# Grouping for Recognition[1]

David W. Jacobs
(dwj@research.nj.nec.com)

NEC Research Institute
4 Independence Way
Princeton, NJ 08540

Abstract

This paper presents a new method of grouping edges in order to recognize objects. This grouping method succeeds on images of both two- and three-dimensional objects. We order groups of edges based on the likelihood that a single object produced them. This allows the recognition system to consider first the collections of edges most likely to lead to the correct recognition of objects. The grouping module estimates this likelihood using the distance that separates edges and their relative orientation. This ordering greatly reduces the amount of computation required to locate objects. Surprisingly, in some circumstances grouping can also improve the accuracy of a recognition system. We test the grouping system in two ways. First, we use it in a recognition system that handles libraries of two-dimensional, polygonal objects. Second, we show comparable performance of the grouping system on images of two- and three-dimensional objects. This demonstrates that the grouping system could produce significant improvements in the performance of a three-dimensional recognition system.

## 1 Introduction

This paper discusses the use of grouping to improve the performance of existing approaches to model-based visual object recognition. We present a data-driven method for forming groups of image edges especially likely to all come from a single object. This allows us to order our search for objects in an image. Our experiments indicate that this ordering is sufficient to reduce the search for objects by several orders of magnitude.

Recent results in the field of object recognition demonstrate the necessity of some type of grouping, or selection, to make the combinatorics of object recognition manageable. Grimson[13] and Grimson and Huttenlocher[14] have recently shown that in the domain of two-dimensional objects, constrained search methods, and Hough transform methods break down in cluttered scenes without some type of grouping. In the domain of three-dimensional objects matched to two-dimensional images, the fastest known methods without grouping are $O(n^4m^4)$, where there are $n$ image features and $m$ model features. This expense is excessive in realistic domains. In fact to overcome these problems, virtually all recognition systems make use of some

[1] Essentially the same paper, with a few fewer references, appeared as MIT AI Memo 1177, 1989.

implicit grouping, using cues such as distance (Bolles and Cain[3]), convexity (Brooks[4]), or continuity (Huttenlocher and Ullman[16]). Grouping overcomes the unfavorable combinatorics of recognition by removing the need to search the space of all matches between image and model features. Only those image features considered likely to come from a single object must be included together in hypothetical matches. And these groups need only be matched with compatible groups of model features.

Grouping may be either model-driven or data-driven. Acronym[4], for example, used a model-driven approach to group edges that might be the projection of a simplified class of generalized cylinders. Mohan and Nevatia[28] group together rectangular edges in a system that performs stereo matching of images of rectangular buildings. And Grimson and Lozano-Pérez[11] used a Hough transform to perform grouping based on a specific model. Model-driven approaches, however, are often limited to handling only objects composed of a specific set of primitives. And approaches such as the Hough transform used in Grimson and Lozano-Pérez do not apply when libraries of objects are used. Data-driven approaches have the advantage that they may apply to more general classes of objects, and scale well when used with libraries of objects.

Lowe[25] has previously explored data-driven grouping in a recognition system. His system, Scerpo, combines parallel, symmetric, or co-terminating edges. Our current work owes much to Lowe's general approach to recognition. However, his approach to grouping has some serious limitations. First of all, these simple cues may not be present in many images. Second, forming only small groups of edges limits the amount that we may reduce search. Our current research demonstrates that forming large groups of image edges can result in dramatic reductions in search.

Since we do not attempt a perfect, bottom-up segmentation of an image into its component parts we may rely on the interaction between a system that forms likely groups and a model-based recognition system to speed up the recognition process even when we form many incorrect groups.

The heart of our grouping system consists of a method of estimating the probability that two convex contours came from the same object. This method uses the distance separating the two contours and their relative orientation. It compares the likelihood that this distance and orientation would occur between two randomly oriented contours from different objects to the likelihood of this distance and orientation occurring between two contours from the same object. This produces an ordering on all pairs of contours.

It is important to emphasize that these probabilities need not be exact to be useful. They allow us to perform an ordered search instead of an unordered search, and so the probability estimates are useful to the extent that they are superior to the uniform estimates implicit in an unordered search.

However, ordering the space of sets of matches between image and model features is only useful if we commit ourselves to an acceptable match as soon as we find it. Otherwise, the entire space of matches must still be searched, and ordering the search does no good. This search termination heuristic is in fact commonly used to reduce the cost of recognition.

Figures 1 through 4 show the performance of our grouping system on images of two- and

three-dimensional objects. In Figures 1 and 2 the grouping system combines with a recognition system to find five two-dimensional objects after considering only fifteen different groups of image edges. This contrasts markedly with most recognition systems, which might search through thousands of combinations of edges before locating an object.

Figures 3 and 4 show the performance of the grouping system alone on an image of three-dimensional objects. We have not implemented a 3d recognition system. However, we can see that the grouping system generates similar, useful groups when applied to this image. This suggests that our grouping system can produce comparable speed-ups in the recognition of 3d objects.

The next section discusses some previous research on grouping and object recognition. Section 3 will describe the grouping system used in GROPER, and provide some theoretical justification for it. Section 4 will briefly discuss GROPER's recognition component. Finally, Section 5 will provide the results of tests of the system.

Jacobs[18] presents GROPER in more detail, but with fewer experimental results. It also discusses the psychophysical implications of this work.

## 2 Previous Work

Psychologists did much influential research on grouping before its recent use in machine vision. Lowe[25] provides a useful summary of this work, and Kohler[22] discusses the origins of this work by the Gestalt psychologists.

Witkin and Tenenbaum[42],[43], suggested the use of grouping in a computational framework. They argue that when sections of an image share properties unlikely to occur by chance, this indicates a probable causal connection. For example, two irregular but parallel curves in an image might be tracks from a rake, but are unlikely to be unrelated.

Lowe has successfully incorporated grouping into an object recognition system. His system, SCERPO, capitalizes on two advantages of grouping. First, he groups together edges thought particularly likely to come from the same object. For example, Lowe argues that nearby parallel edges will more often come from the same object than from randomly oriented edges. Second, SCERPO looks for groups of edges that have some property invariant with the camera viewpoint. For example, three edges that terminate at a vertex will do so regardless of an object's orientation. So, SCERPO needs to try matching three co-terminating image edges only to triples of model edges that co-terminate. Such matches generate relatively few hypothesized object positions.

This paper discusses a generalization of this approach to grouping. Like Lowe, we attempt to analyze the image formation process to determine which groups of edges will have the greatest likelihood of coming from a single object. We extend Lowe's work by providing a general grouping system that can form arbitrarily large groups of edges that do not necessarily have some special property such as parallelism or co-termination.

Many other recognition systems have used more ad-hoc methods of grouping than Lowe's. We will discuss two types of recognition systems, constrained search and alignment. Constrained search recognizes objects with a backtracking search through the space of all mappings between

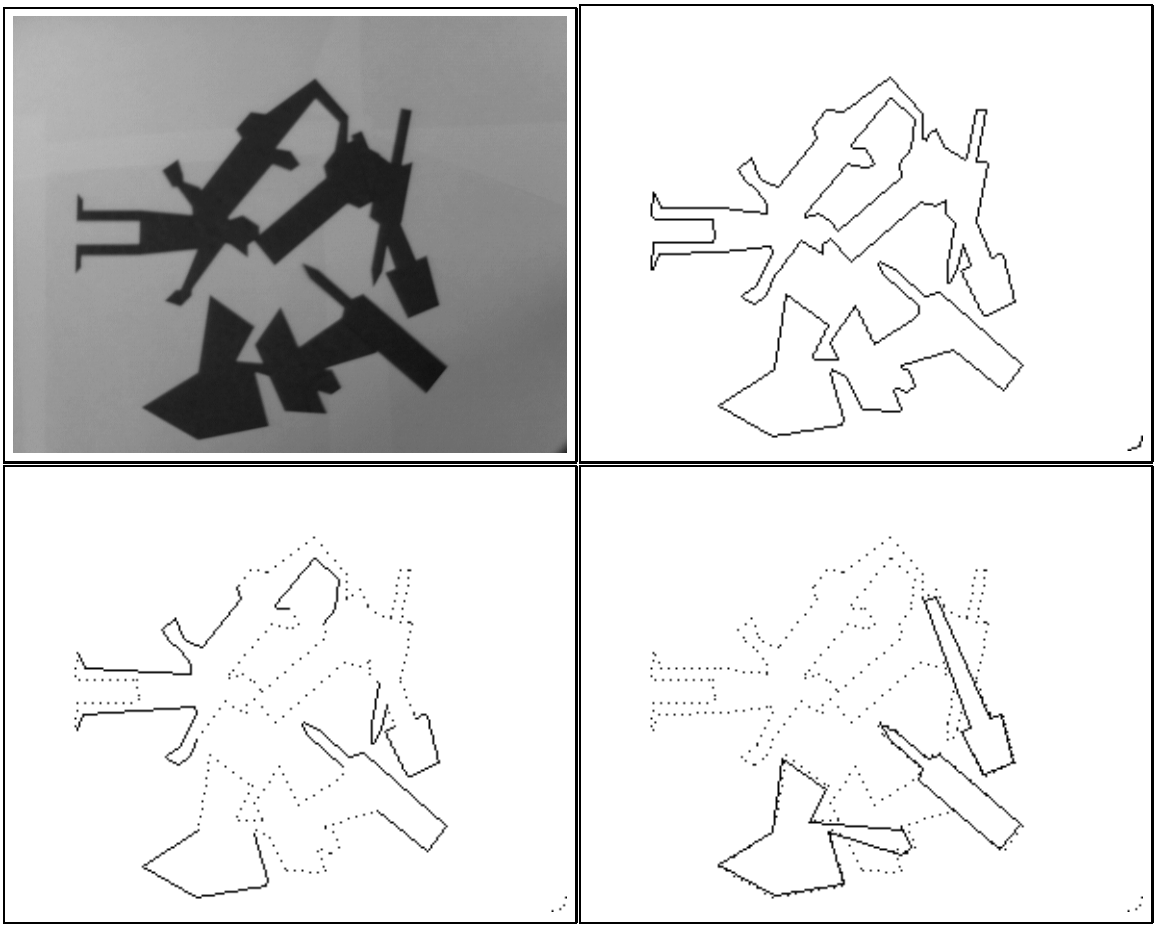

Figure 1: Recognition using grouping. In the upper left corner is a picture of objects cut out of paper. The upper right shows line approximations to the edges found in the image. The lower left shows the groups of convex edges found that contain four or more lines longer than ten pixels. The lower right shows the three objects found by indexing into a data base of sixteen objects (shown in Figure 13) using individual groups. Some big groups contain edges that all come from a single object, but do not lead directly to the recognition of that object. That may be because those edges do not provide enough information or contain edges detected with too much error. (continued in Figure 2.)

