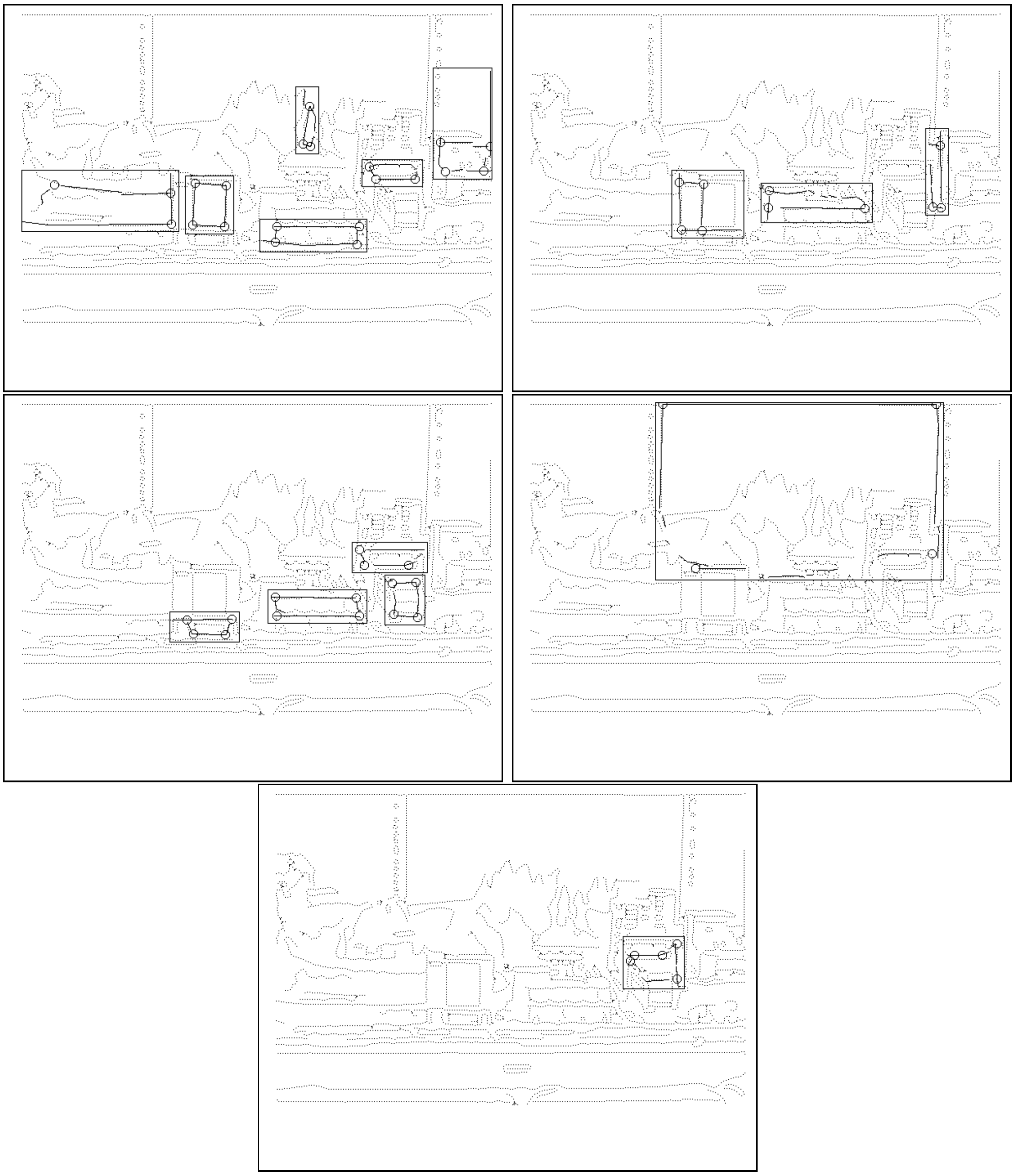


Figure 6.23: First pass through the CMU picture.

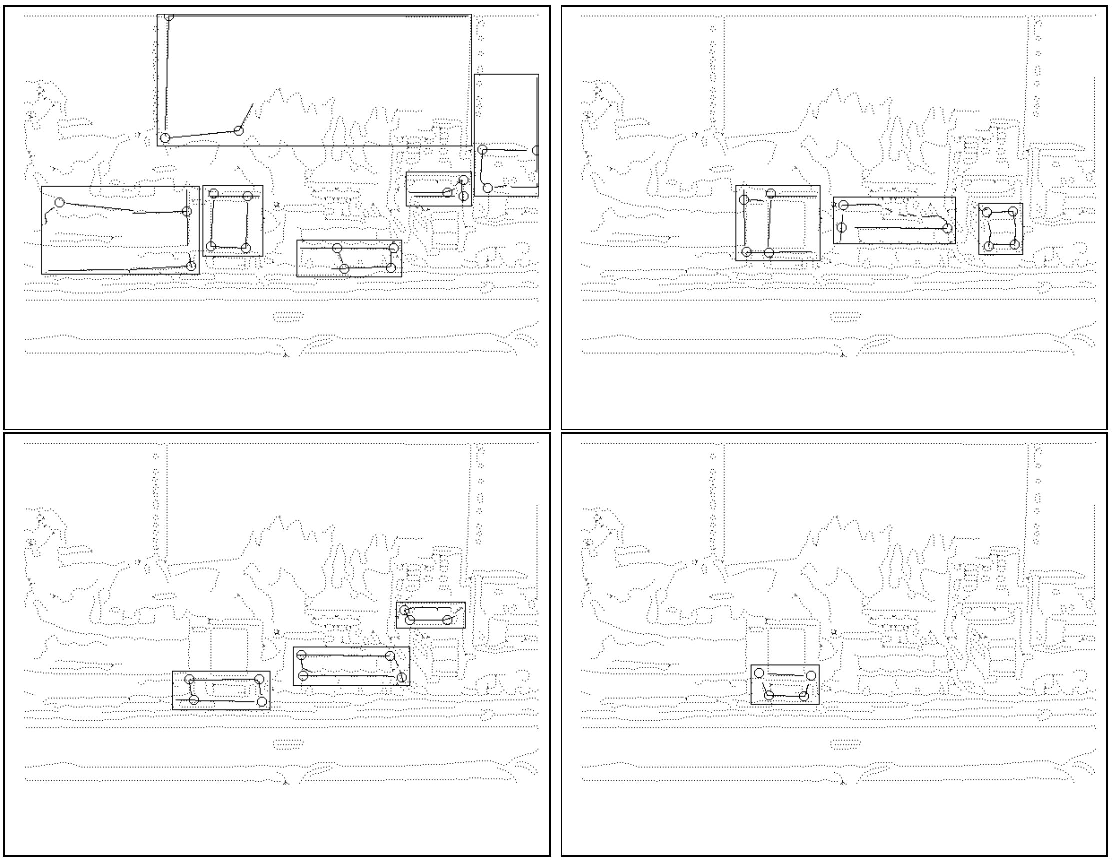


Figure 6.24: Second pass through the CMU picture.

the convex structure of a single object, making them potentially useful for recognition. We also see many remaining challenges to grouping, because many of the groups found either do not appear perceptually salient, or appear to combine either lines from two different objects, or to combine strong lines from one object with noisy or unstable lines.

6.8 Conclusions

Convexity is just one potential grouping clue, but it is an important one to understand thoroughly. Objects often contain at least some convex parts, especially in two important application areas, recognition of buildings and of manufactured objects. For such objects convexity has often been used effectively to assist recognition, or other matching problems. But convexity has usually been handled in ad-hoc ways that are sensitive to local perturbations of the image.

This chapter shows that a simple, global salience measure can be used to efficiently find convex parts of an image. A global definition of our output has the strong advantage of allowing us to anticipate our output, independent of unrelated context. However, local methods have been used previously because global methods appear inefficient. We show here that much of the global constraint provided by salient convexity can be converted into a form in which it can be applied at each step of the search, and that this allows us to build an efficient system.

We demonstrate the system's efficiency both empirically and theoretically. Our analysis provides a quantitative understanding of when our system will be effective, with both theory and practice leading to the same basic conclusions. We also see under what circumstances salient convex grouping itself will be useful.

In addition, we draw attention to some important problems in bridging the gap between grouping and indexing. When dealing with real images, one must avoid creating spurious point features that are sensitive to noise. This is particularly important when these features will be used for indexing, because we must assume that all, or most of the features found in a group actually come from the object for which we are looking. We show how to estimate the instability of features from basic assumptions about the error in our edge and line detection, and that this makes the output of our grouping system much more robust.

We also show that it is important to recognition to more carefully determine both the salience of a particular group, and the relative salience of pairs of groups. We refer the reader to Jacobs[60, 59], however, for a more thorough treatment of this topic.

In chapter 7 we will examine the contribution that this grouping system can make to a complete recognition system. Grouping will reduce the combinatorics of recognition by focusing our search for an object on subsets of the image that are likely to all come from a single object, and by providing us with a canonical ordering of the features in a group. It is not necessary that every group of lines that we find in the image actually comes from the object for which we are looking. It is sufficient if we can locate enough image groups to allow us to recognize an object without having to consider too many image groups, that is, our groups need to provide points that are more likely to come from the object for which we are looking than are randomly selected groups. The greater this likelihood is, the more grouping will help us.

Some of the motivation for using salient convexity as a grouping clue are given in work that we reference. Some of the motivation is the obvious fact that objects often have convex parts that frequently produce convex edges in the image. But our theoretical analysis also helps us to see when salient convexity will be useful. If a random process is unlikely to produce many salient convex groups, than the groups that we do find will be likely to reflect some of the underlying structure of the scene.

Chapter 7

A Recognition System

7.1 Linking the Pieces Together

We have now described the main components of a recognition system. We have shown how to form groups of image points, how to use these groups to index into a data base, matching them to geometrically consistent groups of model points, and how to use these matches to generate hypothetical poses of additional model features. In this chapter we link these pieces together to see how they interact. This will give us some idea of the potential value to a complete recognition system of both our indexing and our grouping system. It will also point out areas where further study is required. We begin by describing how we combine the modules that we have developed into a complete system. In the course of this description, we will mention a number of thresholds that are used. We give the values of these thresholds together, at the end of the description.

In the preprocessing stage we must represent groups of model points in a lookup table. We do this with the following steps. First we take a series of photographs of an isolated object. Then we run our grouping system on each photograph. We look at the most salient groups found in each image, along with their point features, to determine, by hand, which groups are found consistently. This step is somewhat subjective, although it could be automated if we had a CAD model of the object, and knew the viewing direction of each picture. Then, by hand, we match the points that these groups produce between all the images. We now have a list of groups of model points, and for each group we have the location of the points in a number of images. We form some additional groups from subsets of these groups. If a group produces at least four point features, we may form new groups of points by removing one of the points in the initial group. This will allow us to match that group if one of its points is not detected in the image. The choice of which subsets of groups to represent in our lookup table is also subjective, and based on a judgement of which groups are likely to be detected in an image with a missing point feature. These model groups typically produce three to five point features, which are not enough with which to perform indexing. So we form all pairs of these groups, giving us group-pairs that contain six or more points.

For each group-pair, there are only some orderings of the point features that we need to represent in our lookup table. First of all, we know the order of points around the perimeter of each convex group. We consider each point in the group a potential starting point in ordering the whole group, but then given a starting point, the order of the remaining points is determined by just proceeding clockwise about the group. If one of the groups in a group-pair has more point features, we put this group's points first, knowing that we will be able to impose the same ordering at run time. If the groups have the same number of point features we must consider putting either group first. So if the two groups have n_1 and n_2 points, the total number of possible orderings of these points is $n_1 n_2$ if $n_1 \neq n_2$, and is $2n_1 n_2$ if $n_1 = n_2$. Thus we see that grouping allows us to reduce the total number of possible orderings significantly; without grouping there would be $(n_1 + n_2)!$ orderings to consider. For any ordering of the points, the first three points are used as a basis for computing the affine coordinates of the remaining points, with the second point used as the origin of this basis. If we represent all of these orderings of each group-pair in our table, then we may perform indexing using any one of these orderings of a pair of image groups, and we are guaranteed to find the matching ordering of the model points in our table. In practice, in some of our experiments we explicitly represent in the table only one of the n_2 possible orderings of the second group in the pair, in order to save compile time and space. This requires us to perform indexing by considering all possible orderings of the points in the second group of a pair of image groups, and to combine the results. These two methods will produce the same output, because they each compare all matches between the image and the model points. While in a working recognition system we would not want to sacrifice run time efficiency for compile time efficiency and space savings, this can be a useful trade-off when experimenting with a system.

As described in chapter 4, given a series of images of each ordered set of points, we compute the affine coordinates of the points in each image, and then determine the lines in α and β space that correspond to this group-pair. We determine which cells these lines intersect in a discretized version of these affine spaces. The method of discretization is also described in chapter 4. In each intersected cell, we place a pointer to the group-pair. Accessing a cell, then, produces a list of model group-pairs that could produce an image with affine coordinates that fall somewhere in that cell. These steps produce two hash tables that represent the α and β spaces. Each cell in the two spaces that is not empty is represented in the appropriate table, hashed by its coordinates.

We must also represent a model's line segments during preprocessing, so that we may use them to verify hypotheses. To do this we choose by hand a small set of line segments that represent some of the boundary of the object, and that have endpoints that reliably appear in our images of the isolated model. This process could also be automated. In the tests below, we choose line segments whose endpoints all belong to the model groups that we have chosen. Chapter 4 describes how we use images of the endpoints of these line segments to derive lines in α and β space. We derive a different pair of lines for each triple of points that we have used as a basis for computing the affine coordinates of one of the group-pairs. So for every three points that we might use as a basis for computing the affine coordinates of image points for indexing, we have also used those points as a basis for representing the model's line segments. Therefore, whenever indexing produces a match between model and image points, we may use that match to determine the location of the endpoints of the model's line segments.

We may also represent more than one object in our indexing tables in just the same way that we represent a single object.

At run time, we begin by applying our grouping system to an image of a scene that contains the object that we seek. This provides us with a set of convex groups, along with a saliency fraction that measures the value of each group. We drop convex groups if the total length of their line segments falls below some threshold. There are then many different ways that we could order pairs of these groups for indexing. We choose a simple method that demonstrates some of the value of these groups. As described in chapter 6, we make one pass through the convex groups, picking the most salient ones subject to the constraint that each side of each line segment can appear in only one convex group. This typically produces between ten and twenty-five different convex groups for an image of moderate size. Then we form all pairs of these convex groups. We now have some freedom as to how to order the points in this group-pair. If the two convex groups each have the same number of points, we may put either one first, otherwise we put the group with the most points first. And we may pick any point as the starting point in the first convex group, which determines the three points that we will use as a basis. Of all the possible orderings available to us, we choose the one that seems to provide the most stable affine basis. As a simple way of judging the stability of an affine basis, we consider how the affine coordinates of a point described in that basis will change if we perturb them slightly. Then, if we have only made an entry for one ordering of the second convex groups in the index tables, we must perform lookups with all orderings of those points at run time. Note that when we change the ordering of points in the second convex group this will not affect the basis points, and so we only need to reorder the indices we use in the lookup, we do not need to recompute affine coordinates, or the effects of error. We perform indexing with each set of points, as described in chapter 4. This associates a list of matching sets of model points with each group-pair found in the image. We then order these group-pairs based on the number of matches found for each.

Beginning with the image group-pair that matches the fewest sets of model points, we generate hypotheses about the location of the model in the scene. When an image group-pair matches more than one model group-pair, we order these matches arbitrarily. For each match, we use the techniques described in chapter 4 to find the position of the endpoints of the model's line segments in the image implied by a least squares fit between the model and image points that indexing has matched. We then use these line segments to get an overall idea of the value of a hypothesis. First, we only consider a hypothesis if it results in projected model line segments whose cumulative length is above some threshold. This guards against the possibility that a match will cause the model to project to a very small area of the image, where most of its edges could be matched to texture or noise. We then match each model line segment to an image line segment if the image segment is completely within some fixed distance of the projected model segment, and if the angles of the two line segments are similar. More than one image segment may be matched to a model segment, but the total length of the matching image segments cannot exceed the length of the matching model segment. We then divide the length of the image segments that we have matched to the model by the length of the projected model segments, determining what fraction of the model we have matched. In the experiments below, we have examined hypotheses for which this fraction was above some threshold.

This method of verifying an hypothesis is designed to be an easy method of deciding whether we have the right match. It could certainly be improved. Most importantly, in verification, and also in indexing, we have not taken advantage of the fact that not all features of the model are visible from all viewpoints. In indexing, this means that we assume that the model points come from a wire-frame model, and we may match image points to model points with the implicit assumption of a viewpoint from which the model points would not actually be visible. In verification, this means that we make no effort to perform hidden line elimination. This can result in hypotheses that produce impossible projections of the model. Our goal, however, has been to demonstrate just the essentials of a recognition system.

The system that we have described requires us to choose a number of thresholds, which we have mentioned throughout the text. We summarize these choices here. We used a single set of values for these thresholds in all the experiments we describe in this chapter. In running the Canny edge detector, we used a σ of two for Gaussian smoothing. In the split-and-merge algorithm, which we describe in chapter 6, we approximated edges with line segments such that the edges were all within no more than three pixels of the line segments. When grouping, we used a saliency fraction of .75. In chapter 6 we show why that is a good choice in terms of efficiency and of the size of the output. Some thresholds are also used when determining whether two lines are nearly collinear, for in that case we allow a slight concavity in the convex groups. However, we do not discuss our method of judging near-collinearity. When determining the stability of corner features, we assume that the endpoints of line segments are allowed to vary by five pixels. The variation in the relative position of two endpoints is additionally limited to be at most 10% of the distance between them. When determining the stability of corners formed by pairs of connected line segments, we assume that the underlying curve may differ by up to six pixels from the approximating line segments. We compute the possible variation in the location of a corner due to these errors, and only make use of corner features whose location can vary by fifteen pixels or less. If two or more corners are within two pixels of each other, we compress them into a single corner, located at their average. This allows us to eliminate groups of points that are nearly identical. We only use groups for indexing if they contain at least three point features, and if the sum of the length of their line segments exceeds one hundred pixels. In indexing, we divide each dimension of the index space into fifty parts. These intervals are not uniform, and are described in chapter 4. We only represent sections of affine space between twenty-five and minus twenty-five. When indexing, we allow for an error of five pixels in the location of point features. When performing verification we require a projected model's lines to have a collective length of at least one hundred pixels. We match a model line to an image line if the entire image line is within ten pixels of the model line, and if their angles differ by no more than $\frac{\pi}{10}$. Although a significant number of thresholds are used in the entire system, the core components contain few important thresholds. The basic grouping system has just one threshold, the salience fraction, and we have shown both analytically and experimentally how this may be chosen. Several thresholds are used in building the indexing table; these determine the accuracy with which we will represent affine space. And a single threshold is used when indexing to measure our uncertainty about the location of features.

7.2 Experiments

We have run some experiments with this system to provide the reader with an idea of the kinds of images that it can handle. Our main goals are to provide examples of when the grouping system will be sufficient to help recognize an object, to show that the indexing system provides correct matches when the grouping system finds correct image groups, and to provide some idea of the overall reduction in search that grouping and indexing can provide. We also want to give an example of the kind of interactions that can occur between grouping and indexing. Finally, we want to see where the overall system breaks down. This can help tell us which aspects of this system are attractive candidates for additional work.

In these experiments the system recognizes a telephone. This is an object that contains many significant convex parts. On the other hand, the telephone that we use is still quite challenging to our system because it contains many curved edges as well. Some of these curves are gentle, for example the corners of the phone are generally rounded. But even these gentle curves can cause unstable corner points.

Figure 7.1 shows edges found in two images of the isolated telephone used to build a model of the phone. Circles in these images show corner features that appear in salient convex groups. The ones that are used in the model are numbered for reference. Figures 6.13 and 6.14 in chapter 6 show some of the more salient convex groups found in one of these images. Figure 7.2 shows another example of these groups, including some of the groups used in building the model. We only model the portion of the telephone that can be seen from the front, right side, to alleviate problems caused by the lack of hidden line elimination, and to simplify model building. We form groups whose point features have the following indices: (14 15 16 18) (13 3 35 36) (31 32 33 34) (11 17 19 20 21) (11 17 19 21) (11 17 19 20) (11 17 20 21) (11 19 20 21) (17 19 20 21) (9 10 17 11 22) (9 10 17 11) (9 10 17 22) (9 10 11 22) (9 17 11 22) (10 17 11 22) (9 10 11) (10 17 11) (10 17 22) (9 10 17) (0 1 11 22) (0 17 11 22) (1 13 12) (0 1 2 3 4 22) (0 1 2 3 4) (1 2 3) (41 42 43) (19 50 51 20) (20 50 51) (19 50 51). For verification, we represent the model with line segments that connect the points: (0 1) (1 2) (2 3) (3 4) (0 9) (9 10) (9 11) (11 17) (10 17) (3 13) (13 12) (12 4) (14 15) (15 16) (16 18) (18 14) (1 13) (10 12) (17 19). Examples of the projection of these line segments are shown later on, when we show examples of the system running. These segments only describe part of the phone's boundary; since our focus is not on accurate verification we have built only a simple model of some of the phone's line segments, which serves to tell us when we have a reasonably good match.