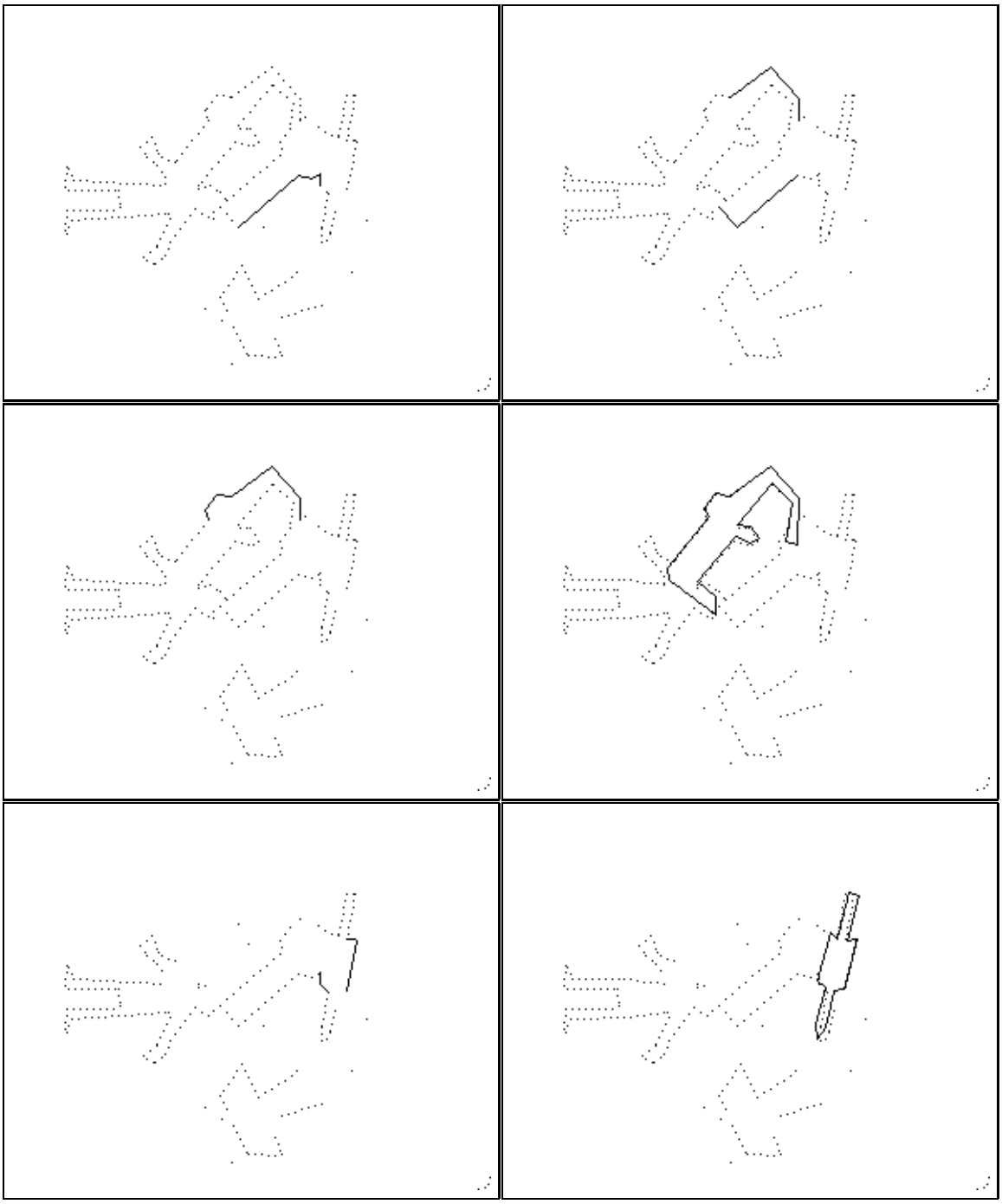

Figure 2: Next, the grouping system selects pairs of convex groups of edges for indexing. The top pictures show the first two pairs chosen. The next two pairs each match a single object. So, after indexing with eleven large groups of edges, and four pairs of convex groups, the system has found five of the eight objects in the image.

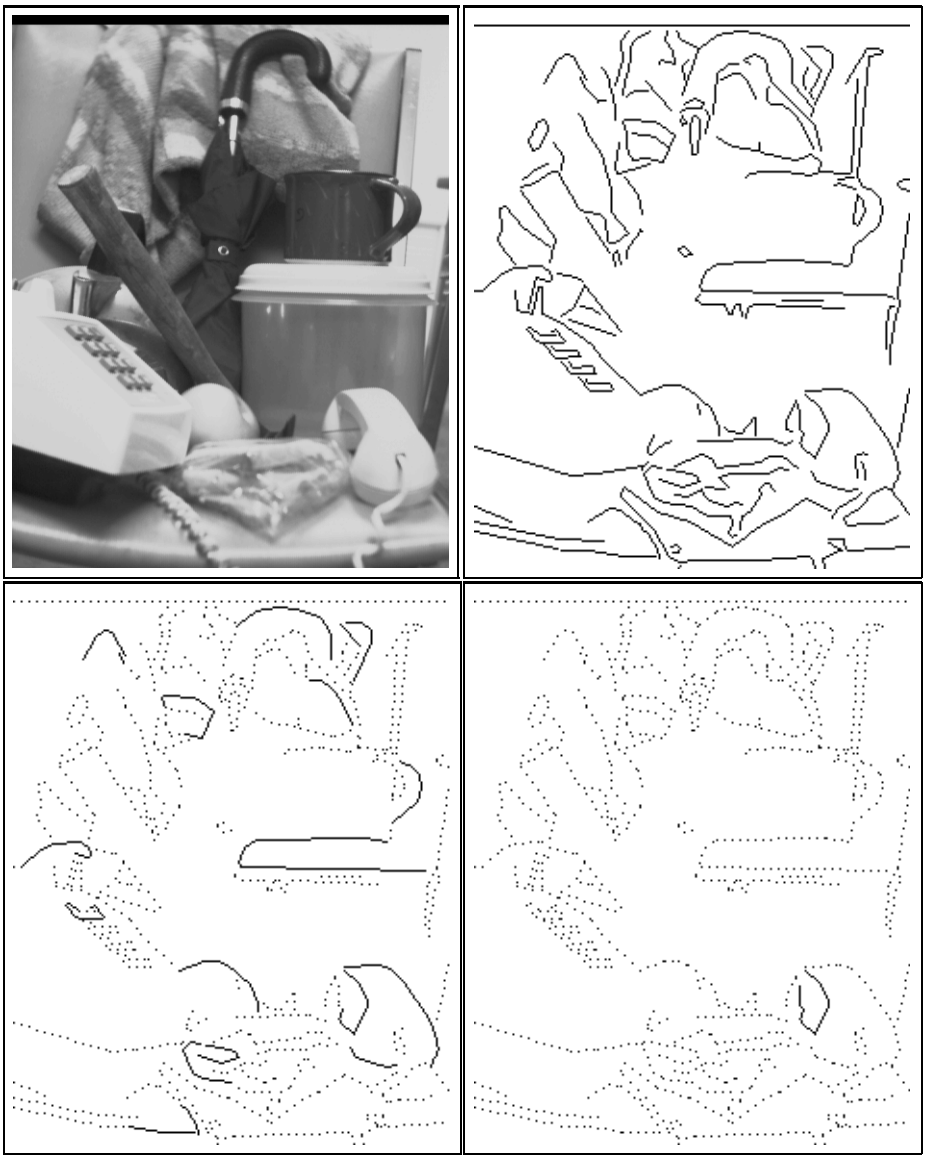

Figure 3: These pictures show comparable performance on a complex image of three-dimensional objects. Our system cannot recognize these objects, so we simply show the groups of edges selected by grouping. In the upper right are line approximations to the edges found in the image on the left. The lower left shows the convex groups that contain four or more edges, including groups from the telephone, apple, umbrella, mug and tupperware. On the lower right hand, and in Figure 4, we show the first five pairs of convex groups of edges chosen by the grouping module.

Figure 4: The second through fifth pairs of convex groups chosen by the grouping system, from the image in Figure 3. The groups on the left each come from a single object, the hammer and the phone cord. The group on the upper right has edges from the sandwich and phone cord. The edges in the lower right come from the folds and stripes of the sweater.

image and model features, looking for consistent ones. Some systems may explicitly calculate a transformation of the model that best aligns it with the image features at each node of the search tree. Or, they may save time by only looking at the pairwise consistency of matched features. Goad[10], Bolles and Cain[3], Brooks[4], Grimson and Lozano-Pérez[11], Van Hove[38] and others perform constrained search.

Grouping may tell us which parts of the search tree to explore first, or allow us to prune sections of the tree in advance. For example, Grimson and Lozano-Pérez cluster matches between image and model features into groups using the Hough transform. This allows them to start with the most promising clusters, and to avoid some collections of matches that could not lead to a correct answer. Bolles and Cain develop hypotheses by only considering sets of matches in which all the image features are nearby. Brooks groups together edges that all appear to come from the same generalized cylinder, partly to reduce the amount of search needed.

Grimson[13] has shown the importance of grouping to constrained search. He shows that the expected complexity of his system, using edges that all come from one object, is quadratic. But the complexity of recognizing objects in an unsegmented, cluttered environment is exponential in the size of the correct interpretation. Grimson and Huttenlocher[14] similarly show that the Hough transform becomes inaccurate in cluttered domains without grouping.

Alignment approaches work by matching a few image features to model features. Each match suggests an hypothesis about the location of the object model in the image. Some systems then perform a verification step on each hypothesis. Ayache and Faugeras[1], Lowe[25], and Huttenlocher and Ullman[16] do this. Other systems cluster the hypotheses and pick the location of the object that the most hypotheses support. Thompson and Mundy's[34], and Tucker et. al.'s[35] systems takes this approach.

As previously mentioned, Lowe uses grouping to focus on only some collections of image edges, not considering others. Huttenlocher and Ullman[16] also use some grouping to focus first on collections of image features that seem likeliest to come from the same object.

Past systems have used these approaches to grouping because of the speed-ups that grouping brings. Grouping becomes even more important as we address more complex domains. The recognition of non-rigid objects requires greater search than the recognition of rigid objects, as Grimson[12] and Ullman[37] discuss, and using a library of objects requires more search than looking for a single object. However, grouping need not grow any more difficult under these conditions, because a bottom-up approach, like GROPER's, does not depend on the properties of the objects sought.

More recently, based on this work, Huttenlocher and Wayner[17] have implemented another grouping system based on convexity. They suggest an improved method of finding the simple convex strings which we describe in Section 3.1. They employ a Delauney triangulation to find these simple convex groups more efficiently. Their approach also allows different criteria than proximity to be used in determining which simple groups to find.

Shashua and Ullman[33] present a system that finds curves that globally minimize a weighted sum of the total curvature of the curve and the total length of gaps in the curve. Along these lines, a number of other systems attempt to extract meaningful curve segments from an image.

Mahoney[26] 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[7] 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[44] and Dolan and Riseman[8]) 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[2]), or cocircularity (Saund[31]).

GROPER's recognition system uses groups to index into libraries of objects. Several previous systems have also addressed this problem, including Knoll and Jain[21], Kalvin et. al. [20], Turney, Mudge and Volz[36] and Wallace[39]. Wallace's indexing system closely resembles GROPER's, but does not take error or occlusion into account. Kalvin et. al.[20] have used an indexing system developed by Schwartz and Sharir[32] that handles distinctively curved, continuous sections of object contours, but does not combine information from unconnected sections of an object's perimeter. There has also been a good deal of work recently on using indexing to recognize planar or 3-D objects in 3-D scenes, including: Lamdan, Schwartz and Wolfson[23], Lamdan and Wolfson[24], Weiss[41], Forsyth et al.[9], Clemens and Jacobs[6], Jacobs[19] and Wayner[40].