To test the system, we have taken several series of photographs. In each series we begin with the isolated telephone, and add objects to the foreground and background to make the scene progressively more complex. This gives us an idea of when the system will work, and when it will break down. We begin by showing each scene, the edges found in it, and the correct answer, when it is also found. We will then describe more details of the algorithm's performance, and analyze its successes and failures more carefully.

For example, figure 7.3 shows a picture of the isolated telephone and the edges found in this image. Figure 7.4 shows the 83'rd hypothesis that the system generates about the location of the telephone in the scene, which is correct. In this figure, lines indicate the hypothesized location of model line segments. The edges found in the image are shown with dots. Figure 7.5 shows the same scene, with some objects added to the background. The figure also shows the edges found in this scene. Figure 7.6 shows the correct hypothesis, which was found for that image. In figure 7.7 we

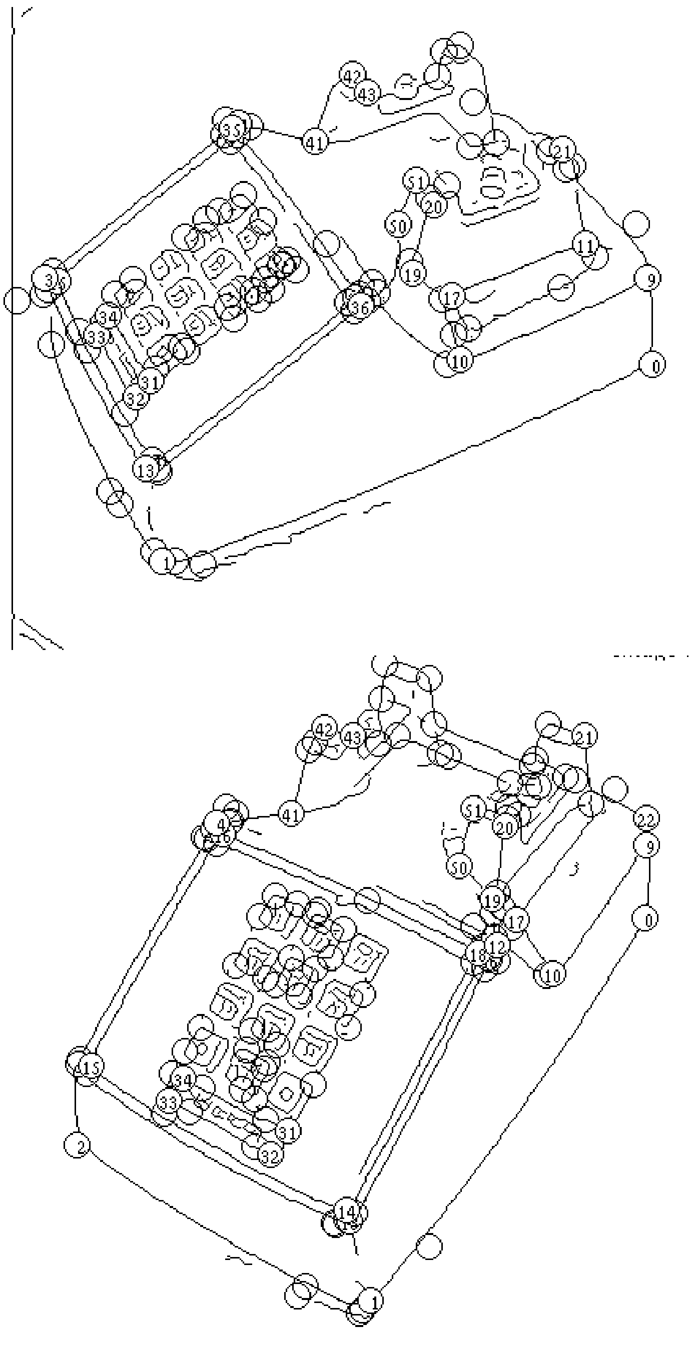


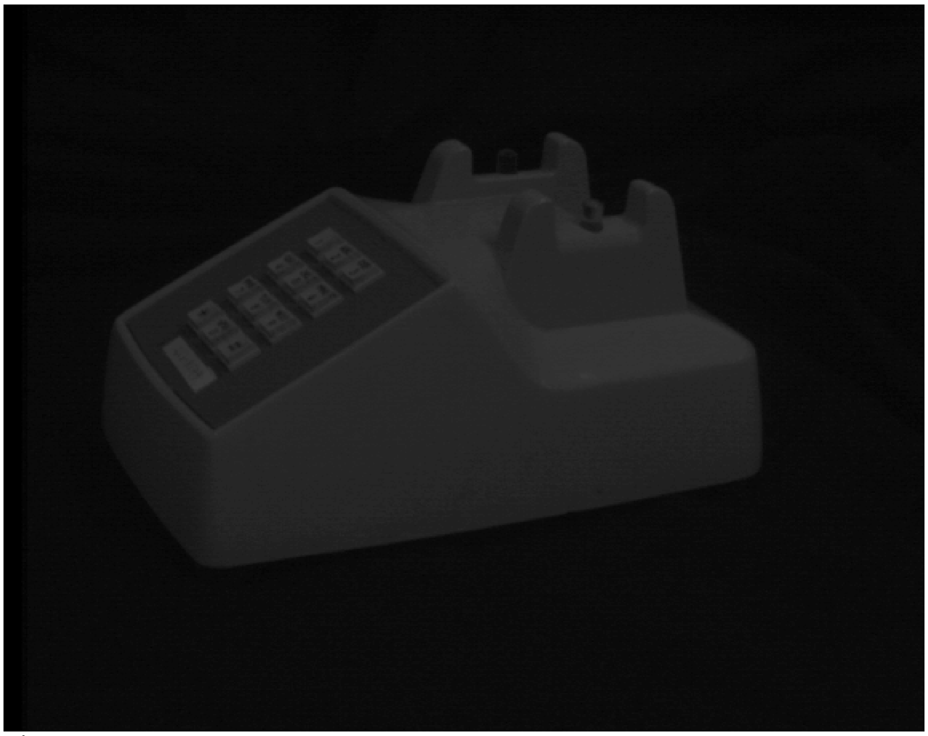
Figure 7.1: Edges from two of the images used to build models of the telephone. Circles show all the point features that appear in a convex group with saliency greater than .75. The numbered circles are the points that appear in groups that are actually used in our model of the telephone. Although numbers between 0 to 51 are used, there are only 29 numbered points.

Figure 7.2: This shows the most salient groups found in one of the images of the telephone. These groups have a saliency fraction of at least .75, and each group contains line segments that do not appear with the same orientation in any more salient group. Dotted lines connect all the lines that were used to form a convex group. There is a box around each separate group. Circles show the corners found in the group. This shows examples of groups formed using points with the indices: (14 15 16 18), (9 10 17 22), (31 32 33 34), (41 42 43), and (19 50 51 20). See figure 7.1 for a key to these corner numbers.

add an occluding object, in front of the telephone. The system again finds the correct hypothesis, which is shown in figure 7.8. It is interesting to note that this correct hypothesis is slightly different from the one found in the previous images; one less point is matched. This appears to be due to slight variations in the output of the edge detector, which cause one corner to disappear. In figure 7.9 we add another occluding object, to make the image a little more difficult. Again, figure 7.10 shows the correct answer. Figure 7.11 shows an additional occlusion, which causes the system to fail. In this image, one of the groups used to generate the previous correct hypotheses is partially occluded. Two more series of tests are shown in figures 7.12 through 7.22.

These tests give us a rough idea of the kinds of images on which the system will work. We see that our system can tolerate moderate amounts of occlusion, because many local groups are represented in the lookup table, and only two must be found in the image to make recognition possible. We would also like to get some idea of the speedup with which grouping and indexing can provide us for these images. We can determine this partly by recording the number of incorrect hypotheses that the system had to consider before reaching a correct hypothesis. The system correctly recognized the telephone in eight of the figures above, figures 7.3, 7.5, 7.7, 7.9, 7.12, 7.14, 7.17, and 7.19. In these images, the correct hypothesis was the 83'rd, 62'nd, 165'th, 80'th, 2nd, 168'th, 525'th, and 545'th hypothesis considered. In figure 7.13 we show a second correct hypothesis, which was the 8th one found. These figures show that grouping and indexing together can reduce the amount of costly verification required to a small amount. For comparison, consider what might happen if we used simple alignment without grouping or indexing. The images shown produce hundreds of point features, of which perhaps ten or twenty might actually come from point features in the model. Therefore we would expect to have to search through thousands of triples of image features before finding three that could all match points in our model. For each of these triples of image points we would have to consider all triples of model points. Since we use about thirty model points, we would have to match each triple of image points to tens of thousands of triples of model points. Our total expected search, then, before finding a correct match could be in the tens of millions. Our experiments also show that the amount of work required tends to grow with the complexity of the scene, but again, more slowly than would a simple alignment system.

Although we already have a system that can recognize objects in moderately complex scenes, in some ways this is still a preliminary system. It is therefore particularly important to understand problems that may exist in the current system, and how we might work to overcome them. We will mention three difficulties. First, and most importantly, we ask why the system fails in some cases to recognize objects. These failures always occur because the grouping system has not located more than one convex group in the image that it can match to the model. Second, we examine



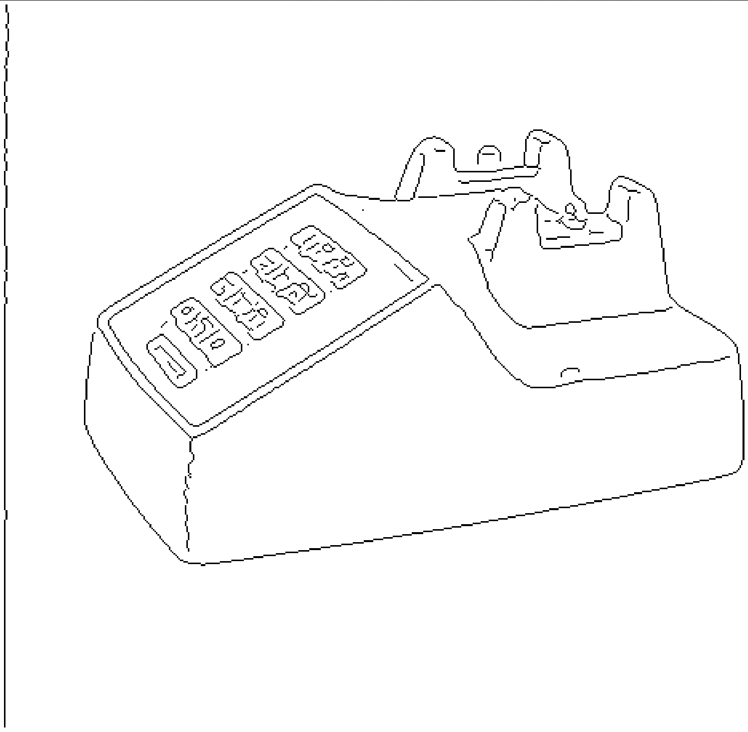


Figure 7.3: On top, an isolated picture of the telephone, from the first series of recognition tests performed. Below, the edges found in the image.

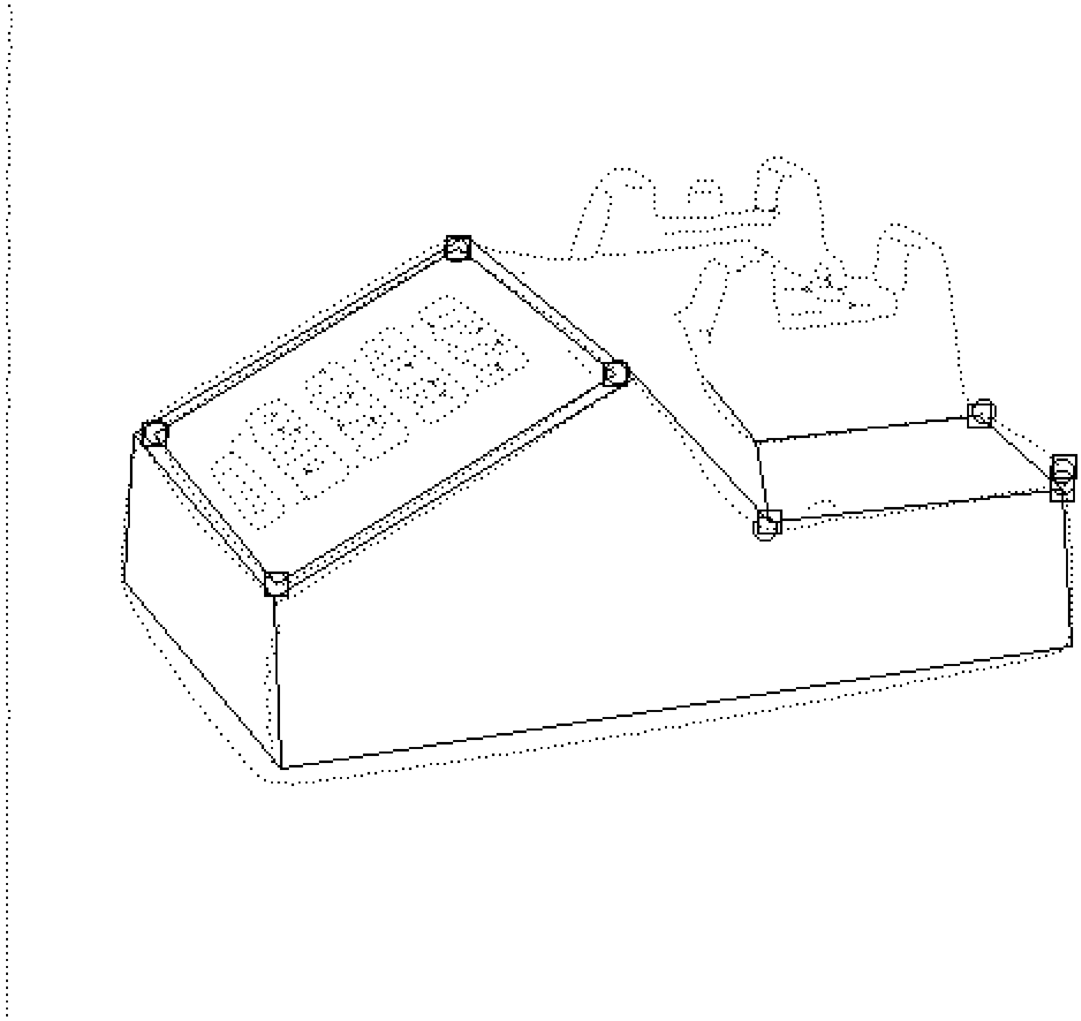


Figure 7.4: This shows the correct hypothesis, which the system found, for the image shown in the previous figure. Lines, which indicate the hypothetical location of model lines, are shown superimposed over a dotted edge map of the image. Circles indicate the location of image points that were used for indexing. Squares show the hypothesized location of the corresponding model points.

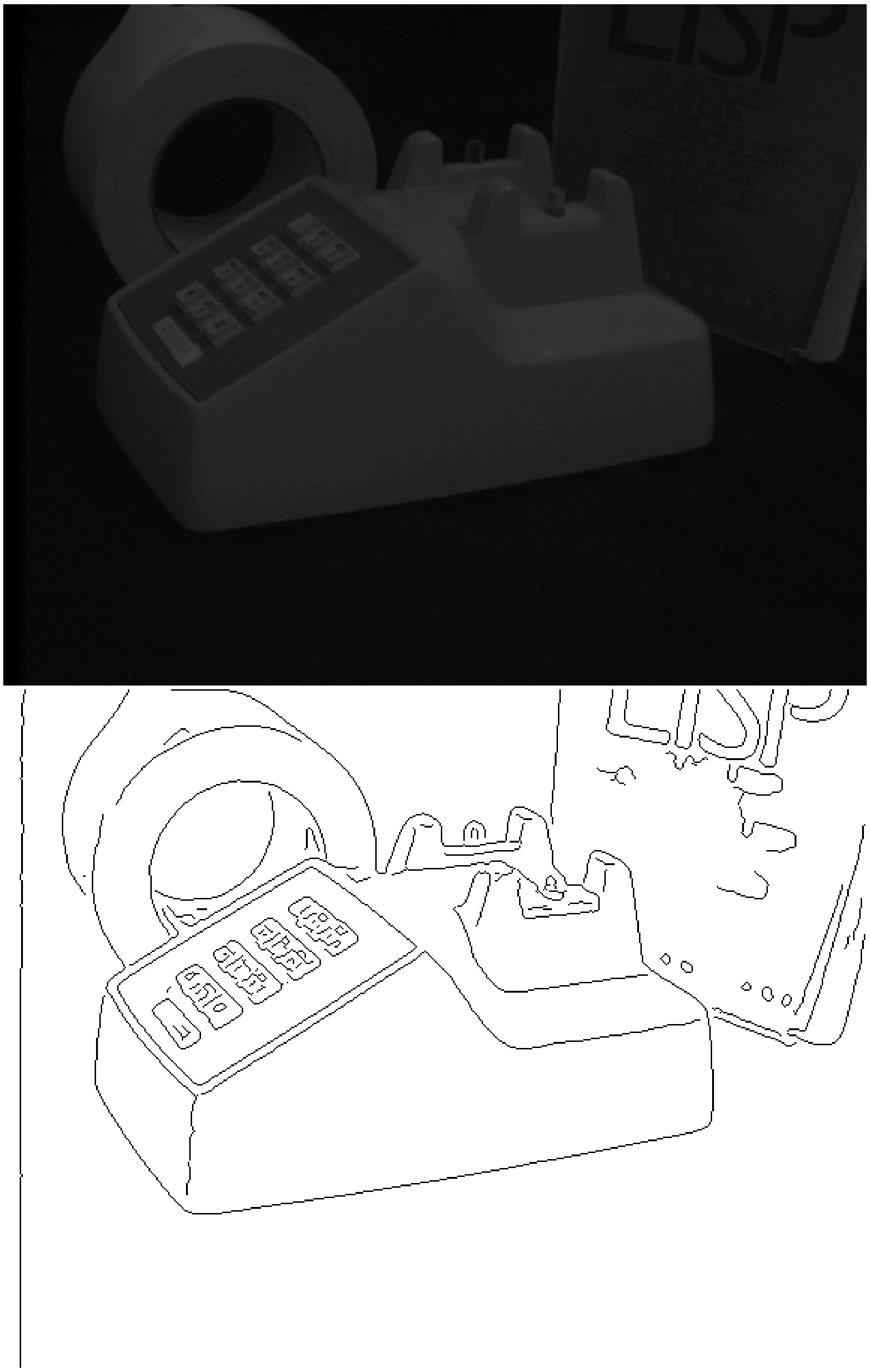


Figure 7.5: On top, a picture of the telephone with some other objects in the background, from the first series of recognition tests performed. Below, the edges found in the image.

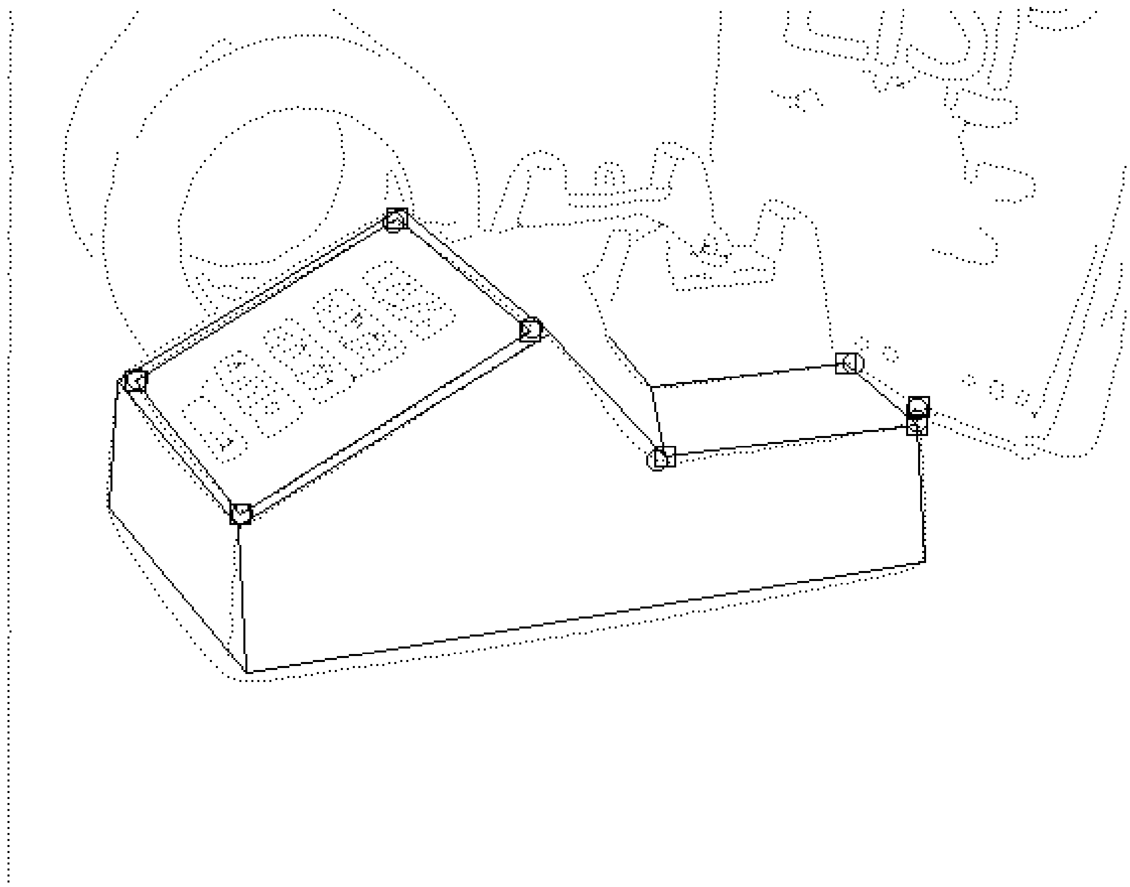


Figure 7.6: This shows the correct hypothesis, which the system found, for the image shown in the previous figure.