# 3 How GROPER performs Grouping

This section describes how GROPER decides on the likelihood of a particular group of edges coming from a single object. But we first outline GROPER's recognition system to explain what tasks the grouping module must perform.

GROPER begins by locating edges in an image using the Canny[5] or Marr-Hildreth[27] edge detectors, and approximating these edges with lines. Next, it finds connected, or nearly connected lines that form convex curves. Convex groups are likely to come from a single object, and provide useful primitives for further grouping. Some convex groups may allow direct recognition of an object, but most are not distinctive. So GROPER calculates a measure of the likelihood that a single object produced the edges in each pair of simple groups. From this GROPER determines the order in which it considers these pairs. Sometimes a pair of simple groups may match a large number of known objects. In this case, GROPER needs to know about more groups that go well with these two.

When GROPER finds an object it removes from future consideration all image edges matching that object. This simplifies the grouping needed to find additional objects.

This approach to recognition means that grouping must perform three tasks. It must find simple, convex contours. It must determine how likely any pair of convex contours is to come from the same object. And it must extend these pairs of contours when necessary, by adding additional convex contours.

The fact that GROPER uses grouping as part of a search has an important implication. Grouping need not work perfectly. If GROPER tries a pair of convex contours that do not

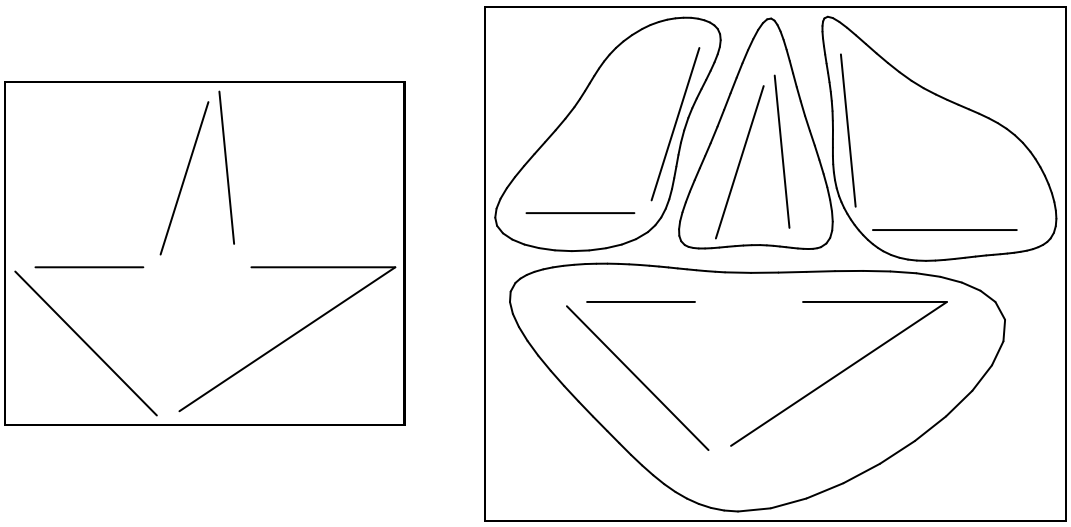

Figure 5: Left: some straight lines. Right: the convex groups they form, circled, and offset slightly from their original position.

come from the same object, this group will not lead to a verified hypothesis, and GROPER will try another. We aim to reduce, not eliminate search.

### 3.1 Simple Groups

GROPER forms all convex groups of line segments that meet two criteria: The lines in a group must have been producible by a convex curve, allowing for a small amount of error. And a line segment is attached to a group only when one of its end points is closer to an end point in the group than to the end point of any line segment outside the group. This bridges small gaps produced by the edge detector. Some edges will appear in two simple groups, and groups may have a single edge. Furthermore, assuming that groups comes from convex sections of a objects tells us on which side of a group the object lies.

GROPER will form simple groups in which figure and background are reversed, and not all the edges come from the same object, when one object occludes another. This does not present a problem as long as each edge also appears in a group with edges from the right object. Occasionally GROPER fails to put an edge in any correct groups, which means that GROPER can not use that edge to help it recognize an object. Figure 5 shows an example of some line segments and the simple convex groups that result from them.

### 3.2 Combining Simple Groups

GROPER estimates the likelihood that two convex groups came from the same object, using the distance between two groups, and their relative orientation. By distance, we mean the

minimum length of object perimeter that could connect them. We describe relative orientation later by dividing orientations into three classes. To do this we only use the first and last edges in each group. We reason that different distances and orientations will occur most frequently depending on whether groups come from the same or different objects.

We do not suggest that these are the most important clues to the likelihood that two groups of edges came from the same object. For example, the pattern of intensities in the image between the two groups tells us much, as do parallelism or symmetry. Our goal here is to understand as well as possible just two pieces of the grouping puzzle before attempting to combine them with other information.

### 3.2.1 The Probabilities that GROPER Computes

GROPER calculates the likelihood that two groups come from the same object by estimating four probabilities that influence this likelihood: $P(d|O_1 = O_2)$, $P(d|O_1 \neq O_2)$, $P(t|d, O_1 = O_2)$, and $P(t|d, O_1 \neq O_2)$. $d$ stands for the distance separating the groups, $t$ stands for their relative orientation. $O_1 = O_2$ indicates that the objects that produced groups one and two are the same. $O_1 \neq O_2$ indicates that the two groups came from different objects. Bayes' rule tells us that:

$$P(O_1 = O_2|d, t) =$$

$$\frac{P(d|O_1 = O_2) * P(t|d, O_1 = O_2) * P(O_1 = O_2)}{P(d|O_1 = O_2) * P(t|d, O_1 = O_2) * P(O_1 = O_2) + P(d|O_1 \neq O_2) * P(t|d, O_1 \neq O_2) * P(O_1 \neq O_2)}$$

In practice we can ignore the terms $P(O_1 = O_2)$ and $P(O_1 \neq O_2)$, because we only use $P(O_1 = O_2|d, t)$ for comparing different combinations of groups to decide which pair to try first. To see this, note that:

$$P(O_1 = O_2|d, t) \leq P(O_1' = O_2'|d', t')$$

iff

$$\frac{P(d|O_1 = O_2) * P(t|d, O_1 = O_2)}{P(d|O_1 = O_2) * P(t|d, O_1 = O_2) + P(d|O_1 \neq O_2) * P(t|d, O_1 \neq O_2)}$$

$$<= \frac{P(d'|O_1' = O_2') * P(t'|d', O_1' = O_2')}{P(d'|O_1' = O_2') * P(t'|d', O_1' = O_2') + P(d'|O_1' \neq O_2') * P(t'|d', O_1' \neq O_2')}$$

because $P(O_1 = O_2) = P(O_1' = O_2')$ and $P(O_1 \neq O_2) = P(O_1' \neq O_2')$. We call this fraction $L(O_1 = O_2)$, and it measures the relative likelihood that groups one and two come from the same object. To find it, we only need to calculate $P(d|O_1 = O_2)$, $P(t|d, O_1 = O_2)$, $P(d|O_1 \neq O_2)$, and $P(t|d, O_1 \neq O_2)$.

The next two subsections will discuss the problem of determining these four probabilities. We have examined a simple domain of random two-dimensional objects to gain intuitions about what factors are important in determining these probabilities. These intuitions allow us to generate some reasonable hypotheses, whose true test is their empirical performance on real images. Even an analysis of a simple domain reveals that $L(O_1 = O_2)$ depends on characteristics

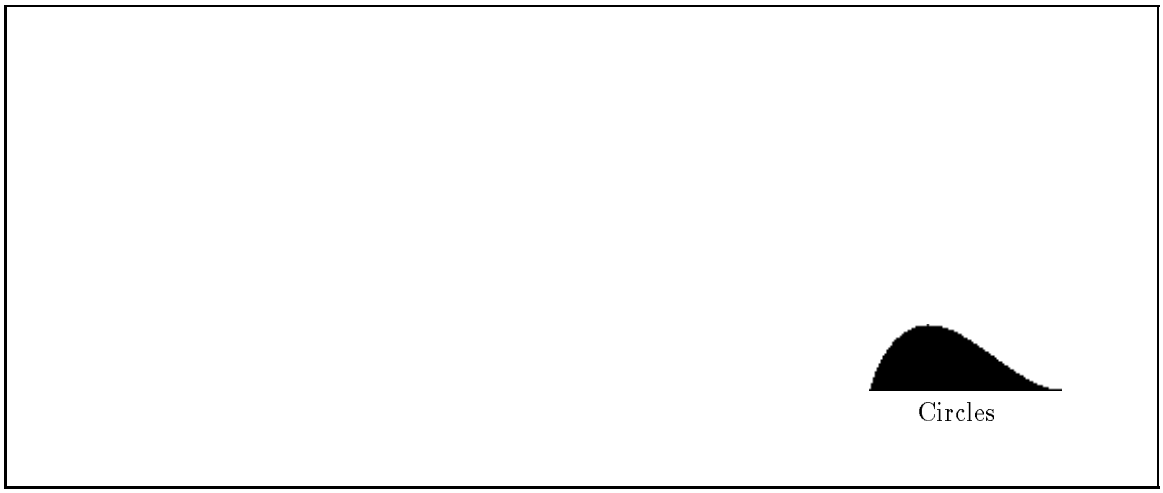


Figure 6: The four graphs on the left show the lengths of occlusions that occur from randomly intersecting random objects. On the right is the distribution resulting from randomly intersecting circles.

of the scenes we view such as the number of objects that appear in scenes, and their sizes. If these factors were crucial, we would need to tune parameters of the system for different types of scenes. In Section 5 we use two methods to show that this is not the case. First, we successfully test the system on images that have a wide variety of characteristics. Second, we test the system with a variety of parameters, and show that the system's performance is not sensitive to these variations.

### 3.2.2 Distance

To determine $P(d|O_1 = O_2)$ we must understand what causes some distance to separate two groups of edges that come from the perimeter of the same object. Examining the lengths of occlusions of objects tells us about one common cause of this separation.

Different collections of objects produce occlusions of different lengths. We have tried running GROPER using different assumptions about this distribution, but to get an idea of which distributions to test, we have done two things. First, we have generated occlusions by randomly positioning random polygons. Second, we have found analytically the lengths of occlusions produced by randomly positioned circles.

As explained in Appendix A, we generated random polygons with different numbers of convex parts. We then randomly oriented these objects within a rectangle, producing occlusions, and measured the distance between the beginning and the end of each occlusion. Figure 6 shows that short occlusions occur much more often than long occlusions, and that the number of points of concavity in an object does not seem to influence the lengths of occlusions much.

To find a different distribution worth considering, we found the lengths of occlusions that occur when we randomly locate in the plane circles of random radii. Integral geometry tells

us that of all convex shapes, circles produce occlusions that maximize the expected amount of object perimeter covered up (see Santalo[30], for example). This suggests that circles will produce an extreme case: the longest occlusions. Appendix B shows that for circles:

$$P(lengthofocclusion = D) =$$

$$-8\left(\frac{1}{2}\sqrt{1-\frac{D^2}{4}} - \frac{D^2}{8}cosh^{-1}\frac{2}{D}\right)\left(\frac{1}{4}\frac{1}{\sqrt{1-\frac{D^2}{4}}}\left(-\frac{D}{2}\right) - \frac{D}{4}cosh^{-1}\frac{2}{D} + \frac{1}{4}\frac{1}{\sqrt{\frac{4}{D^2}-1}}\right)$$

Figure 6 graphs this distribution.

We can see that circles produce a distribution different from random polygons, but with a similar shape. This provides us with a reasonable range of possibilities. For most experiments, we use $P(d|O_1 = O_2) = (MaxD - d)^2$, with $MaxD$, the maximum object diameter, equal to 300 pixels. This function has a shape between that of occlusions caused by circles and occlusions caused by random objects. We also tested GROPER using the distribution of occlusions caused by circles.

GROPER estimates $P(d|O_1 \neq O_2)$ by combining two possibilities. The groups may be randomly located within the image, assuming that different objects have independent random locations. Or, the location of the end points of the two groups may stem from a single cause.

If the two groups have independent locations, we can think of this problem as one of randomly placing two line segments, each with end points **a** and **b**, in the plane. We then take the minimum of the distance between the two points marked **a**, and the two points marked **b**. For short line segments, this distribution increases linearly with the distance. So, GROPER estimates this distribution as:

$$\frac{2\pi d}{\pi MaxDiameter^2} = \frac{2d}{MaxDiameter^2}$$

for $d \leq MaxDiameter$.

Alternately, two groups of edges from different objects may not have independent locations. Consider a simple case. The end point of one group may result from the other group occluding it. Suppose the two objects have the same reflectance. Then the edges produced by the two objects will terminate at the same place, with a point of concavity separating the edges into two simple groups. Two factors will influence the impact of such an occlusion: the probability of such an occlusion, and the distances between the groups when such occlusions occur.

We linearly combine our previous analysis with an analysis of two groups formed by a single occlusion, using the probability of such an occlusion occurring. In the case of an occlusion, one of the groups ends where the other begins. So the distance between them equals zero, unless something else occludes them both. If that happens, the distribution of the distance between the two groups will depend on the length of the second occlusion, a distribution we have already analyzed. So we use the following estimate:

$$P(d|O_1 \neq O_2) = k * P(d|O_1 = O_2) + (1-k) * \frac{2d}{MaxDiameter^2}$$

where $k$ is the probability that two groups of edges from different objects intersect. We used a value of $k = .2$ as a default, also trying values of $k = 0$ and $k = .5$.

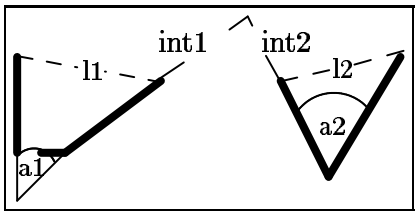


Figure 7: Two groups with a $type_2$ orientation, and the variables that describe them.

### 3.3 Orientation

To analyze the effect of orientation on the likelihood of groups coming from a single object, we divide orientations into three categories. We say that two groups have a $type_1$ relationship if they could come from a single convex section of an object. They have a $type_2$ relationship if they do not have a $type_1$ relationship, but could come from adjacent convex sections of an object. They have a $type_3$ relationship otherwise. This description proves useful because it makes it easy to reason about objects in terms of their convex parts. Many natural objects have a few convex parts, and much of our perception of objects seems to occur in terms of their convex parts (see Hoffman and Richards[15]).

Another way to think about *types* is to consider what we will call the **projection** of a group. Intuitively, projection refers to the area pointed to by a group of edges. More precisely, consider the following three half-planes. First, the line defined by the beginning and end points of a group forms a half-plane that does not include the group. Second and third, the first and last lines in the group define half-planes that include the edges of the group. We call the intersection of these three half-planes the group's projection. Two groups have a $type_1$ relationship if each group falls in the projection of the other. They have a $type_2$ relationship if their projections intersect.

To estimate the likelihood of different *types* occurring we look only at four variables that describe the beginning and the end of each group. $l_1$ stands for the distance between the beginning and end of group one, and $a_1$ for the angle between the first and last lines in group one. We define $l_2$ and $a_2$ similarly for group two. Figure 7 shows an example of these four variables, and two others, $int_1$ and $int_2$, that we will use later.

We now obtain estimates for the two likelihoods:

$$L(t|d, O_1 = O_2) = P(type|l_1, l_2, a_1, a_2, d, O_1 = O_2),$$

$$L(t|d, O_1 \neq O_2) = P(type|l_1, l_2, a_1, a_2, d, O_1 \neq O_2)$$

and use these likelihoods in place of $P(t|d, O_1 = O_2)$ and $P(t|d, O_1 \neq O_2)$.

These two likelihoods do not exhaust all sources of information about the influence of relative orientation on the probability that two groups came from the same object. For example, the fact that two groups have a $type_1$ orientation does not fully describe their relative orientation. And we would need $P(l_1, l_2, a_1, a_2|d, O_1 = O_2)$ and $P(l_1, l_2, a_1, a_2|d, O_1 \neq O_2)$ to compute

$P(t|d, O_1 = O_2)$ and $P(t|d, O_1 \neq O_2)$, information that could be used in grouping. We have considered the probabilities of different *type*s occurring because these seem the most easily understood, and the most useful.

We determine these probabilities for each of three possible types. We start with the probability of types occurring when $O_1 = O_2$. For each type, we divide the probability into three parts. What if the two groups actually come from the same convex section of an object (abbreviated "*same*")? What if they come from adjacent sections (*adj*)? What if they do not (*notadj*)? This creates nine subproblems.

Many of these nine problems have simple answers. For example, $P(type_3|adj, d, l_1, l_2, a_1, a_2, O_1 = O_2)$ equals 0, since if two groups come from adjacent convex sections of an object, they can not look as if they could not come from adjacent convex sections. Two other probabilities also equal zero, while $P(type_1|same, d, l_1, l_2, a_1, a_2, O_1 = O_2)$ equals one.

$$P(type_1|d, l_1, l_2, a_1, a_2, O_1 = O_2)$$

$$= P(type_1, same|d, l_1, l_2, a_1, a_2, O_1 = O_2) + P(type_1, adj|d, l_1, l_2, a_1, a_2, O_1 = O_2)$$
$$+P(type_1, notadj|d, l_1, l_2, a_1, a_2, O_1 = O_2)$$
$$= P(type_1|same, d, l_1, l_2, a_1, a_2, O_1 = O_2) * P(same|d, l_1, l_2, a_1, a_2, O_1 = O_2)etc....$$

However, probabilities like $P(same|d, l_1, l_2, a_1, a_2, O_1 = O_2)$ will depend on the nature of the objects we expect. As before, we handle this by trying different sets of values for these probabilities to show that performance will not depend too delicately on the values we choose. As a default, we set $P(same|d, l_1, l_2, a_1, a_2, O_1 = O_2) = P(adj|d, l_1, l_2, a_1, a_2, O_1 = O_2) = P(notadj|d, l_1, l_2, a_1, a_2, O_1 = O_2) = \frac{1}{3}$, but we also tried setting each probability to .5, while setting the remaining two to .25.

We still must find the distributions of the probabilities that do not have obvious values, such as $P(type_1|adj, d, l_1, l_2, a_1, a_2, O_1 = O_2)$. To do this with a simple analysis, we assume that the convex groups from an object have independent, uniform random orientations. Since we also assume that groups from different objects have independent, uniform random orientations, this tells us that: $P(type_i|notadj, d, l_1, l_2, a_1, a_2, O_1 = O_2) = P(type_i|notadj, d, l_1, l_2, a_1, a_2, O_1 \neq O_2)$, for $i = 1$, 2 or 3. But groups from adjacent convex sections of an object can not appear to come from non-adjacent convex sections, telling us:

$$P(type_i|adj, d, l_1, l_2, a_1, a_2, O_1 = O_2)$$

$$= \frac{P(type_i|adj, d, l_1, l_2, a_1, a_2, O_1 \neq O_2)}{P(type_1|adj, d, l_1, l_2, a_1, a_2, O_1 \neq O_2) + P(type_2|adj, d, l_1, l_2, a_1, a_2, O_1 \neq O_2)}$$

for $i = 1$ or 2.

We now consider the orientations produced by randomly located groups of edges when $O_1 \neq O_2$. To estimate the probability that group one falls in group two's projection, GROPER subtracts the angle group one presents to group two, $\alpha$, from the angle of projection, $\theta$, and divides by $2\pi$, or uses 0 if $\alpha > \theta$. GROPER determines these angles in two ways. If group two

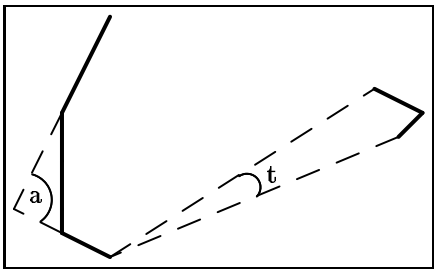


Figure 8: Two groups with a $type_1$ relationship. "t" represents the aspect the right group presents to the left. "a" is the angle of projection of the first group. The probability that the second group falls in the first's projection is approximated by $\frac{a-t}{2\pi}$.

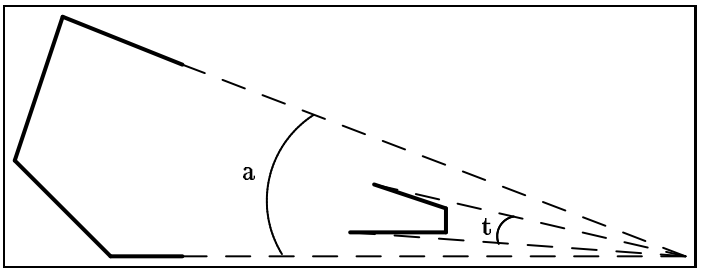


Figure 9: Finite projection. After rotating the right-side group to fall in the left's projection, "t" is the aspect the right group presents, and "a" is the left group's angle of projection. The probability that the right group falls in the left's projection is again approximated by $\frac{a-t}{2\pi}$.

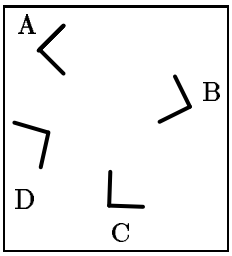


Figure 10: A and B have a $type_1$ relationship, and seem more likely to come from the same object than A and C, which are the same, but have a $type_2$ relationship. These go better together than A and D, which have a $type_3$ orientation.

has an infinite projection, $\theta$ is the angle formed by its first and last edges, that is $a_1$. $\alpha$ is the angle formed by lines from the first point in group two to the first and last points in group one. If group two has a finite projection, then GROPER finds the point, p, where the lines defined by group two's first and last edges intersect. $\theta$ equals the angle formed by lines from p to the first and last points of group two. To find $\alpha$, we rotate group one so that it falls inside group two's projections, without altering its distance from group two. Then, $\alpha$ is the angle formed by lines from p to the beginning and ending of group one. If group one does not fit inside group two's projection, GROPER assigns a zero probability to its randomly falling there. Figures 8 and 9 illustrate these probabilities.

Next, to find the probability that group two falls in group one's projection, we assume this probability is independent of the previous one, and calculate it in the same way. This allows GROPER to estimate $P(type_1|d, l_1, l_2, a_1, a_2, O_1 \neq O_2)$ by multiplying these two probabilities together.