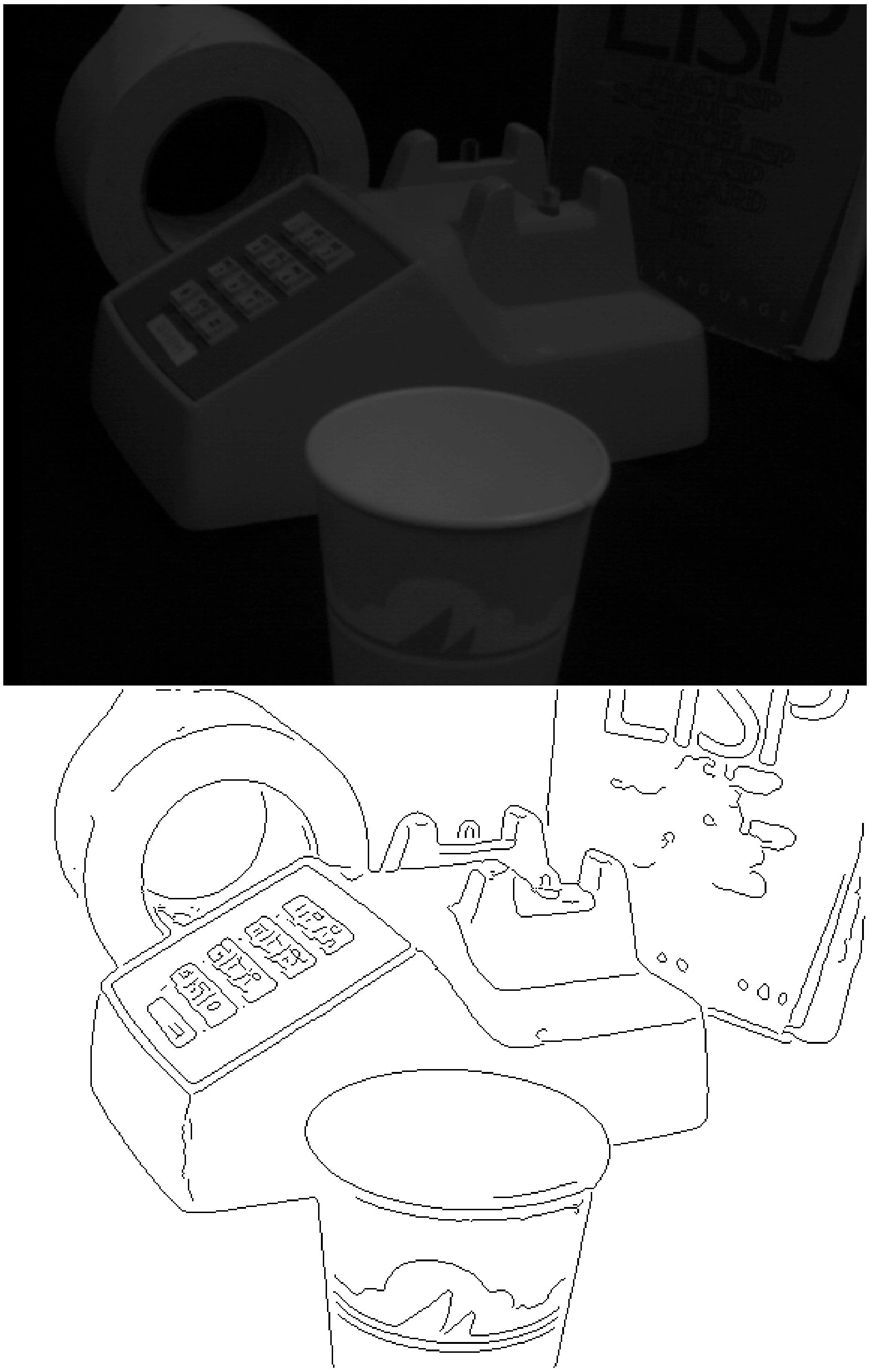


Figure 7.7: On top, a picture of the telephone in which some occlusion is added, from the first series of recognition tests performed. Below, the edges found in the image.

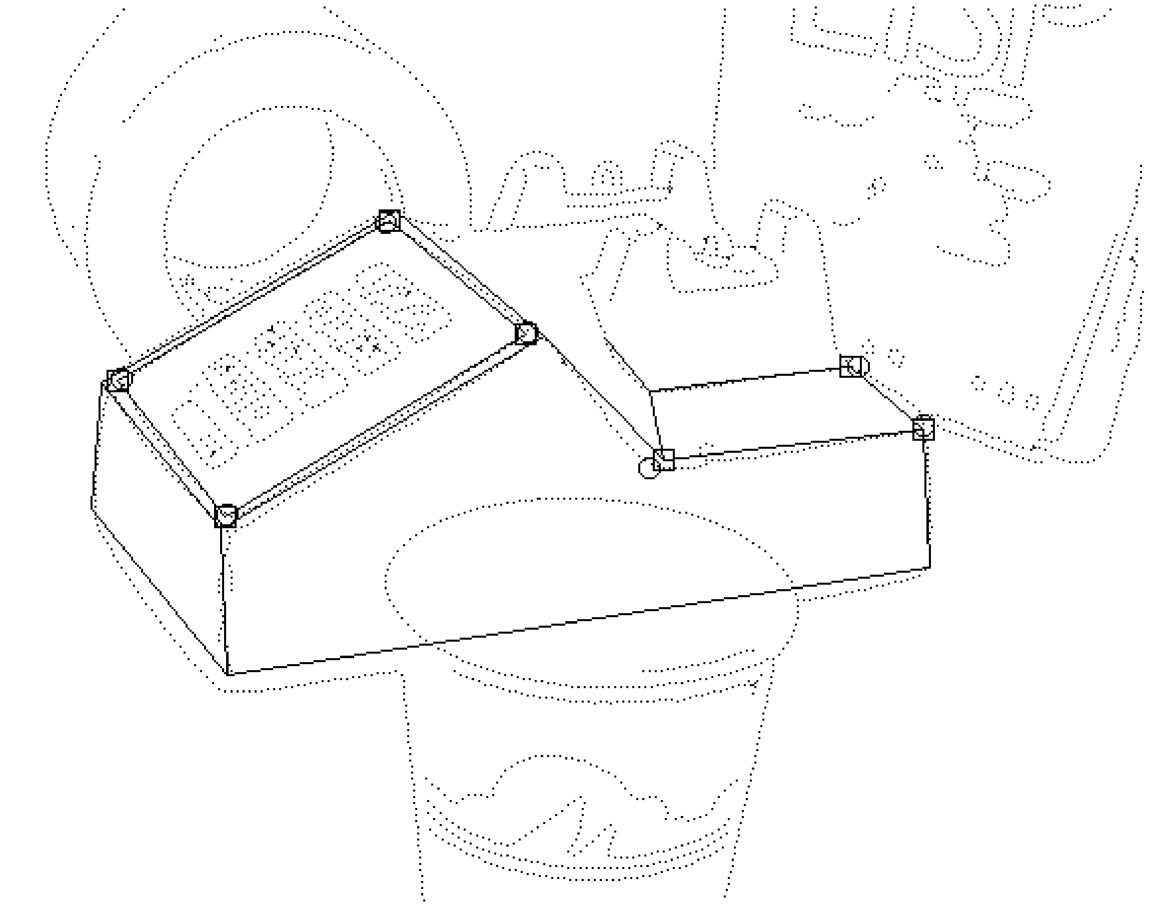


Figure 7.8: This shows the correct hypothesis, which the system found, for the image shown in the previous figure.

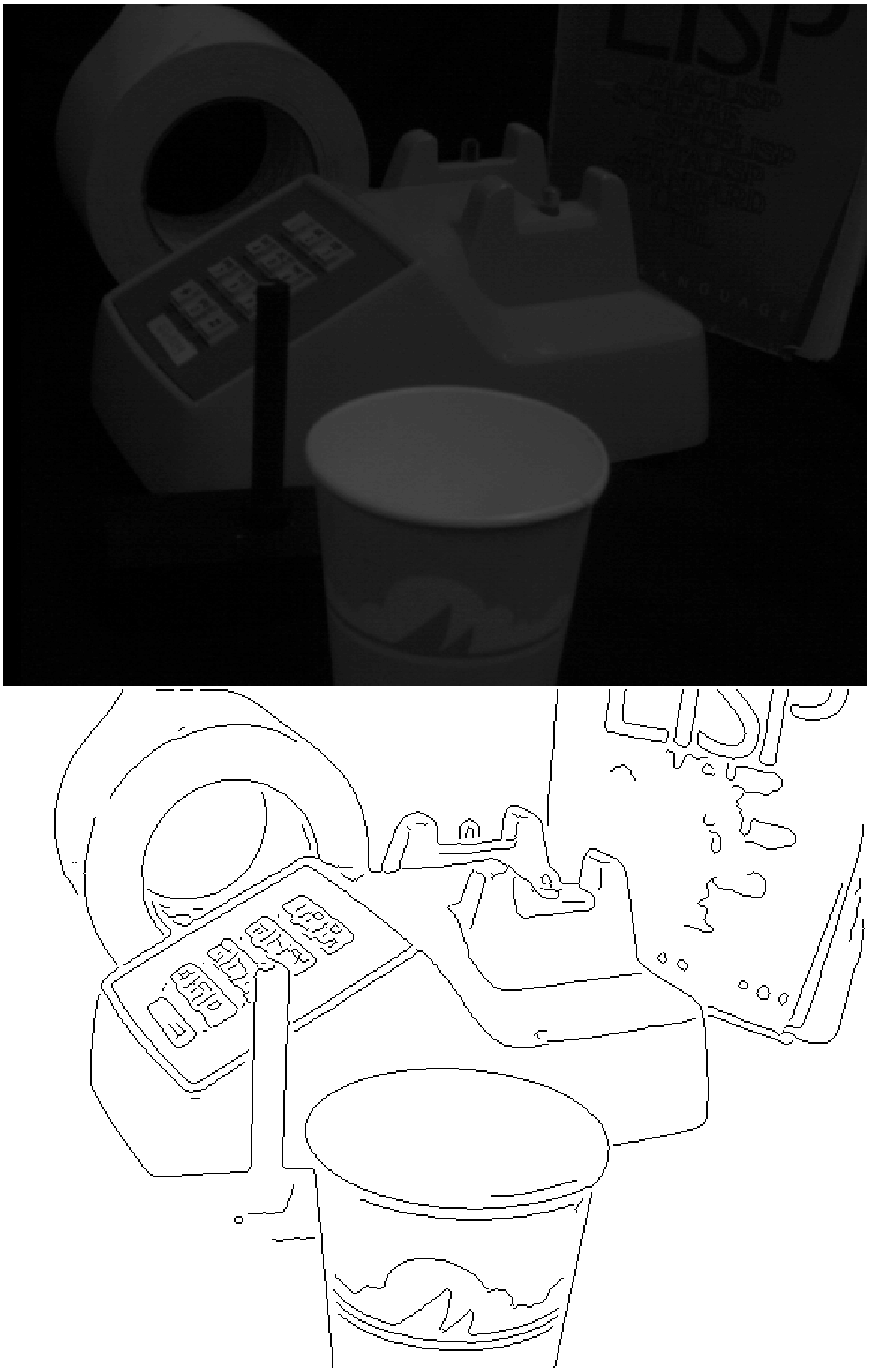


Figure 7.9: On top, a picture of the telephone with some additional occlusion, from the first series of recognition tests performed. Below, the edges found in the image.

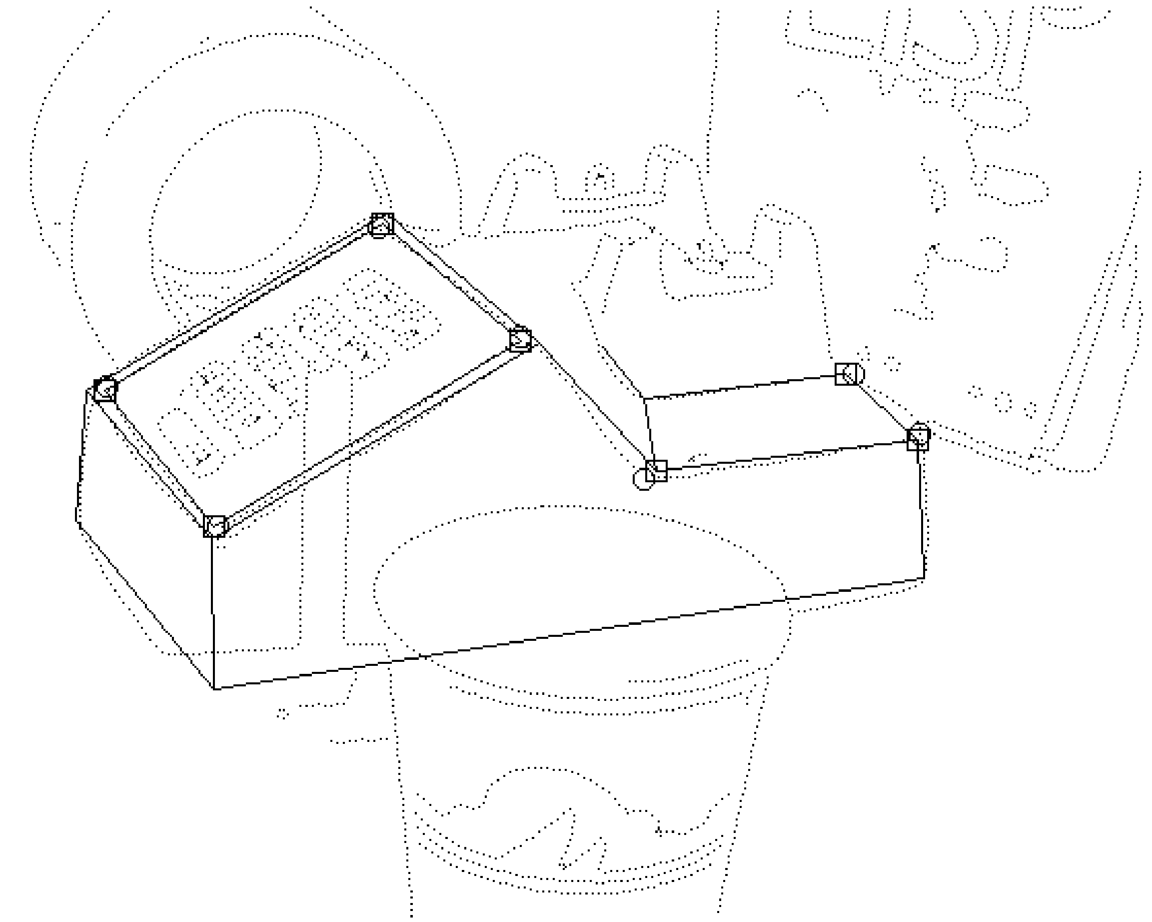


Figure 7.10: This shows the correct hypothesis, which the system found, for the image shown in the previous figure.

Figure 7.11: On top, a picture of the telephone in which some occlusion is added, from the first series of recognition tests performed. Below, the edges found in the image. The correct hypothesis was not found for this image.

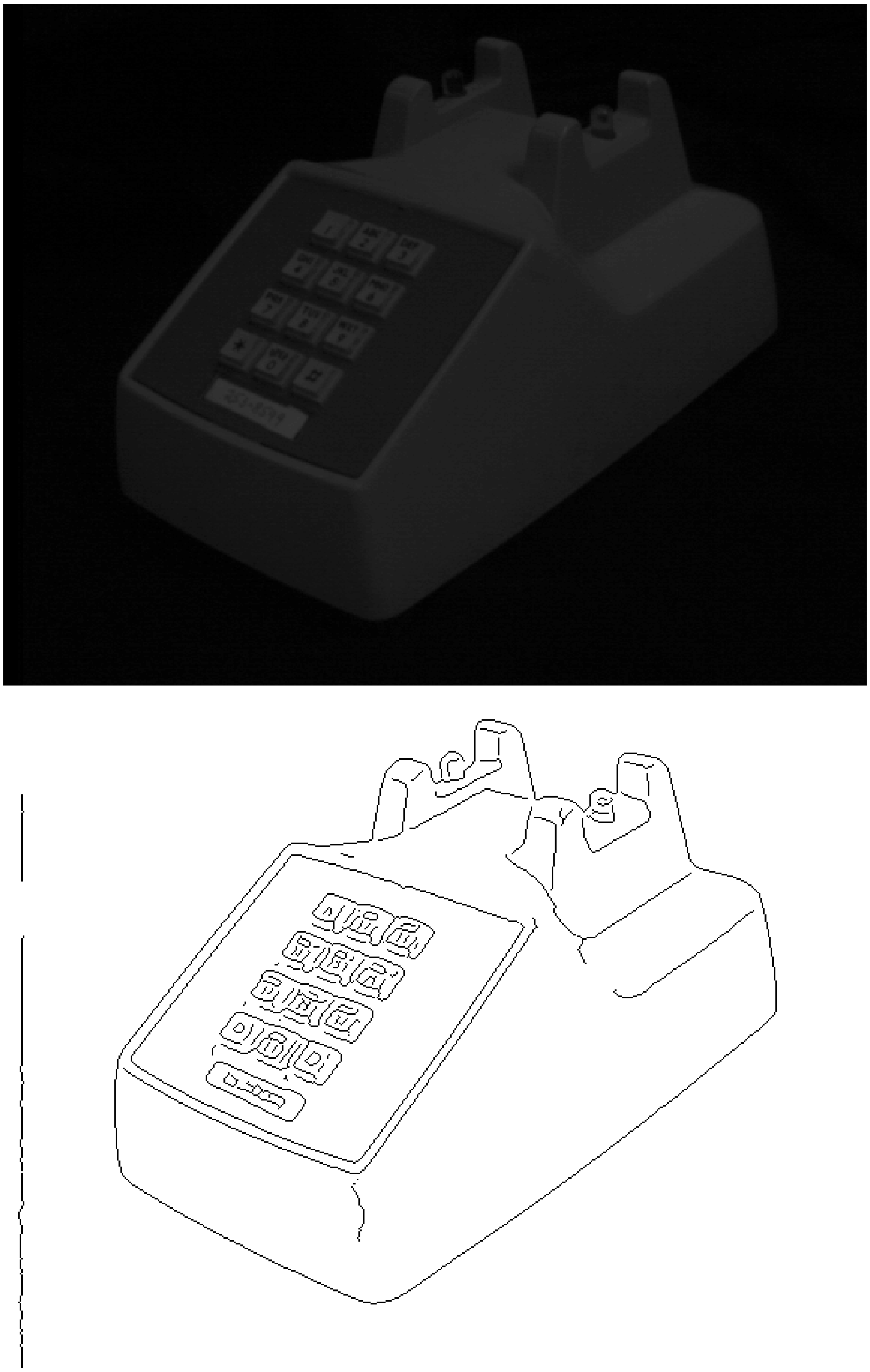


Figure 7.12: On top, an isolated picture of the telephone, from the second series of recognition tests performed. Below, the edges found in the image.