To find $P(type_1|d, l_1, l_2, a_1, a_2, O_1 \neq O_2)$ we look at the simple case where the two groups have only a negligible size. In this case, we find in Appendix C that:

$$P(type_3|O_1 \neq O_2, d, l_1, l_2, a_1, a_2) = \frac{\frac{3\pi}{2} - a_1 - a_2 + \frac{a_1 a_2}{2\pi}}{2\pi}$$

If either group has a finite projection, GROPER uses an angle of 0 for that group.

GROPER finds the probability of $type_2$ occuring using the probabilities of $type_1$ and $type_3$ occurring.

This approach yields some intuitively satisfying results. For example, the above probabilities imply that, all other factors being equal, groups with a lower *type* of orientation have a greater chance of coming from the same object. Figure 10 shows an example.

GROPER uses one more piece of information to evaluate the likelihood that two groups came from the same object. When they have a $type_2$ orientation, GROPER uses the distance from the groups to the intersection of their projections. We call the distance from group one to group two's projection $int_1$, and define $int_2$ similarly for group two. See Figure 7 for an example.

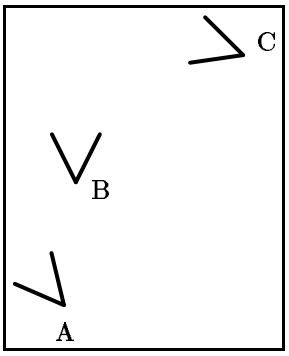


Figure 11: Groups B and C seem to go together better than groups A and B. The same distance separates each pair, but the distance to the intersection of their projections differs greatly.

The greater these distances, the less the probability GROPER assigns to the hypothesis that the two groups came from one object.

GROPER uses this information because of psychophysical intuitions. For example, groups A and B in Figure 11 differ from B and C in that the projections of B and C intersect closer to the groups. Because of this difference the groups on the left seem to go together better. Section 5 shows results that indicate that this constraint improves GROPER's performance considerably. We use these probabilities by multiplying $L(type_2|d, O_1 = O_2)$ by $P(int_1|adj, d, O_1 = O_2) * P(int_2|adj, d, O_1 = O_2)$, and multiplying $L(type_2|d, O_1 \neq O_2)$ by $P(int_1|d, O_1 \neq O_2) * P(int_2|d, O_1 \neq O_2)$.

The likelihood of groups coming from the same object should vary inversely with $int_1$ and $int_2$, but we do not know at what rate. So, as a default we have used $P(int_i|d, O_1 = O_2) = MaxP - d$ and $P(int_i|d, O_1 \neq O_2) = 1$, normalized to be a probability distribution. $MaxP$ stands for the maximum diameter of a convex part, which we set to 150 pixels. We have also tested GROPER using $P(int_i|O_1 = O_2) = (MaxP - d)^2$, and $MaxP = 75$ pixels. Two things are clear, whichever distribution we try, however. When only a small distance separates the end points of two groups, then $int_1$ and $int_2$ will have low values for most orientations of the groups, and so provide little information. Hence in these cases, GROPER does not attempt to use this information. And secondly, when $int_i$ exceeds $MaxP$, the two groups could not really come from adjacent convex parts, and should be treated as having a $type_3$ relationship.

### 3.4 Producing Larger Groups

When two groups do not provide enough information to allow GROPER to recognize an object, GROPER combines other groups with the original two, and estimates the probability that each triple of groups came from a single object. It does this by taking the maximum of: $L(O_1 = O_2) * L(O_1 = O_3)$, $L(O_1 = O_2) * L(O_2 = O_3)$, and $L(O_1 = O_3) * L(O_2 = O_3)$. GROPER does place one restriction on this calculation. Previously, for the distance between

two groups, GROPER used the minimum distance from the start of one group to the end of the other. We do not want to hypothesize a connection between the start of group one and the ends of both group two and three. So we use the distance from the start of group one to the end of group two, and from the end of group one to the start of group three, and then try it the other way around, using the maximum probability that emerges.

# 4 Recognition

We now briefly discuss GROPER's recognition component. This consists of indexing and verification modules. GROPER indexes into a table that describes the relationships between all pairs of edges in each model. Verification uses matches between image and model features to solve for the location of the model, and then looks for additional matches. The entire system performs the following steps:

**1.** Summarize polygonal models of all known objects in a look-up table.

**2.** Make line approximations to all the intensity edges in an image.

**3.** Form simple convex groups.

**4.** For every convex group with four or more edges, perform steps seven and eight.

**5.** Form a pool of all pairs of simple groups. For each pair, estimate $L(O_1 = O_2)$.

**6.** Choose the group from the pool thought most likely to have come from a single object.

**7.** Use indexing to see which model edges might match this group of image edges.

**8a.** If no modeled edges match the image edges chosen, return to step six.

**8b.** If only a few sets of model edges match these image edges, perform a verification step on set of matches. Remove from future consideration the edges matched to a recognized object, and return to step three.

**8c.** If a group could match many objects, pair it with all the other groups and add each combination to the pool. Return to step six, choosing the next five groups to explore from these new ones.

We have discussed grouping. Edge detection is done using the Canny[5] or Marr-Hildreth[27] edge operators. Straight line approximations to edge contours are found using a standard technique. We connect the start and end of a connected string of edge pixels with a line, then break this line at the point of maximum deviation. We then repeat this process recursively, until each line approximates edge contours to within two pixels (see Pavlidis and Horowitz[29] for a description of this type of algorithm). In the remainder of this section we describe GROPER's indexing and verification.

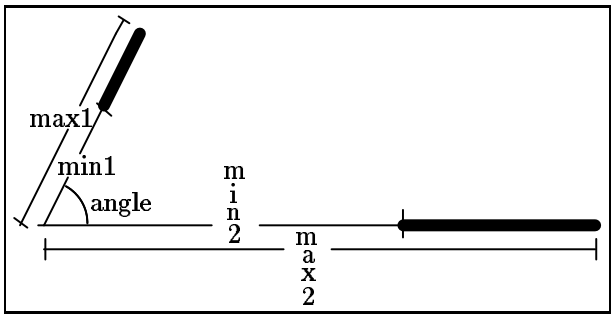


Figure 12: Two edges are in bold. Five parameters describe their relationship.

Indexing determines the sets of edges from modeled objects that might have produced a set of image edges. Verification finds the position of an object in the image using a match between image edges and model edges. This position then allows GROPER to locate additional edges the object might have produced.

Since short lines found by the straight-line approximator tend to be quite uncertain in their angle, GROPER discards all edges less than ten pixels long before attempting recognition. Grouping does use these edges[2].

Because indexing only guides the recognition process, we do not mind if indexing produces some false positive matches. Later, verification will perform a rigorous test and reject them. However, we do not want indexing to miss a correct match, for then we may abandon a group that could lead to recognition.

GROPER uses an indexing scheme closely related to one proposed independently by Wallace[39]. We can describe the relationship between two edges with five variables. We can then fill a five-dimensional array with the values that describe all pairs of edges in the object models in our library. To find objects that might have produced two edges, we calculate these five variables and look in the table. For more than two edges, we take the intersection of the result of a table look up for each pair.

The above account leaves out some details. First, we must worry about sensing error. GROPER makes an entry in the table for every possible set of variables that each pair of edges might produce, given some allowed error. Secondly, GROPER uses two ways of describing the relationship between two edges. If the edges are not parallel, GROPER finds the point where they would intersect. It then calculates the angle between the edges, and, for each edge, the minimum and maximum distance from the edge to this intersection point (see Figure 12). For parallel edges, GROPER uses the angle between the edges (that is 0), the distance between them in the direction of their normals, the minimum and maximum distance from the beginning of one edge to the beginning of the other in the direction of the tangent, and the length of the second edge. Finally, GROPER does not really use a five-dimensional array. It uses a three-

[2] A previous implementation of GROPER discarded short edges before performing grouping. Primarily because of this, some of the results reported in Jacobs[18] differ significantly from those described in Section 5.

dimensional array of angle, and the distance of each edge to the intersection point. For a pair of edges, it makes an entry for every triple of angle, distance from edge 1, and distance from edge 2, that any points on the two edges might produce. It does this because occlusion might cause only a portion of an edge to appear in the image. Then to find the model edges that match a pair of image edges, GROPER combines the results of two lookups, one using the smallest distances from each edge to the intersection point, the other using the largest distances.

This parameterization makes it easy to check for the global consistency of a match, not just its pairwise consistency. If we perform a table lookup for every pair of edges in a group and just intersect the results, we might get a match in which, for every pair of image edges some transformation of the image will align them with the corresponding model edges, but no single transformation exists that will align all the image edges with the model edges. For more on this point and its significance, see Grimson and Lozano-Pérez[11]. The parameterization GROPER uses allows it to keep track, not only of whether two image edges could match two model edges, but also, of which parts of the model edges they could match. This allows GROPER's indexing system to enforce global consistency on the matches it finds. Jacobs[18] describes the method used to do this.

This indexing system allows GROPER to make use of any collection of edges. It also uses a good deal of space. The amount of space required increases linearly with the number of object models, and is proportional to the square of the number of edges in each model. However, the number of table entries GROPER makes for each pair of model edges is quite large. This is partly because GROPER takes a simple approach to finding the table entries that does not miss any correct entries but makes many unnecessary ones. As a result, a table containing sixteen objects that had from six to thirty-one edges each, required over 400,000 table entries. A typical object, with ten edges, required over 14,000 table entries. This space requirement makes it impractical to use a library with more than about a dozen objects.

Verification uses the matches produced by indexing to find a rotation and translation that maximally aligns the model edges of a match with the image edges. GROPER does this using a technique developed by Grimson and Lozano-Pérez[11]. It then looks for image edges close to the proposed location of the object model. If GROPER can in this way account for a given percentage of an object's perimeter it decides it has found the object.

Deciding when to perform verification offers a trade-off between accuracy and speed. If we perform verification when a group of edges matches many model edges, it will take a long time, but if we do not perform verification we risk the possibility that further grouping will fail to turn up any additional edges that will lead to a solution. We perform verification when a group of edges matches five or fewer sets of model edges, but have also found that a threshold of fifteen matches produces similar results.

# 5 Results

We test GROPER to measure the amount its grouping system can reduce the computation needed to recognize objects, and can improve the accuracy of a recognition system. We measure this for images of two-dimensional, polygonal objects by comparing GROPER's perfor-

mance to that of an identical recognition system that does not use grouping. We also run GROPER's grouping system alone on images of curved, three-dimensional objects. We find that GROPER's grouping system produces dramatic improvements in the performance of a recognition system. And, we find that GROPER's grouping system works almost as well on images of real three-dimensional objects as it does on images of simple two-dimensional objects. This leads us to expect that GROPER could produce similar improvements in the recognition of three-dimensional objects.

Finally, we vary the parameters of GROPER's grouping system, as described earlier, to show that the system does not depend on a precise tuning of these variables.

### 5.1 Recognition

SEARCHER uses all GROPER's code except its grouping module. Grouping orders GROPER's search through the space of collections of image edges. SEARCHER performs a backtracking search through this space instead. Like GROPER, when SEARCHER finds that some edges do not match any known object, it does not consider any other collections of edges that include these. And, like GROPER, when SEARCHER recognizes an object it removes from consideration any edges that this object can explain. Since SEARCHER has no way to tell the object side from the background side of an edge, it must consider both possibilities in its search, just as GROPER imposes figure/ground judgments on its groups. By replacing GROPER's guided search with an undirected search, SEARCHER shows us how much GROPER's grouping component adds to its performance.

We tested GROPER and SEARCHER on three sets of images of increasing difficulty. First, we used five images of six objects and a library containing models of those six objects. Then we used a library of sixteen objects. In the second test we used five images, each containing eight of these objects. Finally, we used three images containing all sixteen objects in the library.

Figure 13 shows these objects. They were cut out of black paper and placed on a light background. Figures 14 and 15 show examples of these images and the objects found in them.

Initially, we used loose error bounds for indexing and verification. We allowed for an error of seven pixels in the sensed location of an edge, and an error of $sin(\frac{\pi}{10})$ in the sensed angle between two edges. And, the systems accepted any match that accounted for 25% of an object's perimeter. We chose these bounds because informal experiments showed that tighter bounds occasionally caused GROPER to miss an object.

Figure 16 shows the results of these experiments. Occasionally, a system would decide that a collection of edges could match two or three different objects, and that no additional edges would narrow down the choice. We counted such a decision as correct if one of those objects was correct.

Although GROPER performed well on the first two sets of images, SEARCHER performed terribly, often finding incorrect matches that still fit the loose error bounds we had chosen. To obtain a better comparison we tuned these bounds to optimize SEARCHER's performance. We ran SEARCHER on the first set of images using 48 sets of error bounds. Error in distance varied between 7, 5, 4 and 3 pixels. Error in angle varied between $sin(\frac{\pi}{10})$, $sin(\frac{\pi}{15})$, and $sin(\frac{\pi}{20})$. And we tried object perimeter thresholds of 25%, 35%, 50%, and 60%. SEARCHER obtained

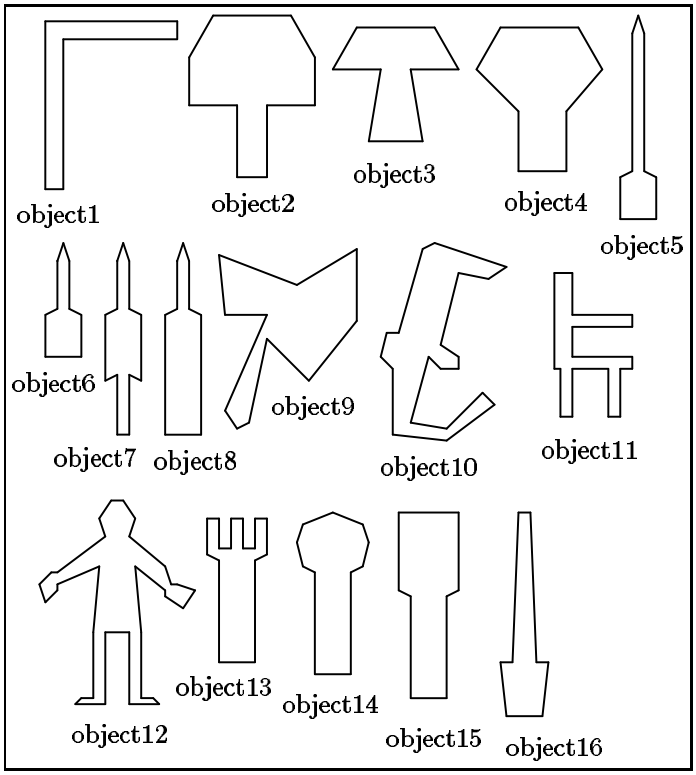


Figure 13: The perimeters of the objects used in tests. The first set of tests used objects 3, 4, 8, 9, 10, and 14. The second and third sets of tests used all sixteen objects.

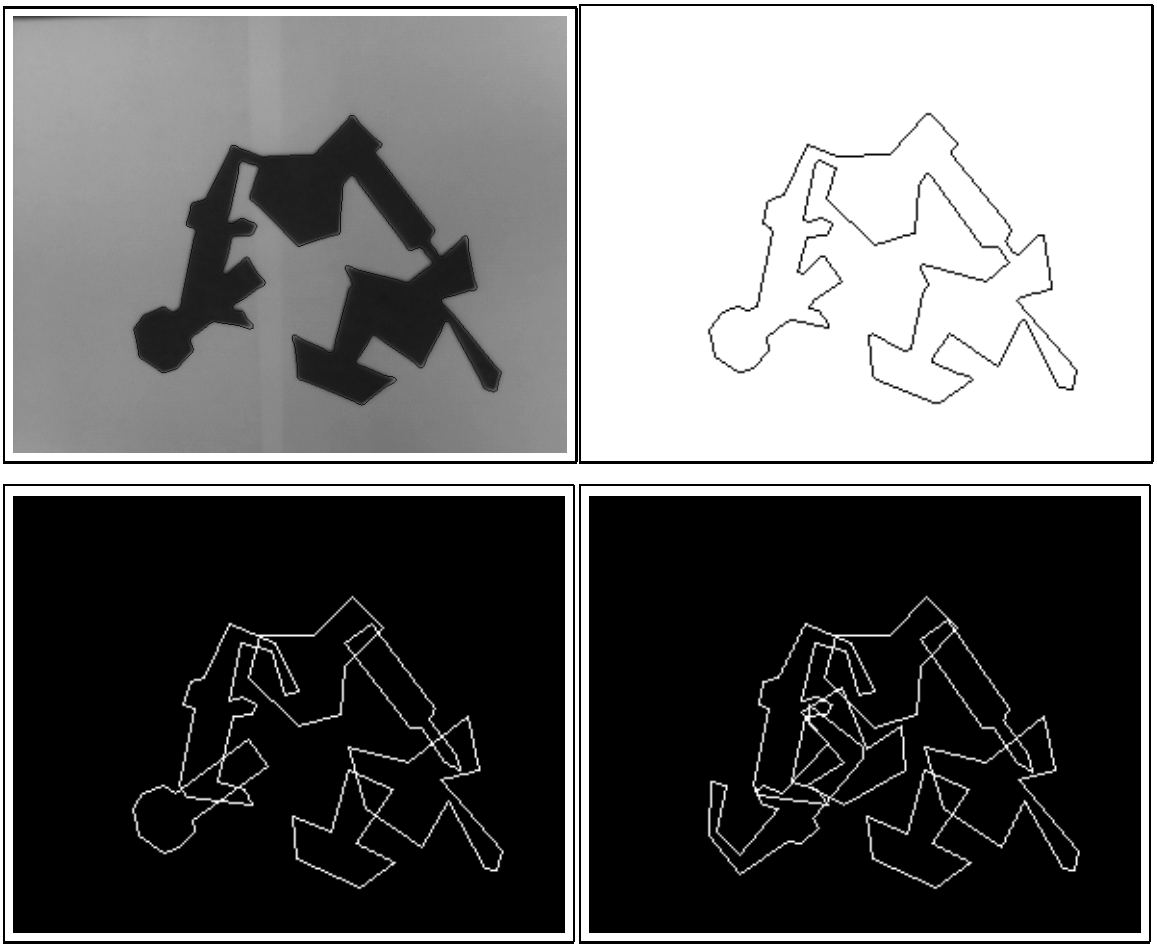

Figure 14: The upper left is a scene from the first set of tests. The upper right shows straight line approximations to the edges found. The lower left shows the objects GROPER found, acccounting for 25% of the object's perimeter, allowing for error of 7 pixels in distance, and $sin\frac{\pi}{10}$ in angle. On the lower right are the objects that SEARCHER found.

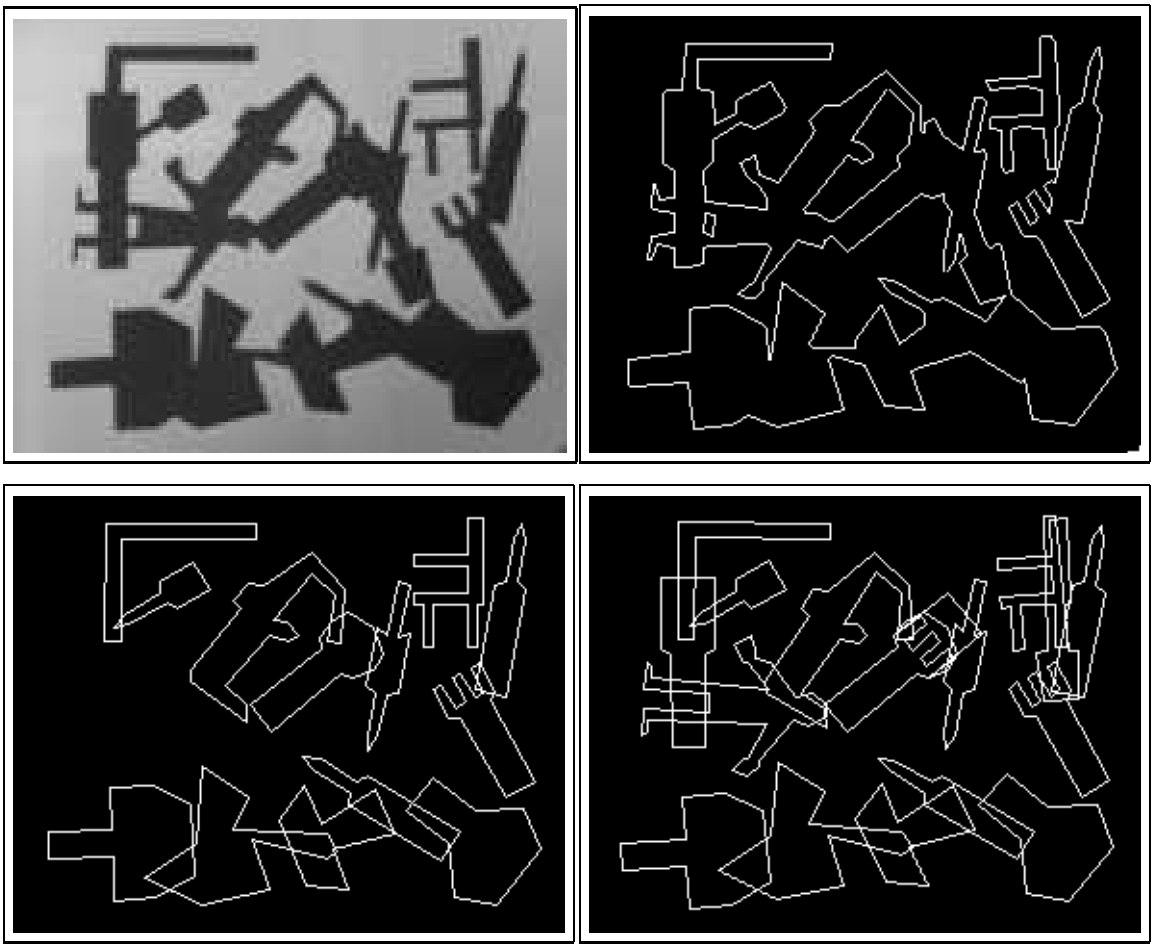

Figure 15: An image from the third set of tests. Again, the picture on the lower left shows the objects GROPER found, and the picture in the lower right shows SEARCHER's finds. Three hypotheses are overlaid where SEARCHER found three models that explained the same edges. 50% of each object was accounted for, with maximum errors of 3 pixels and $sin\frac{\pi}{15}$ radians.