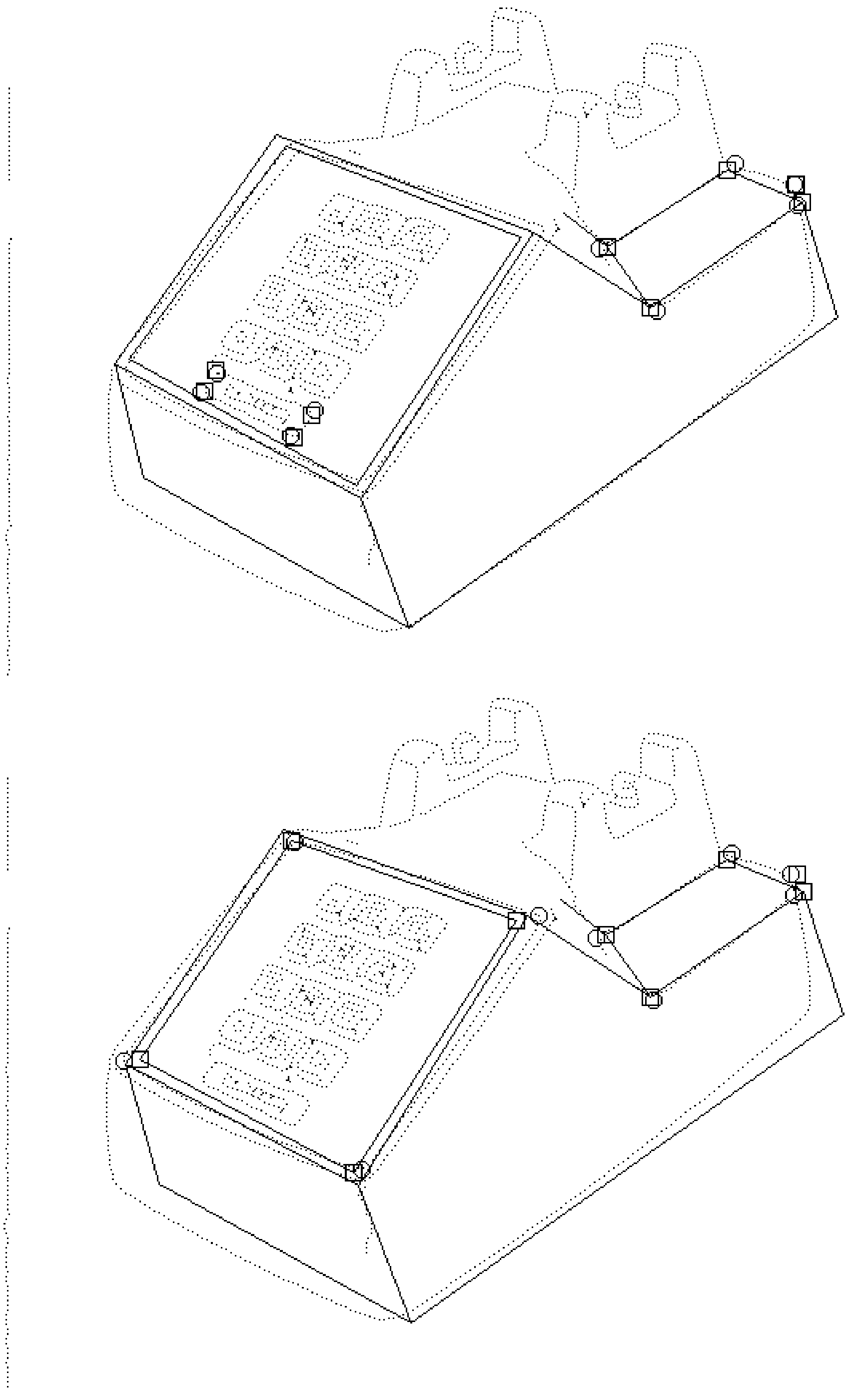


Figure 7.13: This shows two correct hypotheses, which the system found, for the image shown in the previous figure.

Figure 7.14: On top, a picture of the telephone with some objects in the background, and some occluding objects, from the second series of recognition tests performed. Below, the edges found in the image.

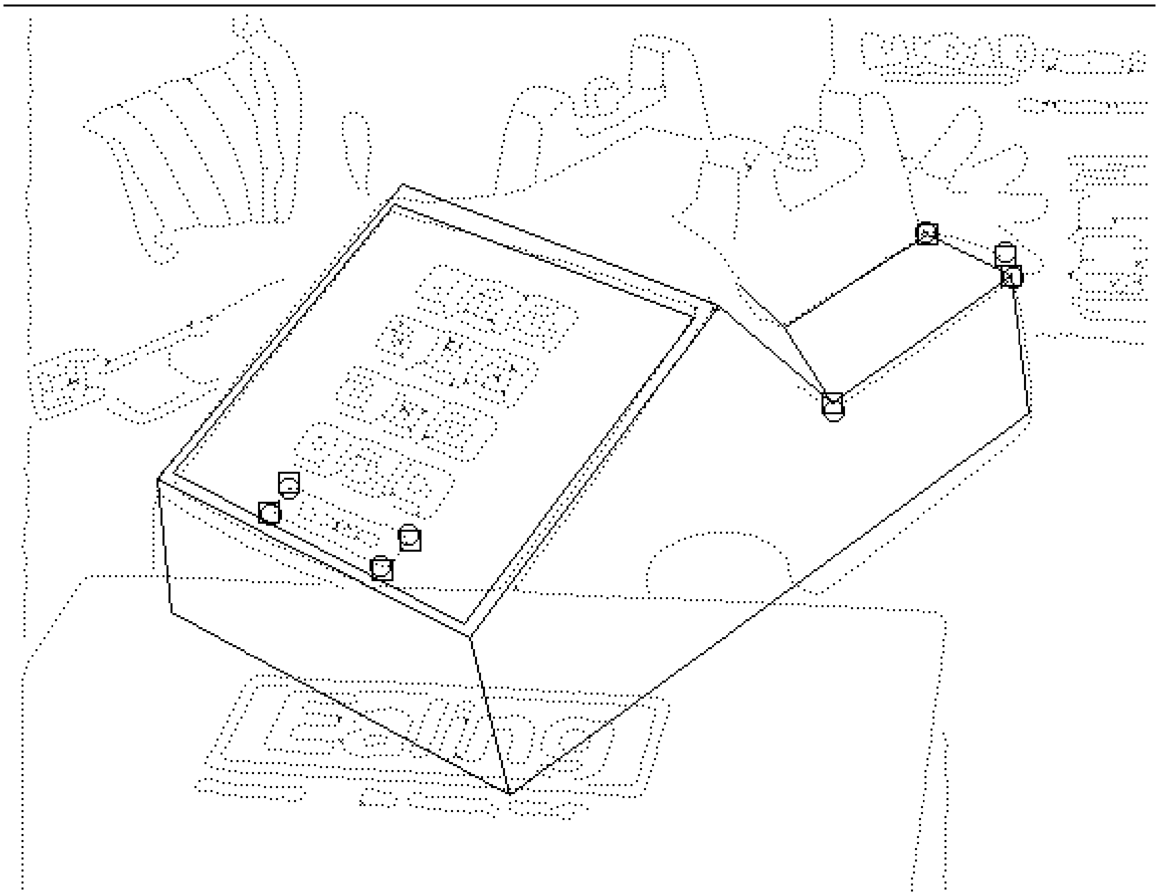


Figure 7.15: This shows the correct hypothesis, which the system found, for the image shown in the previous figure.

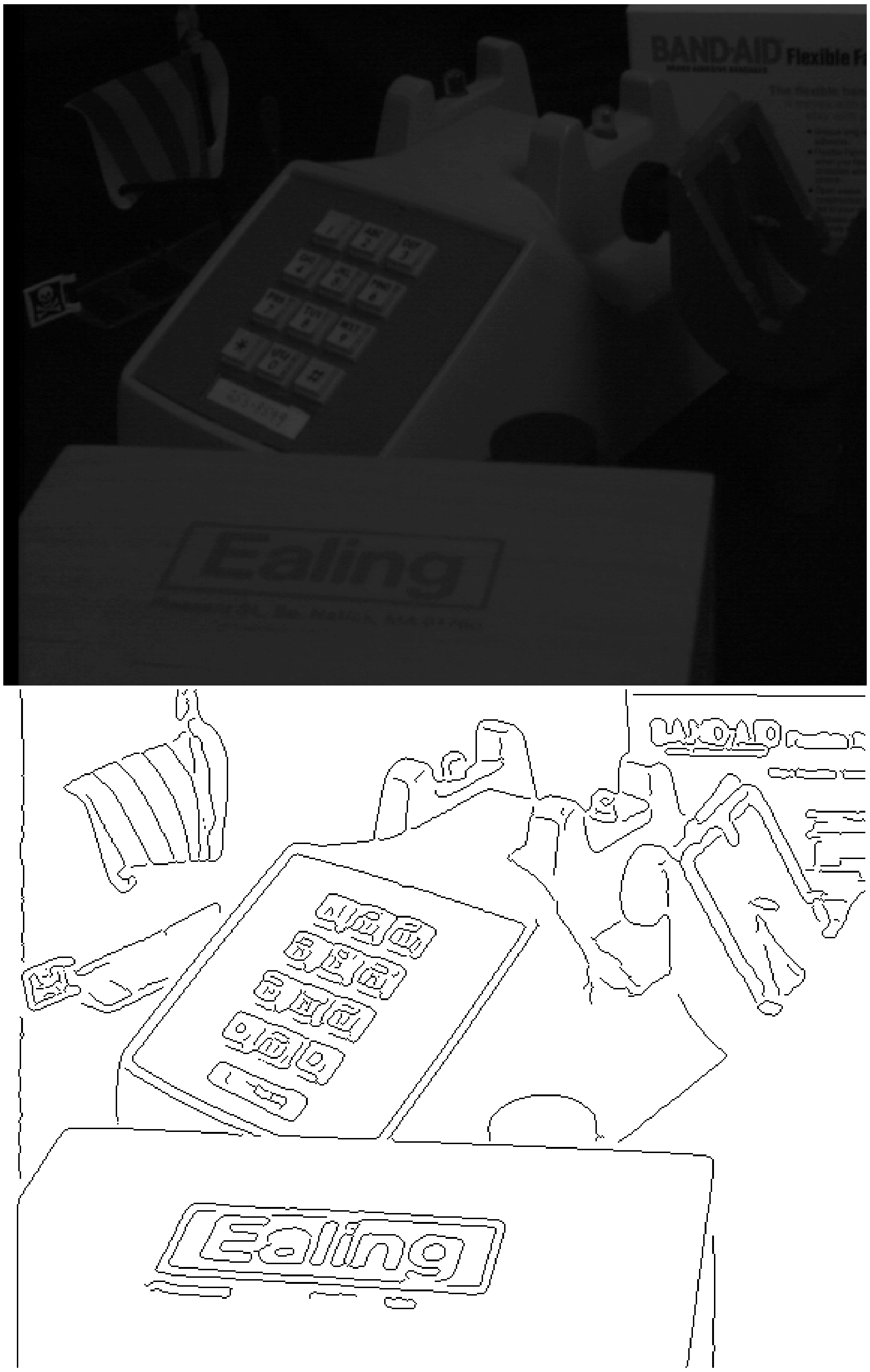


Figure 7.16: On top, a picture of the telephone in which some occlusion is added, from the second series of recognition tests performed. Below, the edges found in the image. The correct hypothesis was not found for this image.

Figure 7.17: On top, a picture of the telephone with some other objects in the background, from the third series of recognition tests performed. Below, the edges found in the image.

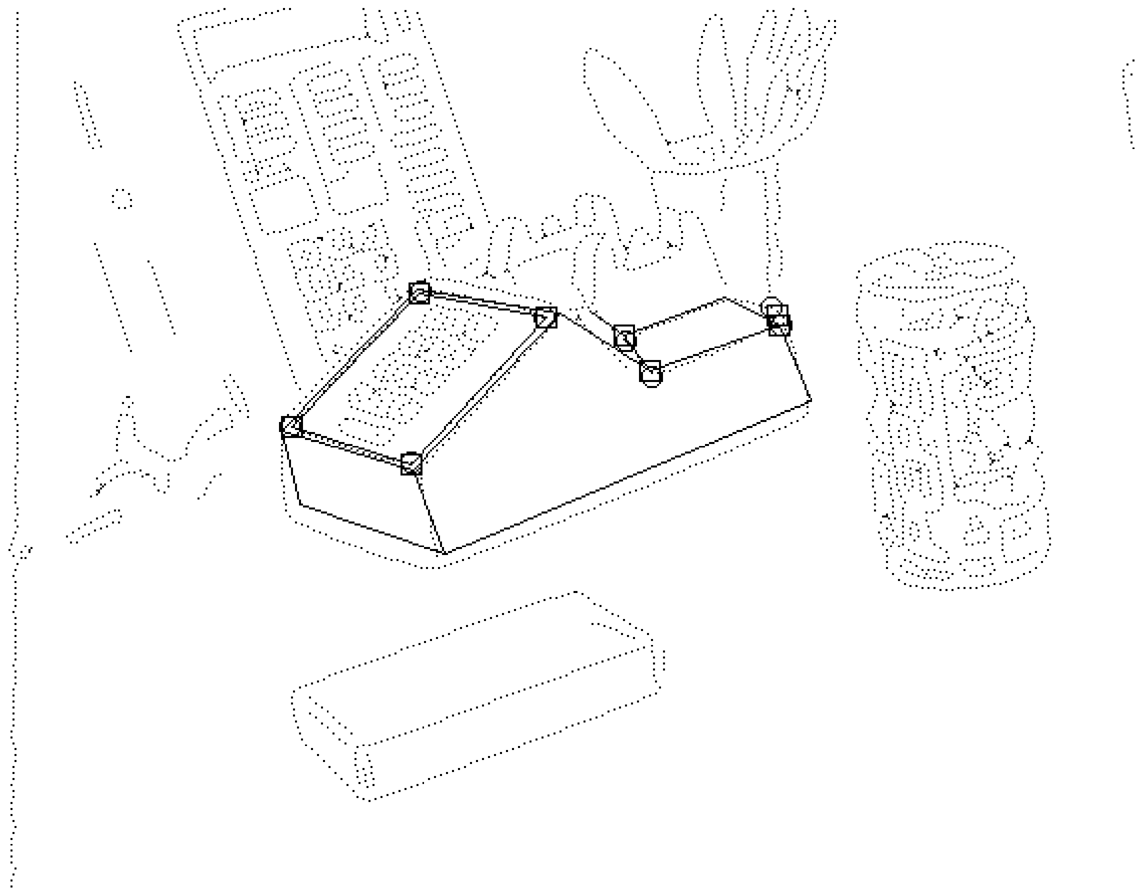


Figure 7.18: This shows the correct hypothesis, which the system found, for the image shown in the previous figure.

Figure 7.19: On top, a picture of the telephone in which some occlusion is added, from the third series of recognition tests performed. Below, the edges found in the image.

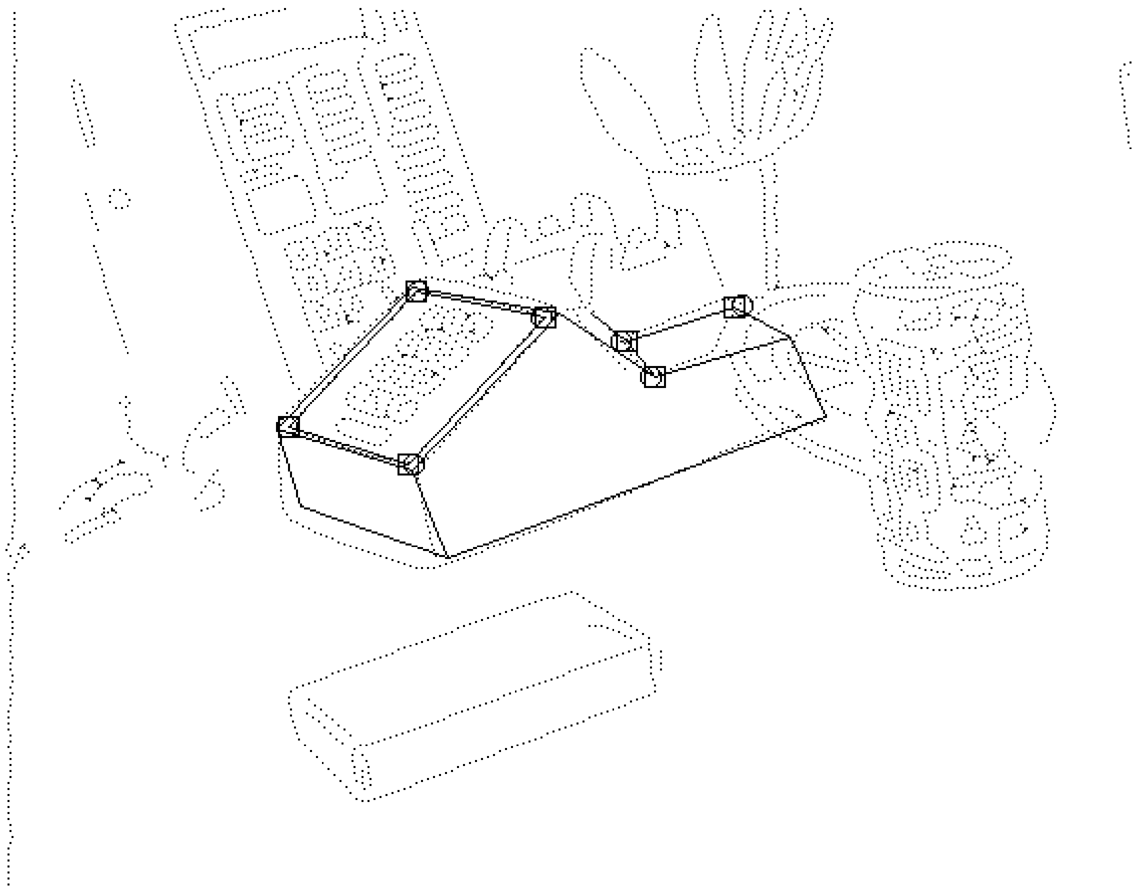


Figure 7.20: This shows the correct hypothesis, which the system found, for the image shown in the previous figure.

Figure 7.21: On top, a picture of the telephone in which some occlusion is added, from the third series of recognition tests performed. Below, the edges found in the image. The correct hypothesis was not found for this image.

Figure 7.22: On top, a picture of the telephone in which some occlusion is added, from the third series of recognition tests performed. Below, the edges found in the image. The correct hypothesis was not found for this image.

the performance of the indexing system now that it is coupled to a grouping system. Finally, we will show some relatively minor problems that occur due to our simplistic method of verification. These final problems should be easily resolved.

In examining the failures in our grouping system, we find a number of simple problems that could be easily fixed to improve the system's performance. We also find a few intriguing failures, that illustrate some difficult problems that remain.

Occlusion is one of the main reasons that we may fail to find a group in an image. Of course if a corner point is occluded, there is no way to find it. We partially compensate for this by including some subsets of the groups in our lookup table, but this is only effective if the occlusion is not too great. Our grouping system may also be effective if part of a group that does not contribute to a corner point is occluded, but again, if the occlusion is too great, the salience of the group may be significantly lowered. Figure 7.10 shows an example of a partly occluded group that is still found and used to recognize an object.

A pervasive problem in the examples that we have shown is that many potentially useful groups are found in the image which we have not represented in our lookup table. For example, each row of buttons on the front of the telephone tends to produce a salient parallelogram in the image. Also, we did not represent the front rectangle of the telephone, which includes points: (1 2 3 13) (figure 7.1 shows the locations of all numbered points), because the line from point 1 to point 13 usually did not appear when we were building our model. However, this group was found in the image shown in figure 7.21, for example, as is shown in figure 7.32, and the presence of that group in the model would have allowed the system to recognize the telephone in that image. A number of other salient groups contain corners that came entirely from the telephone, but were not represented in the lookup table. There is some danger in representing too many groups in our model. Since all pairs of groups must be entered in the lookup table, increasing the number of groups produces quadratic growth in the space requirements of the system, in the compile time requirements of the system, and presumably in the number of spurious matches produced by the system. However, it seems that significantly better performance could be achieved without too great a cost by perhaps doubling the number of groups used.

We should also mention that the failure of the system to find some convex groups appears to be due problems in the system that we have not diagnosed. Also, the system eliminates small groups from consideration, and this seems in practice to have caused it to bypass some useful groups, particularly the ones with points (31 32 33 34).

Overall, it seems that with some quite straightforward effort the system could succeed in recognizing the telephone in all the images shown in this chapter. We can also see, however, that the system fails to find some groups due to problems that would be more challenging to address. One such problem is that a salient, potentially useful group can be overshadowed by a more salient, but spurious group that uses some of the same lines. For example, in figures 7.26 and 7.27 we can see that the group of lines forming the inner square of the keypad, which produces points (14 15 16 18), appears among the set of the second most salient groups in the image, because an occlusion produces a more salient group that contains some of the same lines. The general problem of determining which groups are most meaningful and should be used to attempt recognition is quite a challenging one. Our salience fraction provides only a rough and simple solution.

Also, sometimes an extraneous or occluding line may be included in a group, contributing to an extra point feature. For example, if one closely examines the group in figure 7.30 that appears to produce points (10 17 11), one sees that point 11 is not found, but that an occlusion produces a new point near this location. This group, although slightly incorrect, does contribute to the successful recognition of the object. To handle these sorts of problems, one would need to reason about which points in a group reflect some essential structure, and which points come from occluding or spurious lines. This problem seems quite difficult.

We can also see examples in which the instability of point features can cause difficulties. There are several examples of groups in which one or more point features do not appear due to slight changes in the underlying edges. For example, if one closely compares the correct hypotheses shown in figures 7.6 and 7.8, one sees that point 22 disappears in the second image, even though the underlying scenes and edges appear almost identical. In this case, the problem is handled because we represent that group in the model both with and without this point. As another example, in figure 7.30, we can see that in the group containing points (1 2 3 13) there are two nearby corner points where we would expect point 1 alone to be found. In both these cases, slight variations in the edges can lead to changes in the resulting line segments that either produce or eliminate a corner. While we partially handle this problem, our solution is still not complete.