| Perimeter | Angle Error | Dist Error | System | Test | False Positives | False Negatives | Nodes Explored |
|---|---|---|---|---|---|---|---|
| 25% | sin Π/10 | 7 pixels | GROPER | 1 | 0 | .6 | 8.6 |
| | | | | 2 | 1 | .6 | 39.4 |
| | | | | 3 | 6.7 | 6.3 | 370.3 |
| | | | SEARCHER | 1 | 7.2 | 3.6 | 2,621.8 |
| | | | | 2 | 9 | 5 | 10,202.4 |
| | | | | 3 | 16.7 | 8.6 | 39,653.7 |
| 35% | sin Π/15 | 3 pixels | GROPER | 1 | 0 | .8 | 14.6 |
| | | | | 2 | 0 | 1.2 | 30 |
| | | | | 3 | .7 | 6 | 1,098.7 |
| | | | SEARCHER | 1 | 0 | 0 | 17,510.2 |
| | | | | 2 | 2.2 | .6 | 41,579.2 |
| | | | | 3 | 5 | 2.3 | 226,031.7 |
| 50% | sin Π/15 | 3 pixels | GROPER | 1 | 0 | .8 | 14.6 |
| | | | | 2 | 0 | 1.4 | 29.2 |
| | | | | 3 | .3 | 6.7 | 499 |
| | | | SEARCHER | 1 | 0 | .2 | 19,260.6 |
| | | | | 2 | .4 | .4 | 45,911.2 |
| | | | | 3 | .7 | 1 | 252,420.7 |

Figure 16: This chart shows the average number of mistakes made per image. *Nodes Explored* indicates the average number of indexing steps per image.

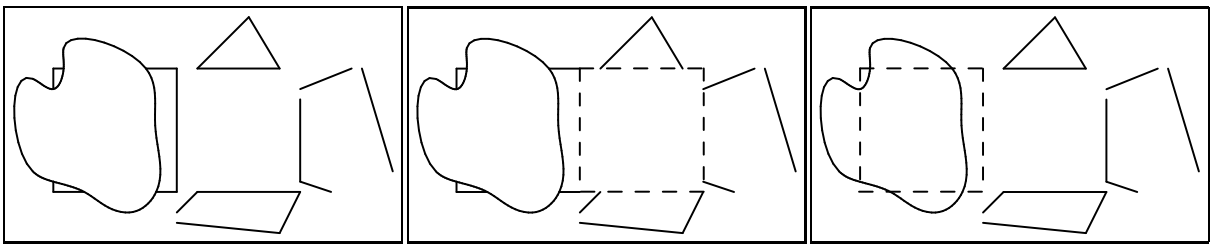

Figure 17: On the left, an hypothetical scene. In the middle, dashed lines show an incorrect hypothesis that accounts for most of a square's perimeter. On the right, an hypothesis that seems more likely to be correct.

perfect performance with a distance error of 3 pixels, an angle error of $sin(\frac{\pi}{15})$, and a threshold of 35%. Higher thresholds caused SEARCHER to miss some objects. We ran GROPER and SEARCHER using these bounds on all three sets of images. These results also appear in Figure 16. Again SEARCHER performed poorly overall. Finally we noticed that SEARCHER did better with a perimeter threshold of 50%, so we also ran experiments with that threshold.

We can see that GROPER requires far less computation than SEARCHER. For example, using the tightest error bounds, SEARCHER performs over 1,000 times as many indexing steps as GROPER. In Section 5.3 we show that all of GROPER's constraints play important parts in this reduction in computation.

We will now discuss the effects of grouping on the errors a recognition system makes. We will spend more time discussing error than speed, since it is a more complex issue. But the main contribution of GROPER is the speed-up that grouping provides.

These experiments tell us that when we do not have error bounds that separate almost all correct hypotheses from incorrect ones, GROPER has much better accuracy than SEARCHER. This is because grouping allows us to accept hypotheses not just on the basis of how much of an object's perimeter it accounts for, but also on the basis of how well the image edges in the hypothesis group together.

Figure 17 illustrates this point. Suppose we are looking for a square among the edges on the left. The dotted lines in the center picture show one hypothesis, which accounts for most of the edges of a square. But this hypothesis seems wrong. On the right, another hypothesis accounts for less of a square's perimeter, but seems correct. This is because the edges hypothesized on the right to come from a square group well together. The edges in the center group poorly together, and each edge in the hypothesis groups well with other edges. In fact, GROPER would only consider the hypothesis on the right, while SEARCHER might try either one first.

This problem emerges when we select error bounds that some incorrect matches can satisfy. Both GROPER and SEARCHER accept the first matches they find that satisfy their error bounds. Many other systems do this too, since it greatly reduces computational requirements (for example, Lowe[25], Huttenlocher and Ullman[16], Ayache and Faugeras[1], and Grimson and Lozano-Pérez[11]). But this means that SEARCHER will frequently consider incorrect matches that satisfy the error bounds before finding all the correct matches. On the other

hand, GROPER will first find the matches that satisfy the error bounds and that also contain edges that group well together.

We found GROPER more robust than SEARCHER as error bounds varied. With optimal error bounds, SEARCHER out-performs GROPER because GROPER's grouping system causes it to ignore some hypotheses. But it proved difficult to choose optimal error bounds, and they varied between test sets. Moreover, slightly sub-optimal error bounds produced significant errors in SEARCHER's performance. For example, SEARCHER performed perfectly on the images in test one, allowing a distance error of 3 pixels, an angle error of $sin(\frac{\pi}{15})$, and accepting matches that accounted for 35% of an object's perimeter. On the second set of five images, these error bounds produced three false negative errors, and eleven false positive errors. Changing error bounds had much less effect on GROPER's performance, and GROPER performed well with weak error bounds. This does not tell us that GROPER performs more accurately than SEARCHER, but that grouping may offer a way of improving the accuracy of a recognition system.

Furthermore, even though GROPER sometimes makes more errors than SEARCHER, it always finds fewer false positive matches. False positive matches are harder to recover from than false negatives, because we could always follow-up GROPER's work with a more exhaustive search of the edges for which it can not account.

Why does SEARCHER perform more accurately than GROPER, when they use tight error bounds? Ironically, most of GROPER's mistakes seem to occur when GROPER forms a group of edges that all come from the same object, but one of them is sensed with error exceeding the allowed bounds. As a result, the whole group can not match the object that produced it. SEARCHER can find the object using just the accurately sensed lines.

### 5.2 Grouping

We have tested GROPER's grouping component by itself on images of curved, three-dimensional objects. We examined the thirty pairs of simple convex groups that GROPER thought most likely to come from a single object. This tested the grouping constraints which form the heart of GROPER's grouping system. And the number of correct groups found gives us an idea of the number of objects we might recognize by considering only thirty groups. We also noted the total number of pairs of simple groups in each image, and the total number of correct pairs, allowing a comparison between GROPER's performance and a random search.

We used three sets of images of three-dimensional objects, and the images of two-dimensional objects discussed in the previous section. Comparing these results shows that grouping performs almost as well on three-dimensional images, leading us to expect comparable improvements in a three-dimensional recognizer.

Figure 18 displays the results of these tests. There are two differences between the way we handled two-dimensional and three-dimensional scenes. For two-dimensional scenes, we counted a group as correct if all the edges came from the same object, and if GROPER correctly resolved figure and background for each edge. For three-dimensional scenes, we counted a group as correct only if all its edges came from variations in an object's shape, not as a result of lighting. In the two-dimensional scenes, lighting variations did not produce edges. So, edges from an

| | Number correct pairs found | Total number Correct pairs | Total number Pairs |
|---|---|---|---|
| Two-Dimensional Test 1 | 7.2 | 19.6 | 564.8 |
| Two-Dimensional Test 2 | 8.4 | 23.6 | 1018.0 |
| Two-Dimensional Test 3 | 6.3 | 56.3 | 4547.0 |
| Three-Dimensional Test 1 | 7.5 | 45.5 | 1331.3 |
| Three-Dimensional Test 2 | 5 | 59.5 | 5613.0 |
| Three-Dimensional Test 3 | 6.7 | * | 9867.0 |

Figure 18: The performance of GROPER's grouping system. The six sets of images are described in the text. For each set, we examined the 30 pairs of simple convex groups that GROPER picked first. * indicates that these images contained so many edges produced by lighting effects and complex objects that we could not accurately count all the correct pairs of simple groups.

object's perimeter, surface markings, or orientation discontinuities counted as correct. But if a group contained edges produced by shadows or specularities, that group counted as incorrect. However, we only looked at figure/ground judgements when an edge really came from an object's perimeter. Also, for two-dimensional scenes we smoothed images using a Gaussian with a sigma of 1, during edge detection. For three-dimensional scenes, we found that a sigma of 3 worked better.

Figure 18 shows that for images of two-dimensional objects, between 6.3 and 8.4 of the first thirty pairs of convex groups selected by GROPER were right. This meant that GROPER recognized many or most of the objects in an image with fewer than 30 indexing steps. For the first set of tests on 3d objects, we formed four images of comparable complexity to the 2d tests. So these had six common objects, mostly without texture. Figures 19 through 22 compare GROPER's performance on one of these 3d scenes with a 2d scene. On average, GROPER found 7.5 correct groups in images of simple 3d objects. This means that some objects produced more than one correct group. This is similar to the results obtained on comparable images of two-dimensional scenes. Figures 19 and 21 show the large, single convex groups found in these images, which are also important in speeding the recognition of objects.

Next, we ran the grouping system on two pictures of our lab. Figure 23 shows one of these pictures. These images contained more pairs of convex groups, and a lower percentage of correct pairs, than any of the two-dimensional scenes tried. Yet GROPER performed only slightly less well on these images than it had on the most difficult images of two-dimensional scenes.

Finally, we tested three images containing non-rigid and translucent objects, obtaining similar results. Figures 3 and 4 show one such scene, and the groups found in it.

These tests show little difference between scenes of three- and two-dimensional objects. GROPER did find fewer correct groups when images contained more groups altogether, and when fewer of these groups were correct.

These results indicate that GROPER could serve as a front-end module to systems that recognize three-dimensional objects in complex scenes, significantly improving their performance. For example, Huttenlocher and Ullman's alignment method can use pairs of vertices in the image matched to model vertices to locate an object. GROPER could order pairs of vertices based on how well they group together. Or, we could perform the type of constrained search that Grimson and Lozano-Pérez use on subsets of the image data that GROPER has formed into groups.

### 5.3 Varying GROPER's Parameters

The theory of computation of grouping left some variables and probability distributions undetermined. We selected values that made intuitive sense, but we must still be concerned about the sensitivity of the system to these choices. Partly, we have done this by first fixing these parameters, and then testing the system on the 22 images described above. But another way to insure that the system is not sensitive to the choice of parameters is by testing it with alternate parameters.