Overall, we can see that our convex grouping system is quite successful at finding salient convex collections of lines that can be used to recognize objects. More work could be done, however, to determine the best ways of making use of these groups. This includes the problems of determining which point features are due to some stable underlying structure, the problem of determining which groups are most salient and most likely to be useful, and the problem of determining which groups should be paired together.

Our experiments also demonstrate the effectiveness of our indexing system. We found no examples in which indexing failed to match a group-pair of image features to the appropriate model features. And by indexing with many image group-pairs and beginning our search with the ones that matched the fewest model group-pairs we also produced short searches.

We can also see some potential for interference, though, between the speedups provided by grouping and the speedups provided by indexing. Indexing using groups provided by our grouping system produced lower speedups than indexing using random point features did in our earlier tests. It is not hard to see why. Our grouping system will produce collections of model points and collections of image points that will cluster in certain portions of our lookup tables. For example, the first four points in a group-pair often come from a single convex group. If these four points are mutually convex, this restricts the set of possible affine coordinates that can describe them. If four points come from a single convex group, than the fourth point cannot have affine coordinates that are both negative, or that are both positive and sum to less than one, for example. Also, our grouping system frequently produces pairs of groups in which the points in each group are nearby, and the points in different groups are widely separated. This again causes both models and images to cluster in certain parts of the lookup table, reducing the potential speedups of indexing. In effect, grouping is doing some of the same work as indexing. Since we only consider matching image points collected together by our grouping system to model points collected together by grouping, grouping is causing us to only consider matches that are more likely to be geometrically consistent than are random image and model groups. This means that when indexing precisely enforces geometric consistency, some of the constraint in this consistency has been already more roughly used by the grouping system.

Finally, we mention that our recognition system produces some incorrect hypotheses that nevertheless pass the thresholds used by our verification system. One reason for this is the simple nature of our verification module. Since we use a few line segments to model the telephone, and since we do not perform hidden line elimination, there are some quite incorrect poses that match a significant number of image lines. An example of this is shown in figure 7.36. This could be handled by a more careful verification system. A second problem can occur if we generate a hypothesis that is almost, but not quite correct, as shown in figure 7.37. In this case, the inner square of the keypad in the image is matched to the outer square of the keypad in the model. In order to handle this problem, we would need some method of improving our estimate of pose by changing it slightly. We have not attempted to address problems of verification in this work, however.

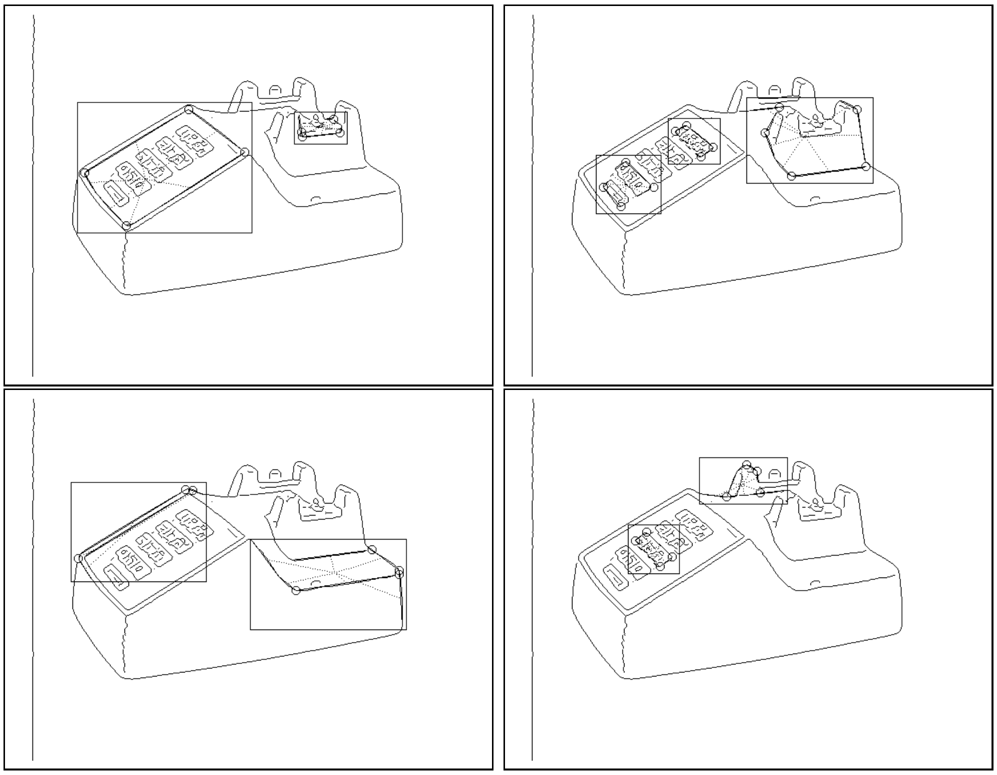


Figure 7.23: Some of the most salient groups found in the image shown in figure 7.3. These are the groups with the highest salience fraction, given that no line segment is allowed to appear in more than one group with the same orientation. These groups are continued in the next figure. It is this set of groups that is used in our recognition tests.

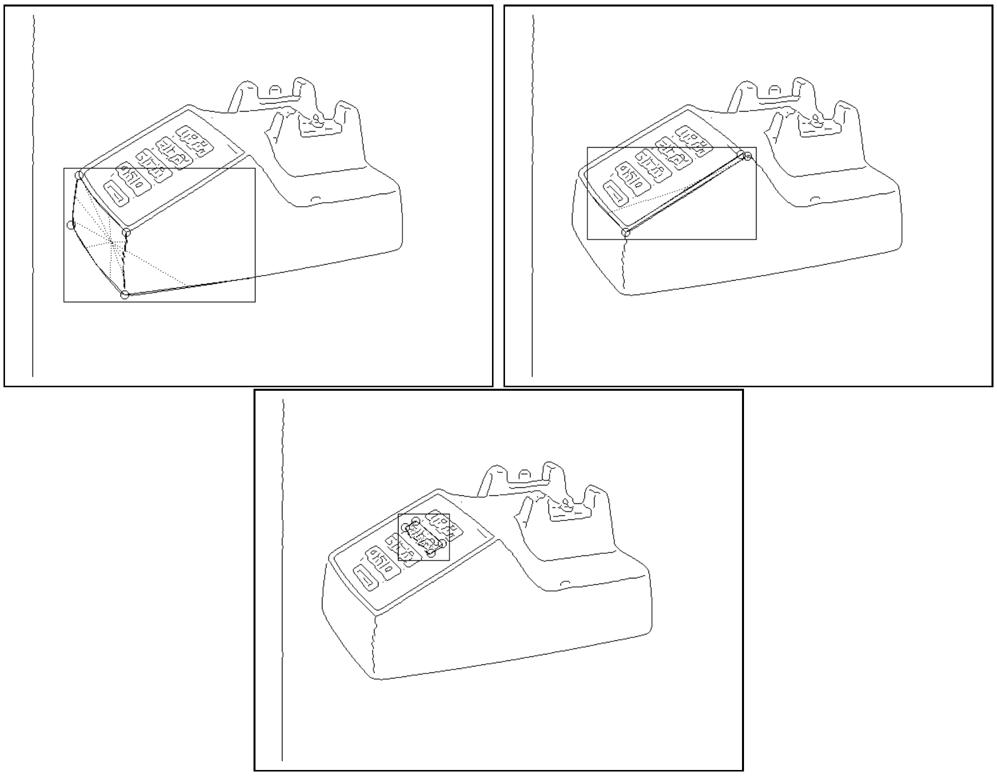


Figure 7.24: The rest of the most salient groups found in the image shown in figure 7.3.

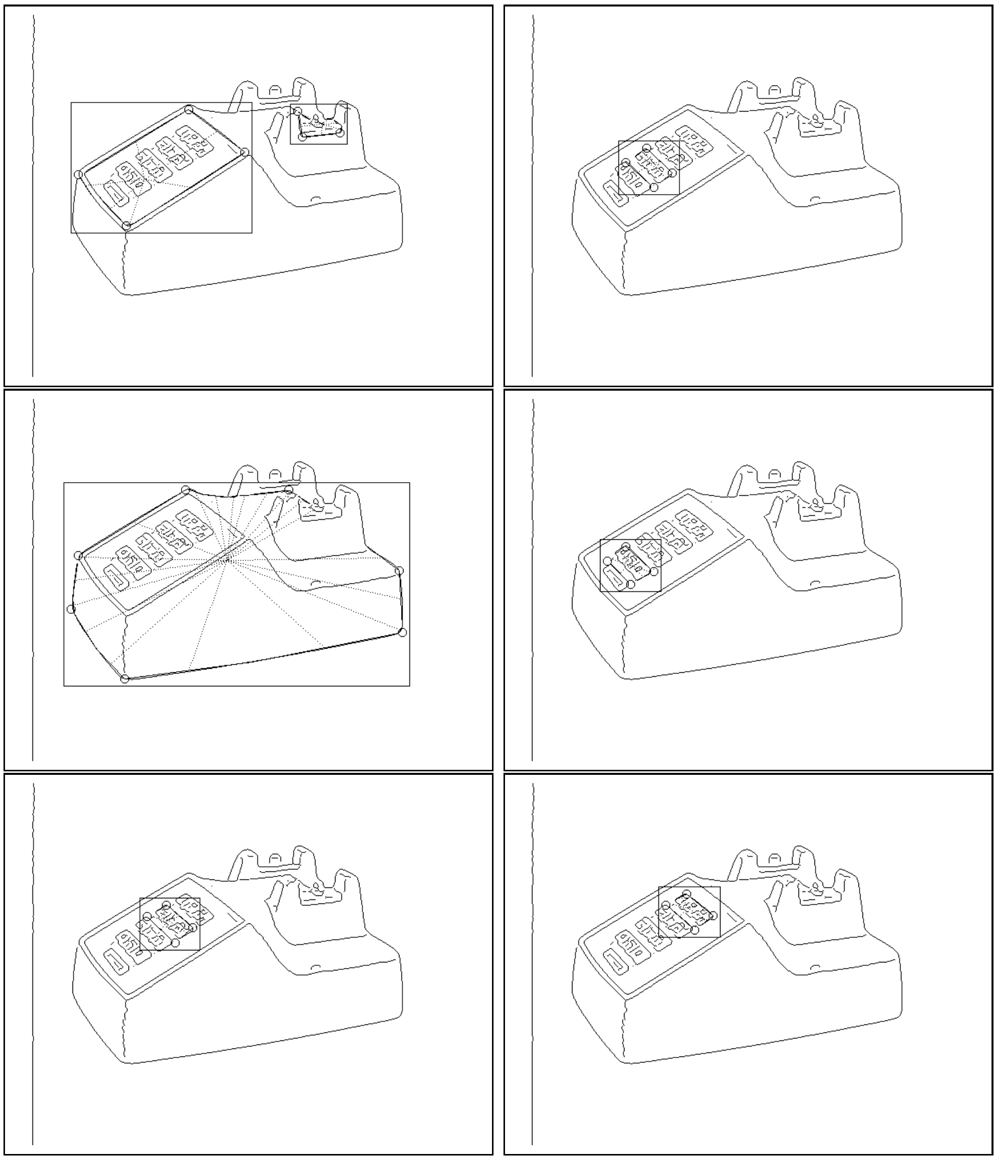


Figure 7.25: A set of the next most salient groups found in the image in figure 7.3. These groups contain lines that may have appeared in a previously chosen salient group. These groups were not used in our recognition tests.

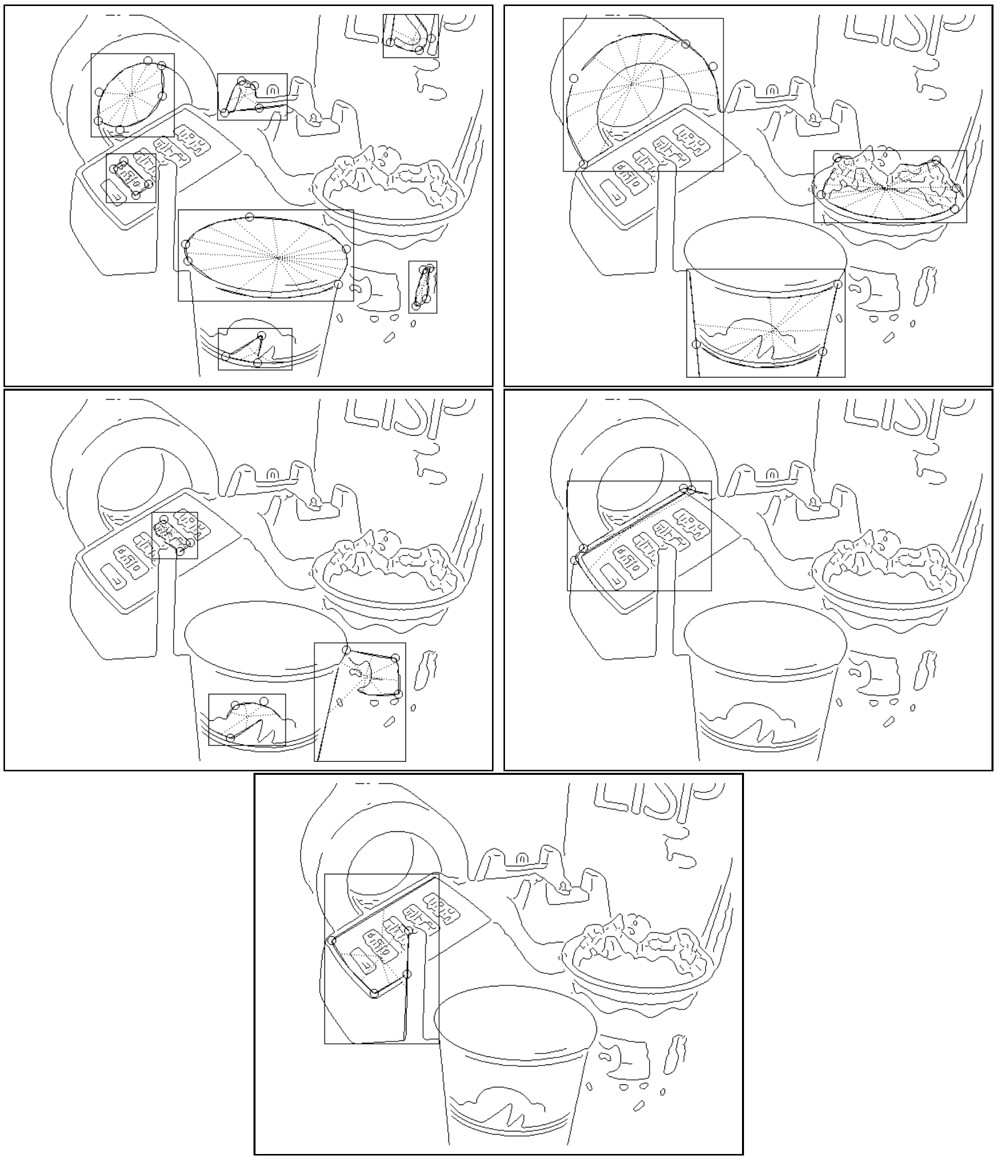


Figure 7.26: This shows the most salient groups found in the image shown in figure 7.11.

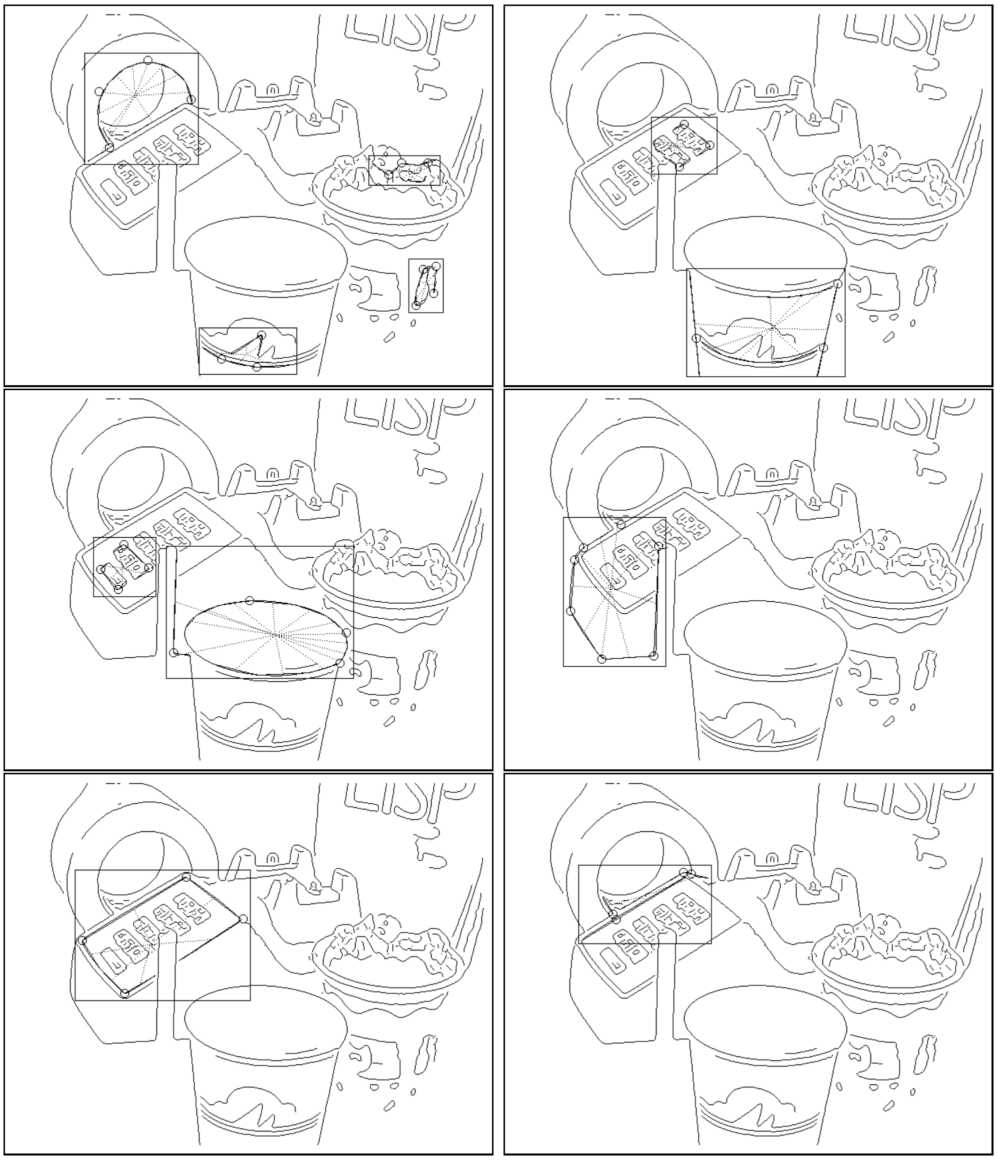


Figure 7.27: This shows the second most salient groups found in the image shown in figure 7.11.

Figure 7.28: The most salient groups found in the image shown in figure 7.17. These are the groups with the highest salience fraction, given that no line segment is allowed to appear in more than one group with the same orientation. It is this set of groups that is used in our recognition tests.

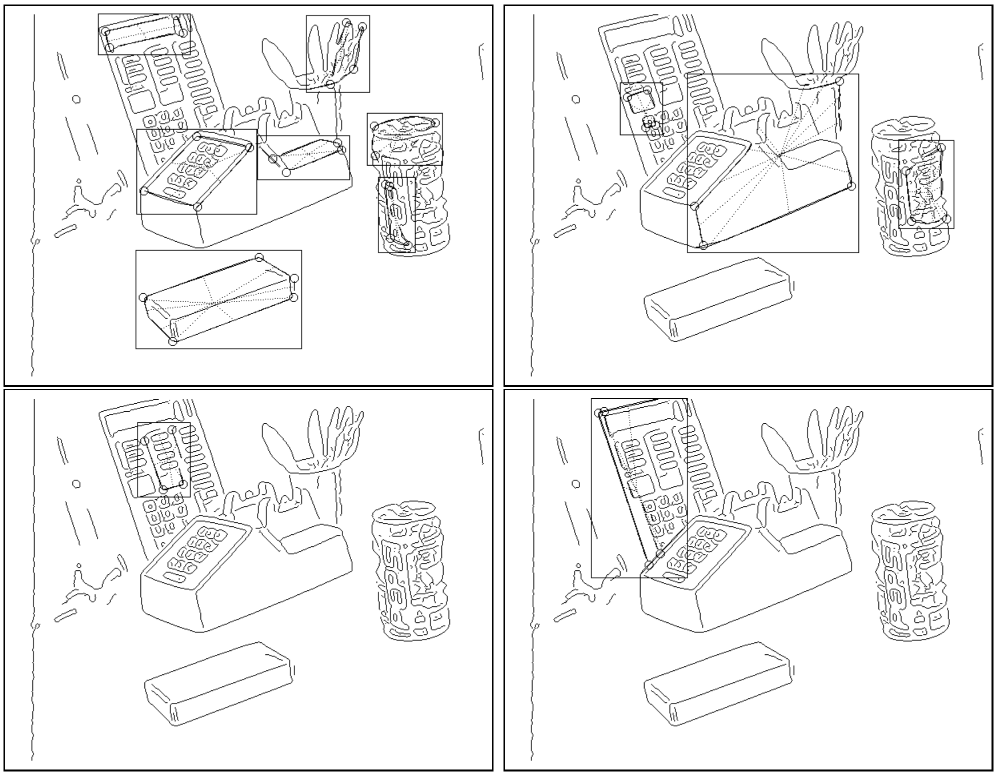


Figure 7.29: The set of the second most salient groups found in the image shown in figure 7.17. These groups were not used in the recognition tests.

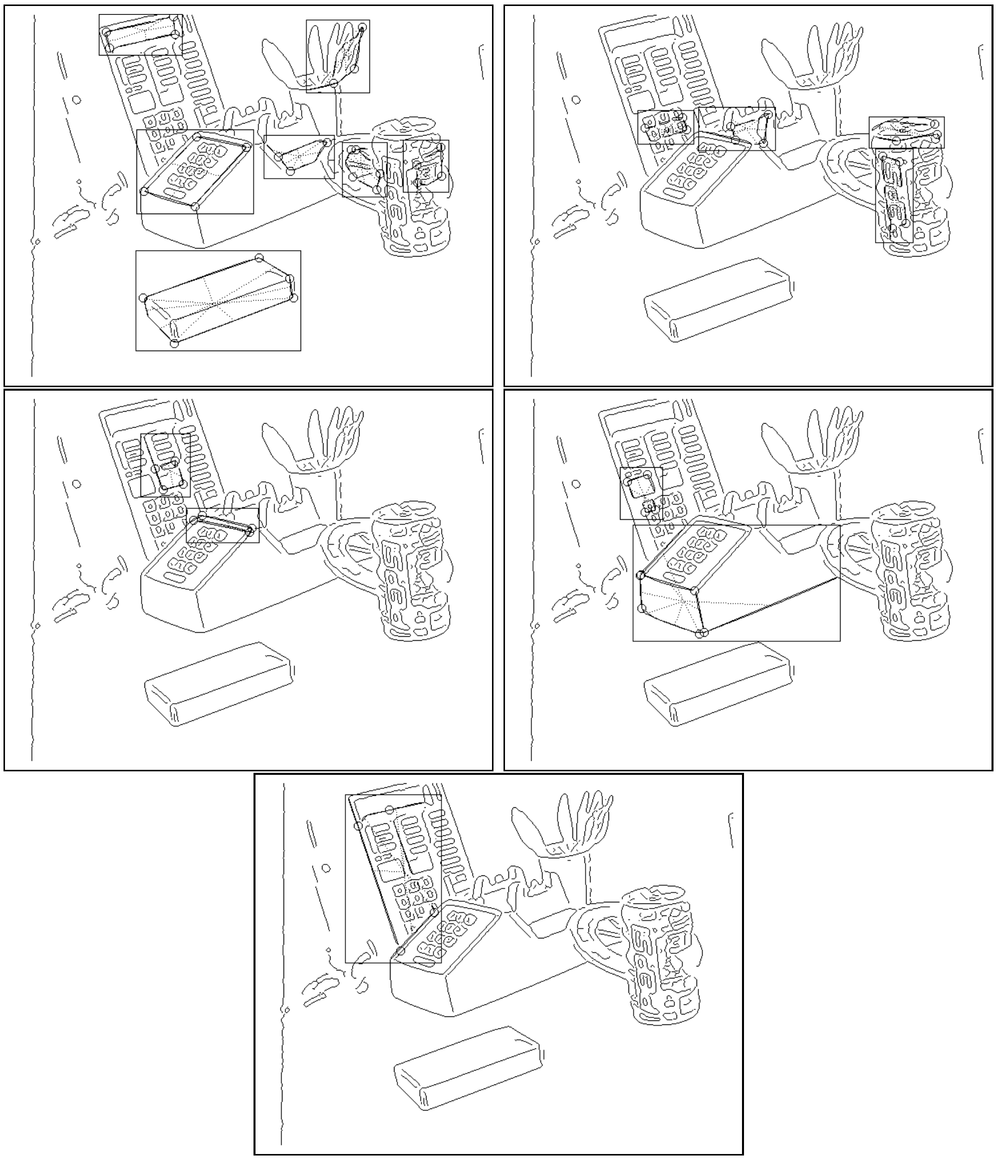


Figure 7.30: This shows the most salient groups found in the image shown in figure 7.19.

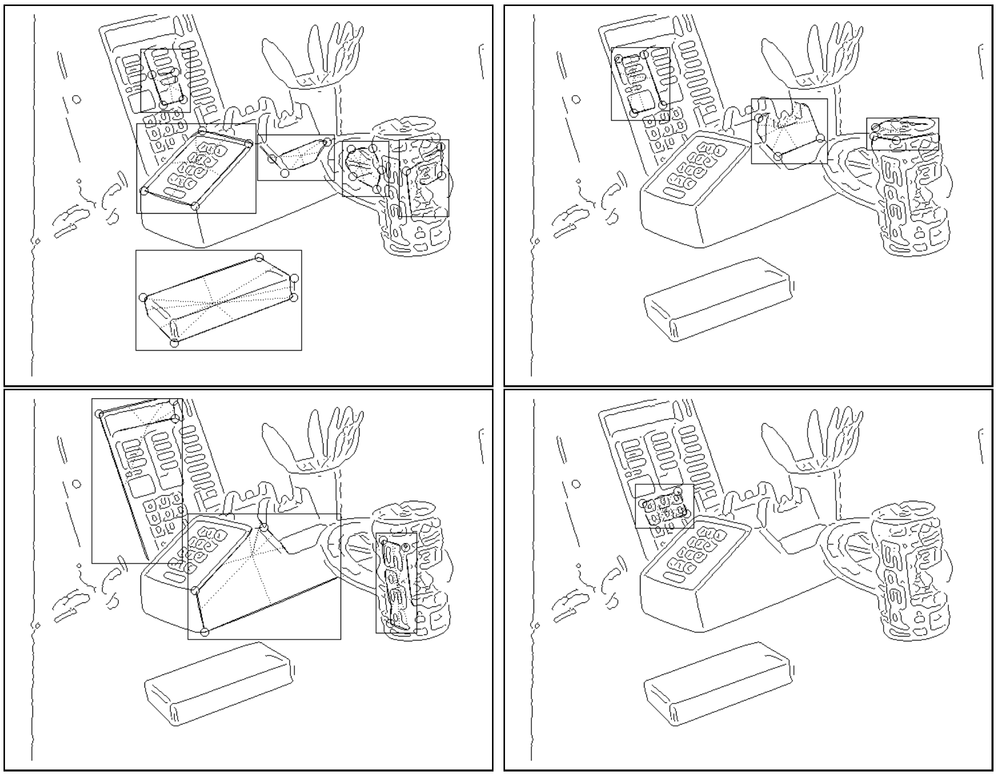


Figure 7.31: This shows the second most salient groups found in the image shown in figure 7.19.

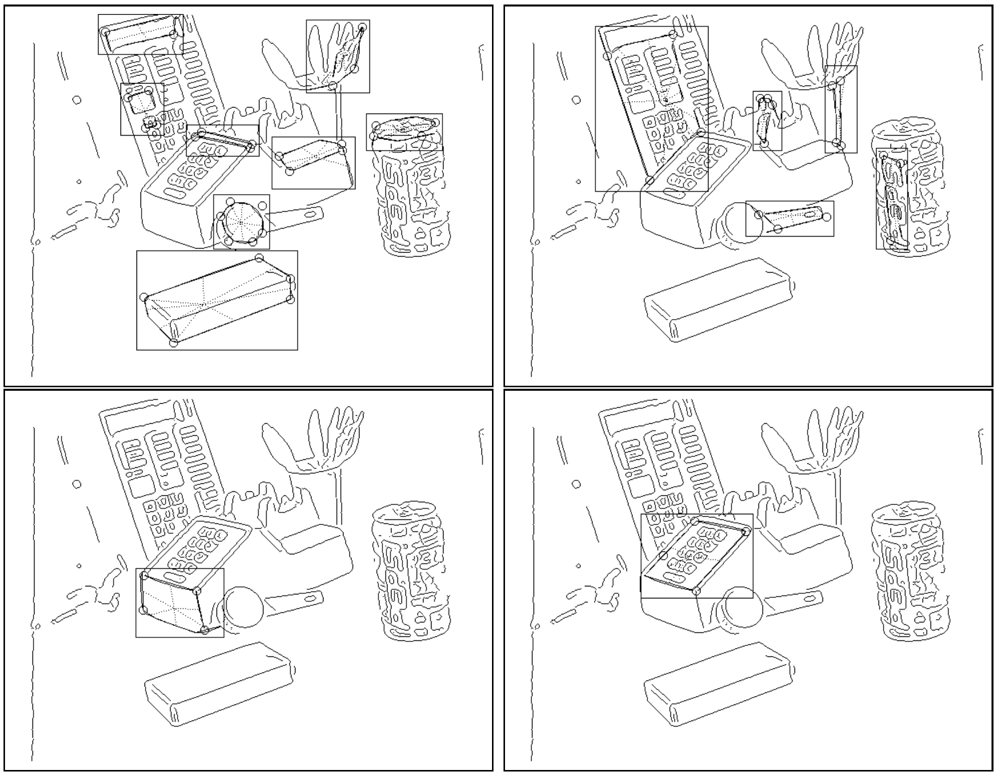


Figure 7.32: This shows the most salient groups found in the image shown in figure 7.21.

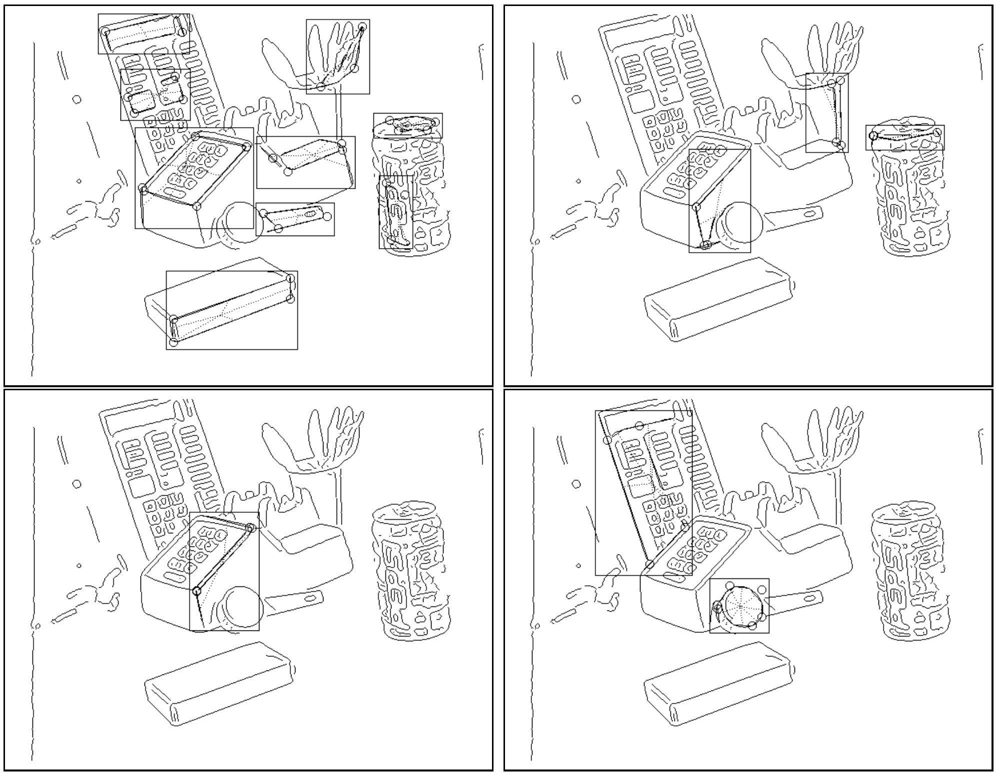


Figure 7.33: This shows the second most salient groups found in the image shown in figure 7.21.

Figure 7.34: This shows the most salient groups found in the image shown in figure 7.22.

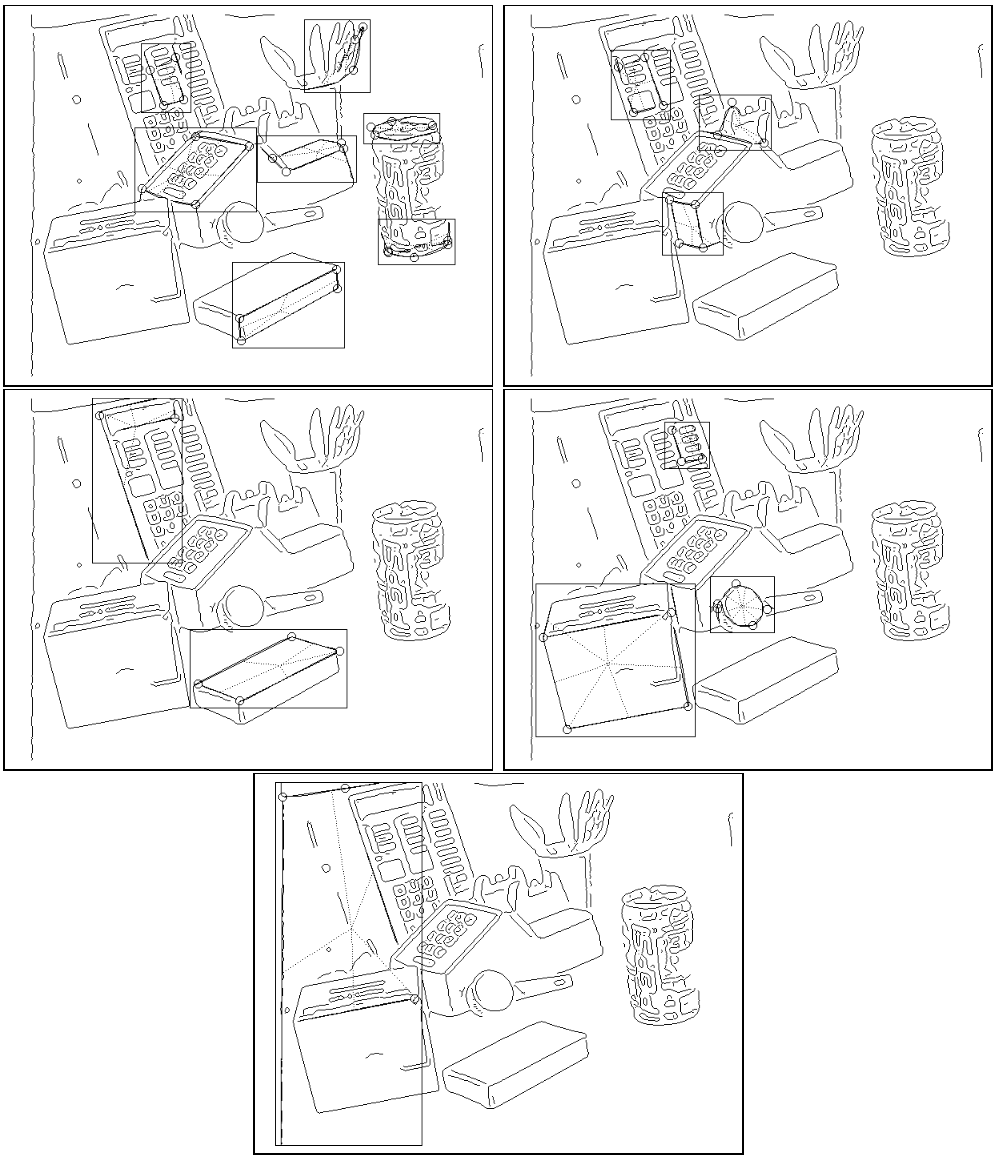


Figure 7.35: This shows the second most salient groups found in the image shown in figure 7.22.

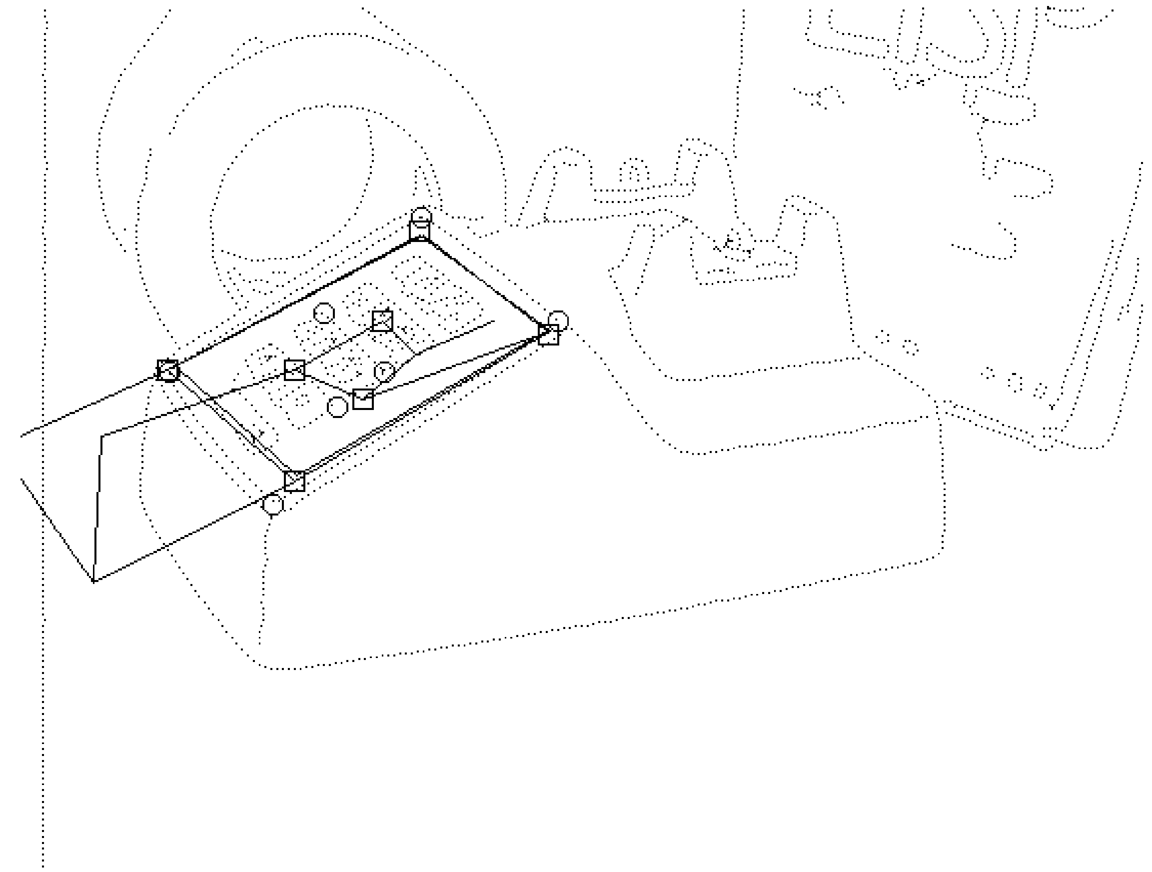


Figure 7.36: More than half of this incorrect hypothesis still matches image line segments, due to the simplicity of the model that we use for verification. As a result, this hypothesis passes our verification threshold.

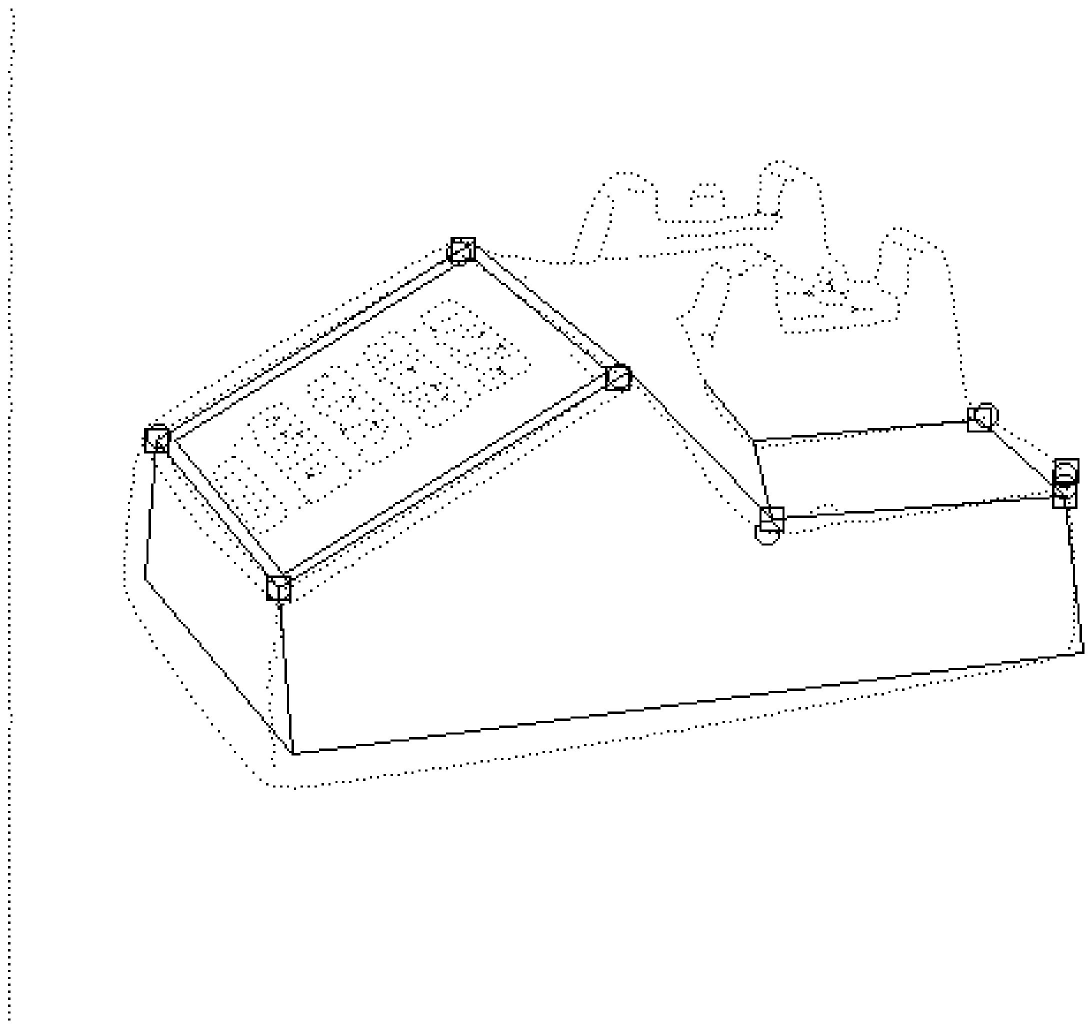


Figure 7.37: In this hypothesis, the inner square on the keypad of the phone in the image is matched to the outer square in the model. This results in a hypothesis that is just a little wrong.

7.3 Conclusions

This chapter has shown that the grouping and indexing systems that we have built can form useful components of a complete recognition system. Our indexing system provides correct matches while using point features located in noisy images by a real feature detector. Our grouping system finds many salient groups that are useful for recognition. Together, we have shown that these two subsystems have the potential to dramatically reduce the amount of search required to locate objects.

One implication of this is that the systems we have built can serve as major components for a practical recognition system in certain domains. Our grouping system will support recognition of objects that have a number of convex parts, at least some of which appear unoccluded in images. And in some domains simple methods of pairing convex groups, using connectivity or proximity for example, may quickly provide pairs of groups that come from a single object. The recognition of buildings using aerial imagery, or the recognition of many manufactured parts in factory environments are two examples of domains in which our grouping system may be adequate as it is, or with simple modifications. And we have demonstrated that our indexing system is sufficiently robust to be useful whenever grouping can provide us with groups of image points that come from the points of a precompiled model group.

Our system also demonstrates the potential of a recognition strategy based on grouping and indexing. The effectiveness of this strategy is limited by our grouping system's ability to produce groups of many image point features that all come from a single object. There is much work to be done before such grouping can be performed reliably in many realistic domains. We have pointed out some of these challenges, already. We must integrate many different grouping clues to achieve robustness when salient convexity alone is insufficient. We must learn to combine small groups into larger groups effectively. And we must robustly derive point features from groups of edges that are often curved or noisy. The greatest potential of indexing will be realized only as we learn to produce larger groups of point features that more reliably match our models. But we have shown that our present grouping system is already sufficient to produce significant speedups in a variety of real situations. As methods of grouping improve, the effectiveness of our strategy will be increased further.

Chapter 8

Conclusions

This thesis considers both difficult long-term questions of visual object recognition and more practical short-term questions involved in building useful applications. To do this we have developed an understanding of recognition in a domain of simple features. In this domain, we have shown that achieving human performance requires a strategy that can control the complexity of the problem, and that grouping and indexing together have the potential to do this. We have also developed tools that allow us to come to grips with some fundamental questions of recognition, such as: "How should we describe a 2-D image so that we can use this description to access our memory of 3-D objects?", and we have provided an example of how the difficult problem of grouping can be approached. At the same time, we have produced some tools that can be of practical value in building recognition systems of more limited scope. We have developed a useful grouping system and an efficient indexing system for point features, and we have broadened our understanding of the effects of error on recognition systems. Our goals in this chapter are to describe the connection between our analysis of a simple domain and the larger problem of general object recognition, and to describe the strengths and limitations of our practical tools, making it clear where more work is needed.

8.1 General Object Recognition

In the introduction, we sketched a view of general object recognition that involves isolating interesting chunks of the image and describing them in a way that can trigger our memories. This approach to recognition gives rise to difficult questions. How do we divide the image into usable pieces? How do we describe these pieces of the image? Is the description in 2-D, 3-D or some mixture of the two? Why would we want to capture some properties of an image in our description, while ignoring others? What is the role of context, prior knowledge, and the needs of the particular situation in determining what is a good description? And perhaps most importantly, why might this overall strategy provide a good way of recognizing objects? In this thesis we have considered a simplified domain where it is easier to thoroughly understand some of these questions.

We can see in our domain that computational complexity presents a tremendous challenge to object recognition systems. Moreover, this does not seem to be an artificial problem produced by the simplicity of the domain, for it appears that as we make our domain more realistic, problems of complexity will only grow worse. We have also shown the relationship between grouping and indexing. Each of these tools has only limited potential by itself to control complexity. Grouping limits our search within the image, but does not reduce the number of models that we must compare to an image group. Indexing can only significantly reduce our search if we can perform indexing with large groups of image features. Without grouping, we have no efficient way of finding a reasonable number of groups with which to perform indexing. Although a strategy of using grouping and indexing for recognition may seem obvious, it is useful to see the necessity of this strategy in a concrete domain. Also, there has been much work on indexing using invariants in recent years. and relatively little work on grouping. It is important to stress that this work on indexing, while valuable, is only half the picture. Without more effective grouping techniques, indexing may be applied to only very simple kinds of images, in which very simple grouping methods will work.

The center of this thesis has been devoted to characterizing the images that a model can produce, and to showing the value of simple solutions to this problem. We have shown that indexing models that consist of point features is equivalent to matching a pair of points, in two image spaces, to pairs of lines that represent possible groups of model points. The spaces are simple Euclidean ones, and any point can correspond to an image, while any line can correspond to a model. By reducing indexing to a simple, symmetric form, we have produced a powerful tool for analyzing various approaches to recognition.

This work allows us to see the limitations in our domain of attempts to infer 3-D structure from a single 2-D image. We see that there are no invariant functions for general 3-D models, that there are no sure inferences of 3-D structure, and that the arguments sometimes put forward to explain the value of perceptually salient structures such as parallelism and symmetry have significant limitations. These results have clear implications for recognition within our domain. They do not settle the issue for more complicated domains. We cannot infer 3-D structure in a world of point features in which we make no assumptions about the a priori distribution of objects. This does not mean that this structure cannot be inferred in more realistic domains. But we have shown that explanations of 3-D inference, or of the special role of perceptually salient structures must lie outside the domain that we have studied. Many past approaches to understanding these problems have been so general as to apply to any domain, and we can see that these explanations must fail.

We have not directly addressed the question of why one description of an image should be used to access memory instead of another. But our conception of visual memory as a problem of geometric matching in some image space provides a framework for addressing this question. A vocabulary for describing images can be a way of creating an image space and decomposing it into equivalence classes that mediate matching. When we describe an image with a set of quantitative values, we are defining an image space. When we describe an image qualitatively, we are making a commitment to treating that image as the same as other images that have the same description. That is, we are dividing image space up into chunks, and treating images that map to the same chunk of image space in the same way. A simple, analytic description of the images that a model can produce in an image space provides us with a tool for understanding the value of any particular choice of image space, or any method of decomposing that space qualitatively.

At the same time, the symmetry of the geometric problem that underlies recognition seems to preclude an answer to this problem in our domain. For example, it is this symmetry that undermines attempts to explain the value of descriptions based on non-accidental properties such as collinearity; it turns out that collinearity is no different from an infinite number of other features. In a similar way, any attempt to prefer one way of describing an image over another seems to be vulnerable to the symmetry of the simple geometric problem that is equivalent to visual memory.

This means that to provide an answer to some of the questions that we have raised, we must push forward into more complex domains. There seem to be two particularly important ways in which our current domain is too simple. First, we assume that models consist of collections of local features instead of surfaces. By expanding our work to surfaces we would be able to describe a world consisting of arbitrary polyhedral objects, a domain of considerable complexity. It is of great interest whether the 3-D structure of a scene may be inferred from a single image of polyhedral objects. As we have pointed out, there has been much work on this problem, but it remains challenging. It is particularly difficult and important to incorporate a notion of error and feature detection failures into such a world. Second, we have implicitly assumed that there is no structure to the kinds of objects that our world contains. We assume that all collections of point features are possible objects, and make no attempt to make use of hypotheses about the likelihood of different possible objects actually occurring, that is, we make no assumptions about prior distributions of objects in the world. In the real world objects are solid and self-supporting, they grow or evolve or are constructed to function in a world that has many physical constraints. Categories of objects exist naturally; for example there are inherent ways in which all camels are similar, and only certain ways in which they may differ. All these effects cause significant patterns in the kind of objects that actually exist. It may be that these patterns account for the kinds of representations that people use in recognizing objects. There are many possible sources of constraint that could contribute to the superiority of some methods of describing images for recognition. Do these constraints lie only in the imaging process? do they lie in the requirement that objects be solid and connected? do they lie in the nature of our physical world? or do they lie in the history of evolution, in the particular set of objects that nature has placed in our world, and in the particular way that categories of these objects may vary? We have looked at only the simplest source of possible constraints, and found it inadequate. We have ignored other elements of the real world of considerable importance. In particular, we mention that our work makes no attempt to explain how we recognize new instances of a category of object with which we are familiar, and that we have considered only the simplest instances of non-rigid objects. But gaining a firmer understanding of a simple domain should provide a useful step in understanding these more complex ones.

Grouping is also a difficult problem. We simplify it significantly by focusing on only a single grouping clue, salient convexity. It seems clear that ultimately we should combine many clues into a grouping system. By choosing one clue we bypass the problem of understanding others, and the particularly difficult problem of integrating multiple clues. We have used a probabilistic analysis to show under what circumstances convex groups may be salient and worth using in recognition. This analysis and our experiments also show that these groups may be located efficiently. By thoroughly understanding some individual grouping clues, we can contribute to a more complete approach that integrates these clues.

Finding convex groups may be useful also because they provide us with regions of the image that might be used to focus an analysis of color, texture or other region-based descriptions. For example, it has proven difficult to segment an image solely using texture, but if convex regions are found using our methods first, it may be easier to use cues such as texture to decide which groups are useful, and to decide which ones should be paired together. Also, if we intend to integrate many clues, it is especially important to be able to characterize the performance of each module that makes use of an individual cue. So in thoroughly exploring one grouping clue, we have attempted to produce work that will be useful to a more ambitious effort.

We have also used this grouping system to help us explore the interaction between grouping and indexing. We find that even an imperfect grouping system may be of value to a recognition system. We also show how a grouping system can simplify the problem of finding a correspondence between an image and a model group by providing additional information about the groups. In our case, convex groups provide information about how to order the point features that we find. This can be used to limit the number of matches that we must consider.

There are many aspects of grouping that are still poorly understood. We have mentioned that we have not explored other grouping cues, or the problem of integrating different cues. We have also not studied the use of some prior knowledge of what we expect to see in a scene, which might vary from situation to situation. In addition, we should stress that many problems remain in determining just how to use shape to perform grouping. There are many object parts that are not convex, such as the tail of a cat, or a banana. Convexity is only one salient shape; we do not have a good characterization of what makes a set of image edges appear to form a part of an object. Furthermore, even when using convexity for grouping, the role played by context is not well understood. We showed in chapter 6 that purely local methods of finding convex groups run into problems, missing good, globally salient groups. But it is also clear that the lines surrounding a convex group can affect its salience, and our approach does not fully take account of this.

We have attempted to support the idea that ambitious recognition problems are best handled with grouping and indexing by showing that this strategy is practical and useful in a simple domain. At the same time, by exploring indexing thoroughly in a simple domain, and by exploring a simple grouping clue thoroughly, we hope to create theoretical and practical tools that can help lead us to a solution to the larger problems of recognition. By characterizing the images that a model can produce, we have created a powerful new tool for understanding the advantages of and the limitations to various ways of describing an image so that we can remember the object that produced it.

8.2 Practical Object Recognition

In the previous section we traced the connections between this thesis and approaches to understanding the process of recognizing objects with the capabilities of a human. There are many less ambitious recognition problems of considerable practical value. We have shown that in these domains, simple grouping techniques and indexing using point features can combine to overcome some current difficulties.

It is quite computationally intensive to even recognize a single rigid 3-D object in a realistic image. Techniques for doing this are usually either slow or apply to a domain in which simple grouping or indexing methods are useful. By expanding the range of useful grouping and indexing techniques we can expand the range of application domains within which we can recognize objects. Some indexing systems have been applied to 3-D recognition, but actually use indexing to match planar parts of an image to planar parts of a model. Other 3-D indexing methods require large amounts of space, and may introduce errors. We have developed an indexing system that can handle arbitrary groups of 3-D points, and we have shown how to account for error in this system. Moreover, we have shown how to represent models for indexing in the most space-efficient possible way. This provides us with a method of indexing that should be more complete, more accurate, and more efficient than previous ones.

At the same time, there is certainly room for improvement in our basic system. As we have pointed out, since error can affect different image groups to varying extents, one should represent the index table at several different levels of discretization, to allow one to look in the table quickly with either large or small error regions. This is an implementation detail. A more challenging improvement to our system would be to more carefully account for the effects of image error. We simplify the problem by placing a rectanguloid about what is actually a more complicated error region. We have shown, however, that there is the potential to achieve greater speedups from the indexing system if we remove this simplification. These changes would be clear improvements to the basic system that we have presented.

This basic system relies on representing models' lines using a simple tessellation of index space. There are a number of ways that we might improve upon that method. First, if the number of model groups represented is not too great, it might be simpler and cheaper to just explicitly compare a group of image points to each group of model points. In that case, our representation gives us a very quick method of comparison; we need only find the distance between a point and a line in a high dimensional space to get a measure of the compatibility of an image and model group. Second, there is no reason to think that a simple tesselation of image space is the best way to represent image space. The problem of matching a rectangle to lines in a high-dimensional space has the familiar flavor of other computational geometry problems that have been solved more accurately and efficiently using other methods of representing a Euclidean space. We can also imagine that even if we want to tesselate the space, that it might prove more efficient, and sufficiently accurate, to represent lower-dimensional projections of the high-dimensional image spaces that we use. We have not explored these paths, however. Third, our system requires considerable space in order to account for partial occlusions of image groups, and uncertainties in the ordering of points in these groups. For example, if we form a pair of convex groups that each produce four feature points, we must consider thirty-two different orderings for these points. We might instead use a canonical ordering of points. An example of a canonical ordering for a different indexing method can be found in Clemens and Jacobs[32]. We might also try to find ways of representing groups of points so that we can quickly match them to subgroups found in the image, even when these subgroups are missing some of the model points due to occlusion. In general, the space requirements of our current system can be rather high because we must represent some permutations and subsets of each group of model points. So work aimed at limiting the need to represent all these variations on a single basic group could be of practical value.

If we could find more space-efficient methods of representing image space we could also hope to efficiently perform indexing with more complicated features whose manifolds may not decompose into 1-D submanifolds. For example, it seems that if we simply tessellate image space we will need large amounts of space to handle oriented point features. These features could be quite valuable, however. Vertices contain significantly more information about the object than do simple point features. Thompson and Mundy[100] have built an effective system using vertices, but the space that their system requires to represent even a small number of groups of vertex features is quite high.

Even oriented point features are relatively simple, and it might also be valuable to understand how to index more complicated image features. For example, it could be quite useful to determine how to represent the images that a 3-D curve can produce when viewed from all directions. We have analyzed non-planar models of points or oriented points by using invariant descriptions of planar models, and then characterizing the set of planar models that can produce the same affine invariant description as a single 3-D model. There are already invariant descriptions available for planar curves, but we do not know how to characterize the set of affine-invariant descriptions that a 3-D curve may produce. Then too, all of the above features assume that some fixed portion of a 3-D model will project to a corresponding image feature as the viewpoint changes. That is, these are all essentially wire-frame models. When objects have curved surfaces, different portions of the object create contours in the image as the viewpoint changes. So it would be particularly valuable to understand how to characterize the edges that a curved 3-D surface can produce from different viewpoints. That problem goes well beyond what we have done in this thesis, but might be accomplished using the same basic strategy of characterizing the affine invariant feature descriptions that a 3-D model may produce. This work seems essential if we are to efficiently recognize complex objects that do not contain some simple point features that can be reliably located in images.

Perhaps the biggest bottleneck in recognition systems lies in the grouping problem, and we are far from understanding how to build good general grouping systems. But it is not hard to build a useful grouping system for a limited domain, and any improvement in these methods widens the range of applications for our vision systems. For example, practical systems have been built that rely on grouping together vertices connected by a line, or that rely on finding parallelograms in an image. These systems are useful for locating objects that produce such groups, when occlusion is limited. We have attempted to push forward the range of objects that grouping can handle by finding general convex groups of lines. And we have focused on improving the robustness of grouping systems by optimizing a global criteria that measures the salience of these groups. Salient convexity will not be an effective grouping method for all objects or all types of images. But we are able to characterize when it will be effective by determining the level of salience that a group must possess for our system to be able to efficiently locate it.

One of the things that proves difficult in using grouping to recognize an object is that grouping is most effective when we may assume that all of the features in an image group come from the object for which we search. This can cause two types of problems. First, if a group is partially occluded, we must sort out which part of the group comes from the object for which we are looking, and which part comes from the occlusion. This is quite difficult, although Clemens[30] provides one example of such reasoning. The second problem is that even if grouping provides us with a set of edges that all come from a single object, we must reliably turn those edges into features. Simple methods of finding lines or vertices in edges may work when objects are completely polyhedral. But even real objects that appear polyhedral usually contain many curves. These can result in lines or vertices that appear or disappear due to changes in the viewpoint or due to small amounts of sensing error. We need a method of detecting local features that will find the same features from a set of edges regardless of error or changes in viewpoint. We have made some progress on this problem, but our methods could stand considerable improvement. And improved methods of finding local 2-D features robustly from the projections of 3-D models would be of value to many other approaches to recognition, as well as to stereo or motion systems.

8.3 A Final Word

In conclusion, we view this thesis as an initial formulation of a strategy for understanding how to recognize objects as well as humans do, including some concrete steps towards implementing that strategy. Often the best way to clarify a difficult problem is to attack it in a simple domain where some real understanding may be gained. This is only true, however, if we continue to ask the hard questions even as we answer some easier versions of them. For this reason, although this thesis has provided answers to some questions of practical importance, we want to stress the questions that are raised and perhaps brought into sharper focus by this thesis. The most important of these questions is: How can we describe an image so that this description can remind us of an object? In this thesis we have attempted to provide some tools that can help us to analyze different possible answers to this question.

Appendix A

Projective 3-D to 2-D Transformations

In Chapter 2 we show geometrically that when a group of 3-D points form an image under perspective projection, that the set of images they can produce must be represented by at least a 3-D surface in any index space. Here we derive a slightly stronger result algebraically. We show that for any 3-D model, three of the projective invariants of the model's images can take on any set of values. This appendix will rely on some knowledge of elementary analytic projective geometry. The interested reader may refer to many introductory books on geometry, including Tuller[102]. Our description of the analytic formulation of projection from 3-D to 2-D will closely follow Faugeras[42].

In projective geometry we analytically represent points in the plane using three coordinates, which we will call x, y and w. This representation has the property that (x_0, y_0, w_0) represents the same point as (x_1, y_1, w_1) if and only if there exists some non-zero value k such that $(x_0, y_0, w_0) = k(x_0, y_0, w_0)$. We similarly represent 3-d points using quadruples of coordinates, which we will call x, y, z and w.

A projective transformation can be defined as one which applies any perspective projection to a set of 3-D points, and then applies any 2-D projective transformation to the resulting 2-D points. In this case, a group of 3-D points, $p_1, p_2, ...p_n$ can produce a set of 2-D points, $q_1, q_2, ...q_n$ if and only if there exists a four by three matrix, M, and a set of scalars, $k_1, k_2, ...k_n$ such that, for all i, $k_i q_i = M p_i$. In brief, this is allowable because a 3-D projective transformation can map any five points to any other five points, while a 2-D transformation maps any four points to any other four points.

If we assume that there are no degeneracies in the model or image, then without loss of generality we may set: $p_1 = (1, 0, 0, 0), p_2 = (0, 1, 0, 0), p_3 = (0, 0, 1, 0), p_4 = (0, 0, 0, 1), p_5 = (1, 1, 1, 1)$, and $q_1 = (1, 0, 0), q_2 = (0, 1, 0), q_3 = (0, 0, 1), q_4 = (1, 1, 1)$.

The remaining points may take on any values, and we denote them as: $p_6 = (p_6^x, p_6^y, p_6^z, p_6^w)$, $q_5 = (q_5^x, q_5^y, q_5^w)$, $q_6 = (q_6^x, q_6^y, q_6^w)$.

This implies that the matrix M has the form:

$$M = \begin{pmatrix} k_1 & 0 & 0 & k_4 \\ 0 & k_2 & 0 & k_4 \\ 0 & 0 & k_3 & k_4 \end{pmatrix}$$

The values of k_1, k_2, k_3, k_4 can only be determined up to a multiplicative factor, because any two matrices that are identical up to a multiplicative factor will produce the same images, since two points are identical when their coordinates are multiples. So, without loss of generality we can set k_4 to 1, leaving three unknowns.

From the projection of the fifth and sixth points we find that:

$$\begin{array}{ccc} k_1 + 1 = k_5 q_5^x & k_2 + 1 = k_5 q_5^y & k_3 + 1 = k_5 q_5^w \\ k_1 p_6^x + p_6^w = k_6 q_6^x & k_2 p_6^y + p_6^w = k_6 q_6^y & k_3 p_6^w + p_6^w = k_6 q_6^w \end{array}$$

The question becomes, for a particular set of values for $p_6^x, p_6^y, p_6^z, p_6^w$, which provides all information about the projective shape of the model points, what values can be produced for $q_5^x, q_5^y, q_5^w, q_6^x, q_6^y, q_6^w$, which tells us the projective shape of the image points, given that $k_1, ...k_6$ can take on any values.

Except for degenerate cases, if we choose any values for $q_5^x, q_5^y, q_5^w, q_6^x, q_6^w$, the first four and the sixth of the above equations give us five independent linear equations with five unknowns. Therefore, we can find values of k_i to produce any set of values for these five image coordinates. The value of these image coordinates will in turn determine the value of q_6^y.

The fifth and sixth image points each give rise to two projective invariants, the values: $\frac{q_5^x}{q_5^w}, \frac{q_5^y}{q_5^w}, \frac{q_6^x}{q_6^w}$, and $\frac{q_6^y}{q_6^w}$. We have shown that any model can produce an image that has any values for three of these invariants, and that the model's structure, along with the values of three of these invariants will determine the value of the fourth.

Bibliography

[1] Asada H. and M. Brady, 1986, "The Curvature Primal Sketch," *IEEE Transactions on Patern Analysis and Machine Intelligence*, **8**(1):2-14.

[2] Ash, R., 1972, *Real Analysis and Probability*, Academic Press, New York.

[3] Ayache, N. and O. Faugeras, 1986, "HYPER: A New Approach for the Recognition and Positioning of Two-Dimensional Objects," *IEEE Transactions on Patern Analysis and Machine Intelligence*, **8**(1):44-54.

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

[5] Ballard, D.H., 1981, "Generalizing the Hough Transform to Detect Arbitrary Patterns," *Pattern Recognition*, **13**(2): 111–122.

[6] Barrow, H., J. Tenenbaum, R. Bolles, and H. Wolf, 1977, "Parametric Correspondence and Chamfer Matching: Two New Techniques for Image Matching," *Proceedings of the International Joint Conference on Artificial Intelligence*:659-663.

[7] Bennett, B., D. Hoffman, and C. Prakash, 1989, *Observer Mechanics*, Academic Press, San Diego.

[8] Bergevin, R. and Levine, M., 1993, "Generic Object Recognition: Building and Matching Coarse Descriptions from Line Drawings," *IEEE Transactions on Patern Analysis and Machine Intelligence*, **15**(1):19-36.

[9] Biederman, I., 1985, "Human Image Understanding: Recent Research and a Theory," *Computer Graphics, Vision, and Image Processing*, (32):29-73.

[10] Binford, T., 1971, "Visual Perception by Computer," *IEEE Conference on Systems and Control*.

[11] Binford, T., 1981, "Inferring Surfaces from Images," *Artificial Intelligence*, **17**:205-244.

[12] Boldt, M. R. Weiss, and E. Riseman, 1989, "Token-Based Extraction of Straight Lines," *IEEE Transactions on Systems, Man and Cybernetics*, **19**(6):1581-1594.

[13] Bolles, R. and R. Cain, 1982, "Recognizing and Locating Partially Visible Objects: The Local-Feature-Focus Method," *The International Journal of Robotics Research*, **1**(3):57-82.

[14] Borgefors, G., 1988, "Hierarchical Chamfer Matching: A Parametric Edge Matching Algorithm," *IEEE Transactions on Patern Analysis and Machine Intelligence*, **10**(6):849-865.

[15] Bowyer, K. and C. Dyer, 1990, "Aspect Graphs: An Introduction and Survey of Recent Results," *International Journal of Imaging Systems and Technology*, **2**:315-328.

[16] Brady, M. and Yuille, A., 1983, "An Extremum Principle for Shape from Contour," *Proceedings of the 8th International Joint Conference on Artificial Intelligence*:969-972.

[17] Breuel, T., 1990, "Indexing for Visual Recognition from a Large Model Base," MIT AI Memo 1108.

[18] Breuel, T., personal communication.

[19] Breuel, T., 1991, "Model Based Recognition using Pruned Correspondence Search," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 257–268.

[20] Breuel, T., 1992, *Geometric Aspects of Visual Object Recognition*, MIT AI TR–1374.

[21] Brooks, R., 1981, "Symbolic Reasoning Among 3-D Models and 2-D Images," *Artificial Intelligence*, **17**:285-348.

[22] Burns, J., R. Weiss, and E. Riseman, 1990, "View Variation of Point Set and Line Segment Features," *DARPA IU Workshop*:650-659.

[23] Burns, J. and E. Riseman, 1992, "Matching Complex Images to Multiple 3D Objects using View Description Networks," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 328–334.

[24] Burns, J., R. Weiss, and E. Riseman, 1992, "The Non-Existence of General-Case View-Invariants," *Geometric Invariance in Computer Vision*, edited by J. Mundy, and A. Zisserman, MIT Press, Cambridge.

[25] Canny, J., 1986, "A Computational Approach to Edge Detection," *IEEE Transactions on Patern Analysis and Machine Intelligence*, **8**(6):679-698.

[26] Cass, T., 1992, "Polynomial Time Object Recognition in the Presence of Clutter, Occlusion and Uncertainty," *Second European Conference on Computer Vision*:834-842.

[27] Cass, T., 1992, *Polynomial Time Geometric Matching for Object Recognition*, PhD Thesis, MIT Department of Electrical Engineering and Computer Science.

[28] Chen, C. and A. Kak, 1989, "A Robot Vision System for Recognizing 3-D Objects in Low-Order Polynomial Time," *IEEE Transactions on Systems, Man, and Cybernetics*, **19**(6):1535-1564.

[29] Clemens, D., 1986, *The Recognition of Two-Dimensional Modeled Objects in Images*, Master's Thesis, MIT Department of Electrical Engineering and Computer Science.

[30] Clemens, D., 1991, *Region-Based Feature Interpretation for Recognizing 3D Models in 2D Images*, MIT AI TR–1307.

[31] Clemens, D. and D. Jacobs, 1990, "Model-Group Indexing for Recognition," *DARPA IU Workshop*:604-613.

[32] Clemens, D. and Jacobs, D., 1991, "Space and Time Bounds on Model Indexing," *IEEE Transactions on Patern Analysis and Machine Intelligence*, **13**(10):1007-1018.

[33] Clowes, M., 1971, "On Seeing Things," *Artificial Intelligence*, **2**:79-116.

[34] Costa, M., R.M. Haralick, and L.G. Shapiro, 1990, "Optimal Affine-Invariant Point Matching," *Proceedings 6th Israel Conference on AI*, pp. 35–61.

[35] Cox, I., J. Kruskal, and D. Wallach, 1990, "Predicting and Estimating the Accuracy of a Subpixel Registration Algorithm," *IEEE Transactions on Patern Analysis and Machine Intelligence*, **12**(8):721-734.

[36] Cox, I., J. Rehg, and S. Hingorani, 1992, "A Bayesian Multiple Hypothesis Approach to Contour Grouping," *European Conference on Computer Vision*, pp. 72–77.

[37] Cutting, J., 1986, *Perception with an Eye for Motion*, MIT Press, Cambridge.

[38] Cyganski, D. and J.A. Orr, 1985, "Applications of Tensor Theory to Object Recognition and Orientation Determination," *IEEE Transactions on Patern Analysis and Machine Intelligence*, **7**(6):662–673.

[39] Cyganski, D., J. Orr, T. Cott, and R. Dodson, 1987, "Development, Implementation, Testing, and Application of an Affine Transform Invariant Curvature Function," *Proceedings of the First International Conference on Computer Vision*:496-500.

[40] Denasi, S., G. Quaglia, and D. Rinaudi, 1991, "The use of Perceptual Organization in the Prediction of Geometric Structures," *Pattern Recognition Letters*, **13**(7):529-539.

[41] Dolan, J. and E. Riseman, 1992, "Computing Curvilinear Structure by Token-based Grouping," *IEEE Conference Computer Vision and Pattern Recognition*, pp. 264–270.

[42] Faugeras, O., 1992, "What can be Seen in Three Dimensions with an Uncalibrated Stereo Rig?" *Second European Conference on Computer Vision*:563-578.

[43] Fischler, M. and R. Bolles, 1981, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Analysis and Automated Cartography," *Communications of the Association of Computing Machinery*, **24**(6):381-395.

[44] Forsyth, D., J.L. Mundy, A. Zisserman, C. Coelho, A. Heller, and C. Rothwell, 1991, "Invariant Descriptors for 3-D Object Recognition and Pose", *IEEE Transactions on Patern Analysis and Machine Intelligence*, **13**(10):971–991.

[45] Gigus, Z., J. Canny, and R. Seidel, 1991, "Efficiently Computing and Representing Aspect Graphs of Polyhedral Objects," *IEEE Transactions on Patern Analysis and Machine Intelligence*, **13**(6):542-551.

[46] Goad, C., 1983, "Special Purpose Automatic Programming for 3D Model-Based Vision," *Proceedings ARPA Image Understanding Workshop.*: 94-104.

[47] Grimson, W.E.L., 1990, *Object Recognition by Computer: The role of geometric constraints*, MIT Press, Cambridge.

[48] Grimson, W.E.L. and D.P. Huttenlocher, 1990, "On the Sensitivity of Geometric Hashing," *Proceedings Third International Conference Computer Vision*, pp. 334–338.

[49] Grimson, W.E.L., D.P. Huttenlocher, and D. Jacobs, 1991, "Affine Matching With Bounded Sensor Error: A Study of Geometric Hashing & Alignment," MIT AI Lab Memo 1250.

[50] Grimson, W.E.L. and T. Lozano-Pérez, 1987, "Localizing Overlapping Parts by Searching the Interpretation Tree," *IEEE Transactions on Patern Analysis and Machine Intelligence* **9**(4):469-482.

[51] Guzman, A., 1969, "Decomposition of a Visual Scene into Three-Dimensional Bodies," *Automatic Interpretation and Classification of Images*, edited by A. Grasseli, Academic Press, New York.

[52] Hoffman, D. and Richards, W., 1984, "Parts of Recognition," *Visual Cognition*, edited by S. Pinker, MIT Press, Cambridge.

[53] Horn, B., 1986, *Robot Vision*, MIT Press, Cambridge.

[54] Huang, T., 1991, "Reply Computer Vision Needs More Experiments and Applications," *Computer Vision, Graphics and Image Processing*, **53**(1):125-126.

[55] Huffman, D., 1971, "Impossible Objects as Nonsense Sentences," *Machine Intelligence 6*, edited by Meltzer, B. and D. Michie, Edinburgh University Press, Edinburgh.

[56] Huttenlocher, D., 1988, *Three-Dimensional Recognition of Solid Objects from a Two-Dimensional Image*, MIT AI Lab Technical Report 1045.

[57] Huttenlocher, D. and S. Ullman, 1990, "Recognizing Solid Objects by Alignment with an Image," *International Journal of Computer Vision*, **5**(2):195–212.

[58] Huttenlocher, D. and Wayner, P., 1992, "Finding Convex Edge Groupings in an Image," *International Journal of Computer Vision*, **8**(1):7-29.

[59] Jacobs, D., 1987, "GROPER: A Grouping Based Object Recognition System for Two-Dimensional Objects," *IEEE Workshop on Computer Vision*, pp. 164-169.

[60] Jacobs, D., 1989, "Grouping for Recognition," MIT AI Memo 1177.

[61] Jacobs, D., 1991, "Optimal Matching of Planar Models in 3D Scenes," *IEEE Conference Computer Vision and Pattern Recognition*, pp. 269–274.

[62] Jacobs, D., 1992, "Space Efficient 3D Model Indexing," *IEEE Conference Computer Vision and Pattern Recognition*, pp. 439–444.

[63] Kalvin, A., E. Schonberg, J. Schwartz, and M. Sharir, 1986, "Two-Dimensional, Model-Based, Boundary Matching Using Footprints." *The International Journal of Robotics Research*, **5**(4):38-55.

[64] Kanade, T., 1981, "Recovery of the Three-Dimensional Shape of an Object from a Single View," *Artificial Intelligence*, **17**:409-460.

[65] Koenderink, J. and van Doorn, A., 1991, "Affine Structure from Motion," *Journal of the Optical Society of America*, **8**(2):377-385.

[66] Koenderink, J. and van Doorn, A., 1976, "The Singularities of the Visual Mapping," *Biological Cybernetics*, **24**:51-59.

[67] Kohler, W., 1959, *Gestalt Psychology*, Mentor Books, New York.

[68] Korn, G.A. & T.M. Korn, 1968, *Mathematical Handbook for Scientists and Engineers*, McGraw-Hill, New York.

[69] Kriegman, D. and J. Ponce, 1990, *International Journal of Computer Vision*, **5**(2):119–136.

[70] Lamdan, Y., J.T. Schwartz and H.J. Wolfson, 1990, "Affine Invariant Model-Based Object Recognition," *IEEE Transactions Robotics and Automation*, **6**:578–589.

[71] Lamdan, Y. & H.J. Wolfson, 1988, "Geometric Hashing: A General and Efficient Model-Based Recognition Scheme," *Second International Conference Computer Vision*, pp. 238–249.

[72] Lamdan, Y. & H.J. Wolfson, 1991, "On the Error Analysis of 'Geometric Hashing'," *IEEE Conference Computer Vision and Pattern Recognition*, pp. 22–27.

[73] Lowe, D., 1985, *Perceptual Organization and Visual Recognition*, Kluwer Academic Publishers, The Netherlands.

[74] Mahoney, J., 1987. *Image Chunking: Defining Spatial Building Blocks for Scene Analysis*, MIT AI TR–980.

[75] Malik, J., 1987, "Interpreting Line Drawings of Curved Objects," *International Journal of Computer Vision*, **1**(1):73-103.

[76] Marimont, D., 1984, "A Representation for Image Curves," *AAAI*:237-242.

[77] Marr, D., 1977, "Artificial Intelligence - A Personal View," *Artificial Intelligence*, **9**:37-48.

[78] Marr, D., 1982, *Vision*, W.H. Freeman and Company, San Francisco.

[79] Marr, D. and Nishihara, H., 1978, "Representation and Recognition of the Spatial Organization of Three Dimensional Structure," *Proceedings of the Royal Society of London B*, **200**:269-294.

[80] Marill, T., 1991, "Emulating the Human Interpretation of Line-Drawings as Three-Dimensional Objects," *International Journal of Computer Vision*, **6**(2) 147-161.

[81] Mohan, R. and Nevatia, R., 1989, "Using Perceptual Organization to Extract 3-D Structures," *IEEE Transactions on Patern Analysis and Machine Intelligence*, **11**(11):1121-1139.

[82] Mokhtarian, F. and A. Mackworth, 1986, *IEEE Transactions on Patern Analysis and Machine Intelligence*, **8**(1):34-43.

[83] Moses, Y. and Ullman, S., 1991, "Limitations of Non Model-Based Recognition Schemes," MIT AI Memo 1301.

[84] Moses, Y. and Ullman, S. 1992, "Limitations of Non Model-Based Recognition Schemes," *Second European Conference on Computer Vision*:820-828.

[85] *Geometric Invariance in Computer Vision*, edited by J. Mundy, and A. Zisserman, MIT Press, Cambridge.

[86] Nalwa, V., 1988, "Line-Drawing Interpretation: A Mathematical Framework," *International Journal of Computer Vision*, **2**(2) 103-124.

[87] Pavlidis, T. and S. Horowitz, 1974, "Segmentation of Plane Curves," *IEEE Transactions on Computers*, **C**(23):860-870.

[88] Pentland, A., 1987, "Recognition by Parts." *Proceedings of the First International Conference on Computer Vision*:612-620.

[89] Poggio, T., 1990, "3D Object Recognition: On a Result of Basri and Ullman," Istituto Per La Ricerca Scientifica E Tecnologica IRST Technical Report 9005-03.

[90] Richards, W. and A. Jepson, 1992, "What Makes a Good Feature?" MIT AI Memo 1356.

[91] Roberts, L., 1966, "Machine Perception of Three-Dimensional Solid Objects," *Optical and Electro-optical Information Processing*, edited by J. Tippett, MIT Press, Cambridge.

[92] Rothwell C., A. Zisserman, J. Mundy, and D. Forsyth, 1992, "Efficient Model Library Access by Projectively Invariant Indexing Functions," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 109-114.

[93] Saund, E., 1992, "Labeling of Curvilinear Structure Across Scales by Token Grouping," *IEEE Conference Computer Vision and Pattern Recognition*, pp. 257–263.

[94] Schwartz, J. and M. Sharir, 1987, "Identification of Partially Obscured Objects in Two and Three Dimensions by Matching Noisy Characteristic Curves," *The International Journal of Robotics Research*, **6**(2):29-44.

[95] Shashua, A., 1991, "Correspondence and Affine Shape from Two Orthographic Views: Motion and Recognition," MIT AI Memo 1327.

[96] Sha'ashua, A. and Ullman, S., 1988, "Structural Saliency: The Detection of Globally Salient Structures Using a Locally Connected Network," *IEEE International Conference on Computer Vision*:321-327.

[97] Sinha, P., 1992, *The Perception of Shading and Reflectance*, Master's Thesis, MIT Department of Electrical Engineering and Computer Science.

[98] Stein, F. and G. Medioni, 1992, "Structural Indexing: Efficient 3-D Object Recognition," *IEEE Transactions on Patern Analysis and Machine Intelligence*, **14**(2):125-145.

[99] Syeda-Mahmood, T., 1992, "Data and Model-Driven Selection using Color Regions," *Second European Conference on Computer Vision*:115-123.

[100] Thompson, D. & J.L. Mundy, 1987, "Three-Dimensional Model Matching From an Unconstrained Viewpoint", *Proceedings IEEE Conference Rob. Aut.*, pp. 208-220.

[101] Tucker, L., Feynman, C., and Fritsche, D., 1988, "Object Recognition Using the Connection Machine," *Proceedings of Conference ON Computer Vision AND Pattern Recognition*:871-878.

[102] Tuller, A., 1967, *A Modern Introduction to Geometries*, D. Van Nostrand Company, Inc., Princeton.

[103] Ullman, S., 1979, *The Interpretation of Visual Motion*, MIT Press, Cambridge.

[104] Ullman, S., 1989, "Aligning Pictorial Descriptions: An Approach to Object Recognition," *Cognition* **32**(3):193-254.

[105] Ullman, S. and Basri, R., 1991, "Recognition by Linear Combinations of Models," *IEEE Transactions on Patern Analysis and Machine Intelligence*, **13**(10):992-1007.

[106] Van Gool, L., P. Kempenaers & A. Oosterlinck, 1991, "Recognition and Semi-Differential Invariants," *IEEE Conference Computer Vision and Pattern Recognition*, pp. 454–460.

[107] Wallace, A., 1987, "Matching Segmented Scenes to Models Using Pairwise Relationships Between Features," *Image and Vision Computing*, **5**(2):114-120.

[108] Waltz, D., 1975, "Understanding Line Drawings of Scenes with Shadows," *The Psychology of Computer Vision*, edited by Winston, P., McGraw-Hill, New York.

[109] Wayner, P.C., 1991, "Efficiently Using Invariant Theory for Model-based Matching," *IEEE Conference Computer Vision and Pattern Recognition*:473-478.

[110] Weinshall, D., 1991, "Model Based Invariants for Linear Model Acquisition and Recognition," IBM Research Report RC-17705 (#77262).

[111] Weiss, I., 1988, "Projective Invariants of Shape," *DARPA IU Workshop*, pp. 1125–1134.

[112] Weiss, I., 1992, "Noise Resistant Projective and Affine Invariants," *IEEE Conference Computer Vision and Pattern Recognition*:115-121.

[113] Wertheimer, M., 1938, "Laws of Organization in Perceptual Form," *A Source Book of Gestalt Psychology*, edited by Ellis, W., Harcourt, Brace and Company, New York.

[114] Willard, S., 1970, *General Topology*, Addison-Wesley, Reading, MA.

[115] Witkin, A. and J. Tenenbaum, 1983, "On the Role of Structure in Vision," In *Human and Machine Vision*, edited by Beck, Hope, and Rosenfeld. Academic Press, New York.

[116] Zucker, S., 1983, "Cooperative Grouping and Early Orientation Selection," In *Physical and Biological Processing of Images*, edited by Braddick and Sleigh. Springer-Verlag, Berlin.