For each parameter, we have selected a quite different value that also makes reasonable intuitive sense. We have found that using these alternate parameters makes relatively little

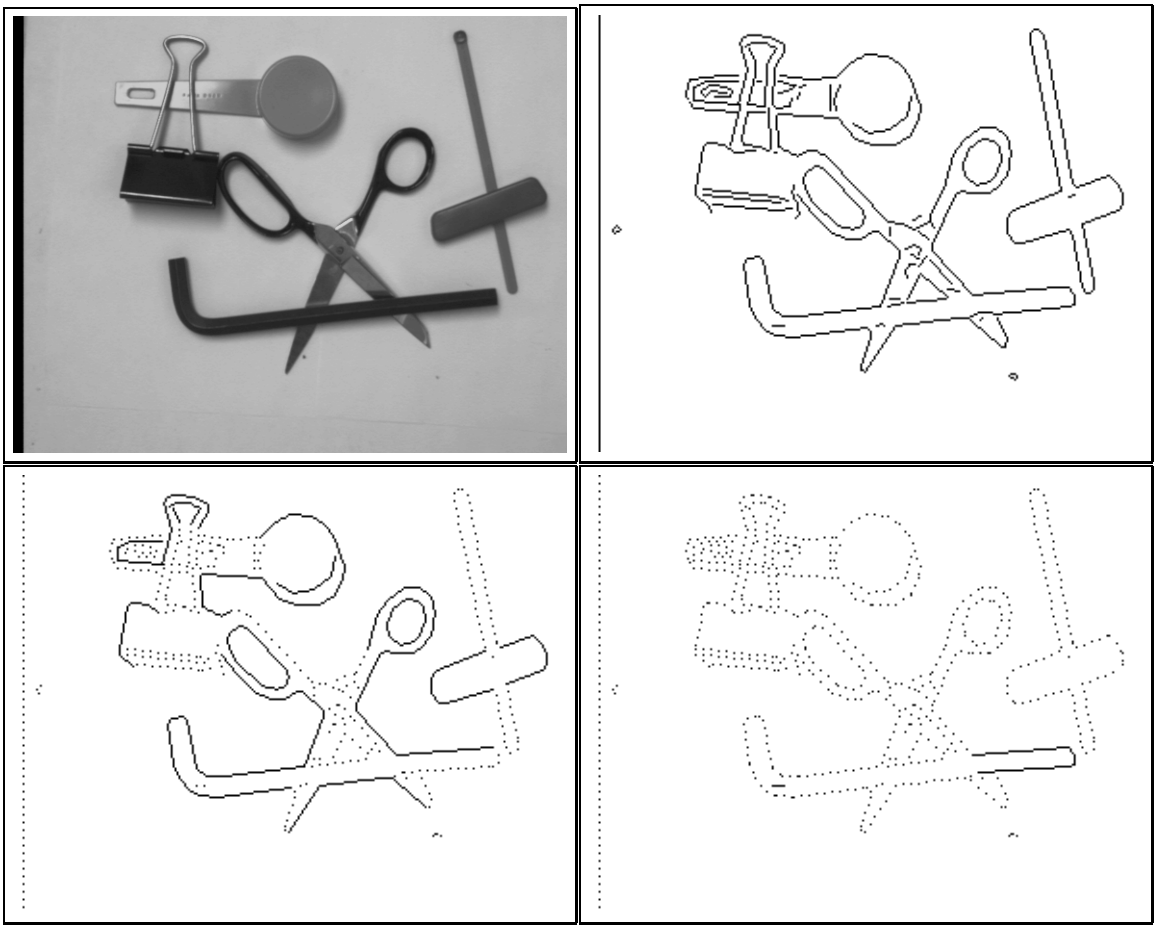

Figure 19: Above, an image and straight-line approximations to the edges found in it. The lower left shows the convex groups of four or more edges found in the image. The lower right shows the first pair of convex groups chosen by the grouping system. These edges all come from the wrench, but one comes from a specularity. This group would not count as correct. Figure 20 shows the next six groups chosen.

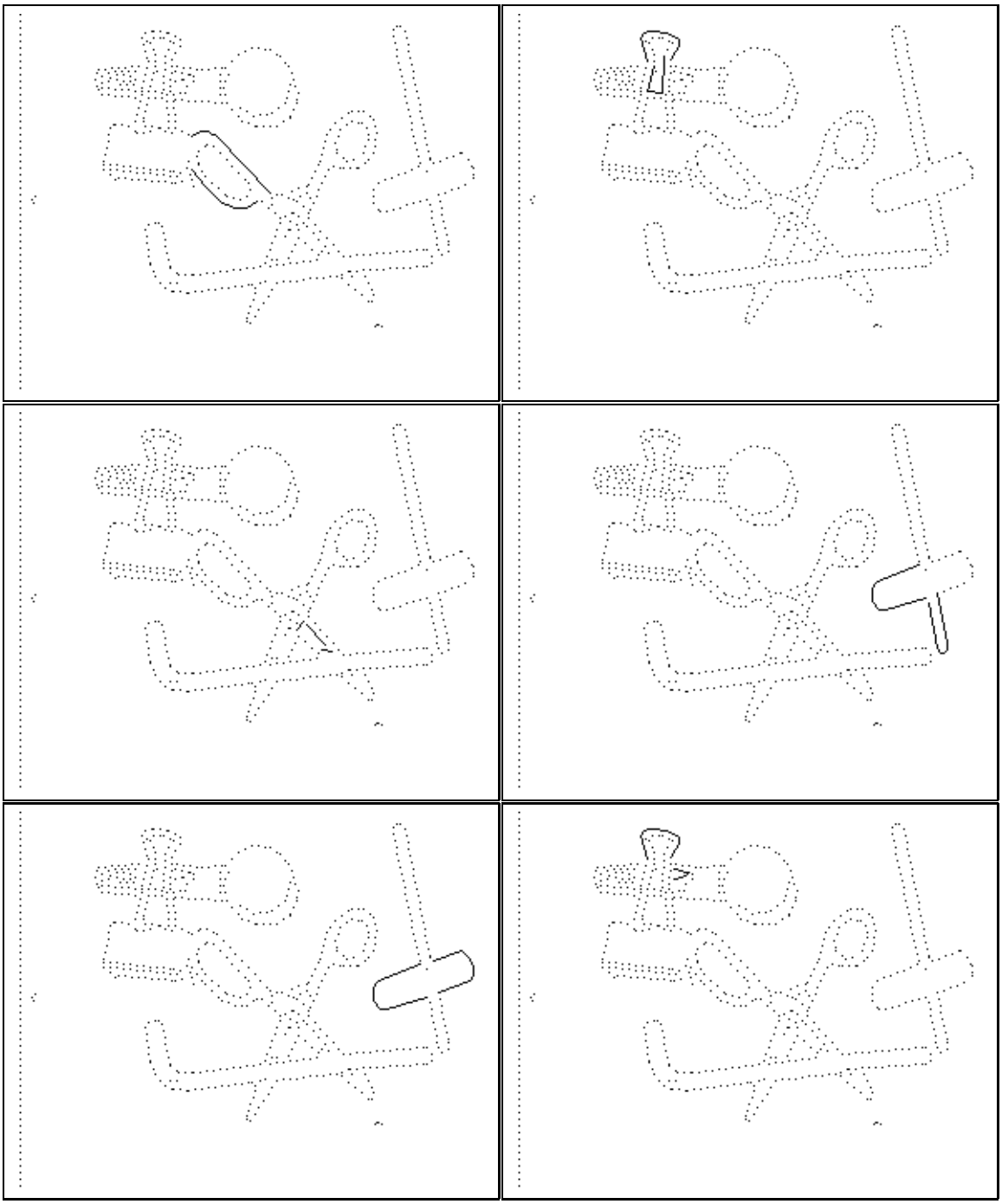

Figure 20: The second through seventh pairs of convex groups selected by the grouping system, from the picture in Figure 19. The second and sixth pairs are correct.

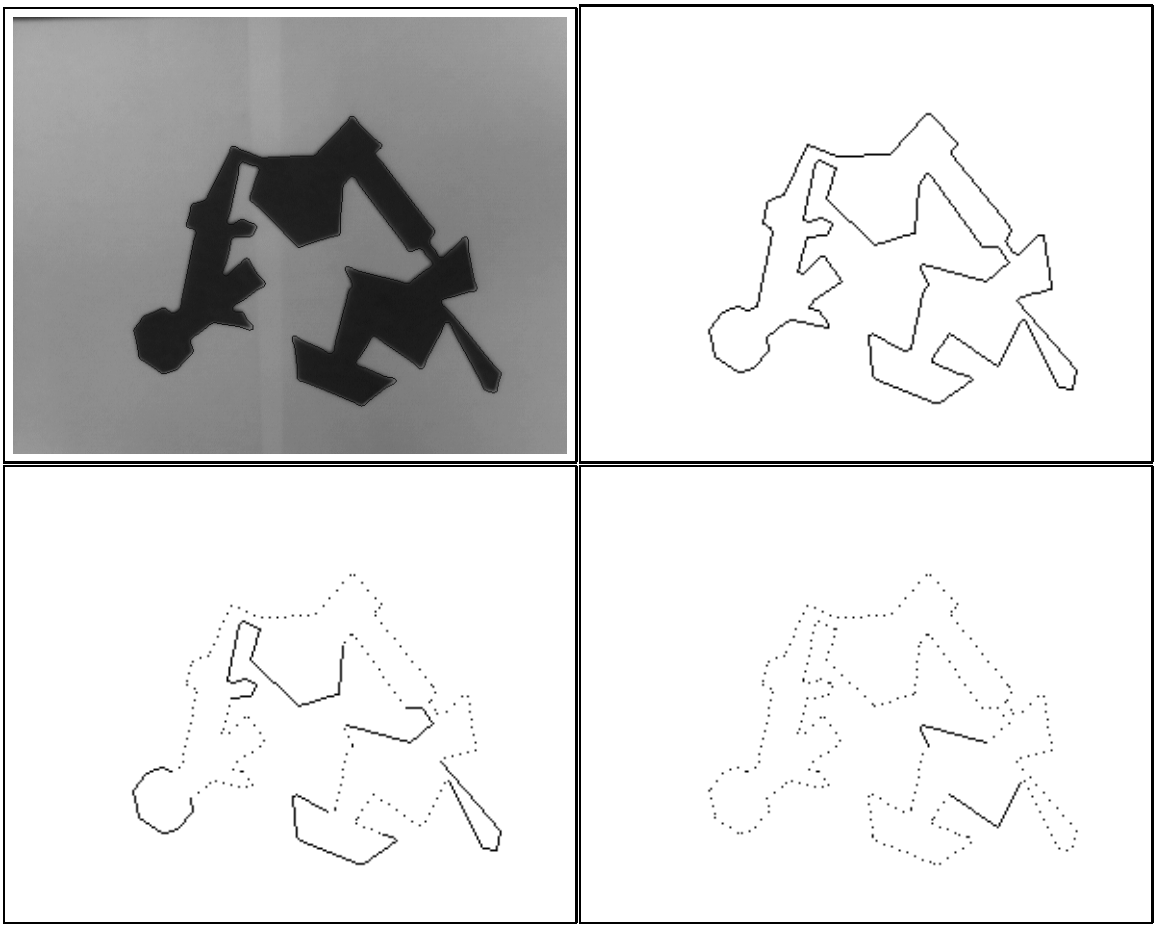

Figure 21: Above, a picture of two-dimensional objects and line approximations to the edges found. The lower left shows the convex groups of four or more edges. The lower right shows the first pair of convex groups chosen by the grouping system, which is correct. Figure 22 shows the next six pairs chosen.

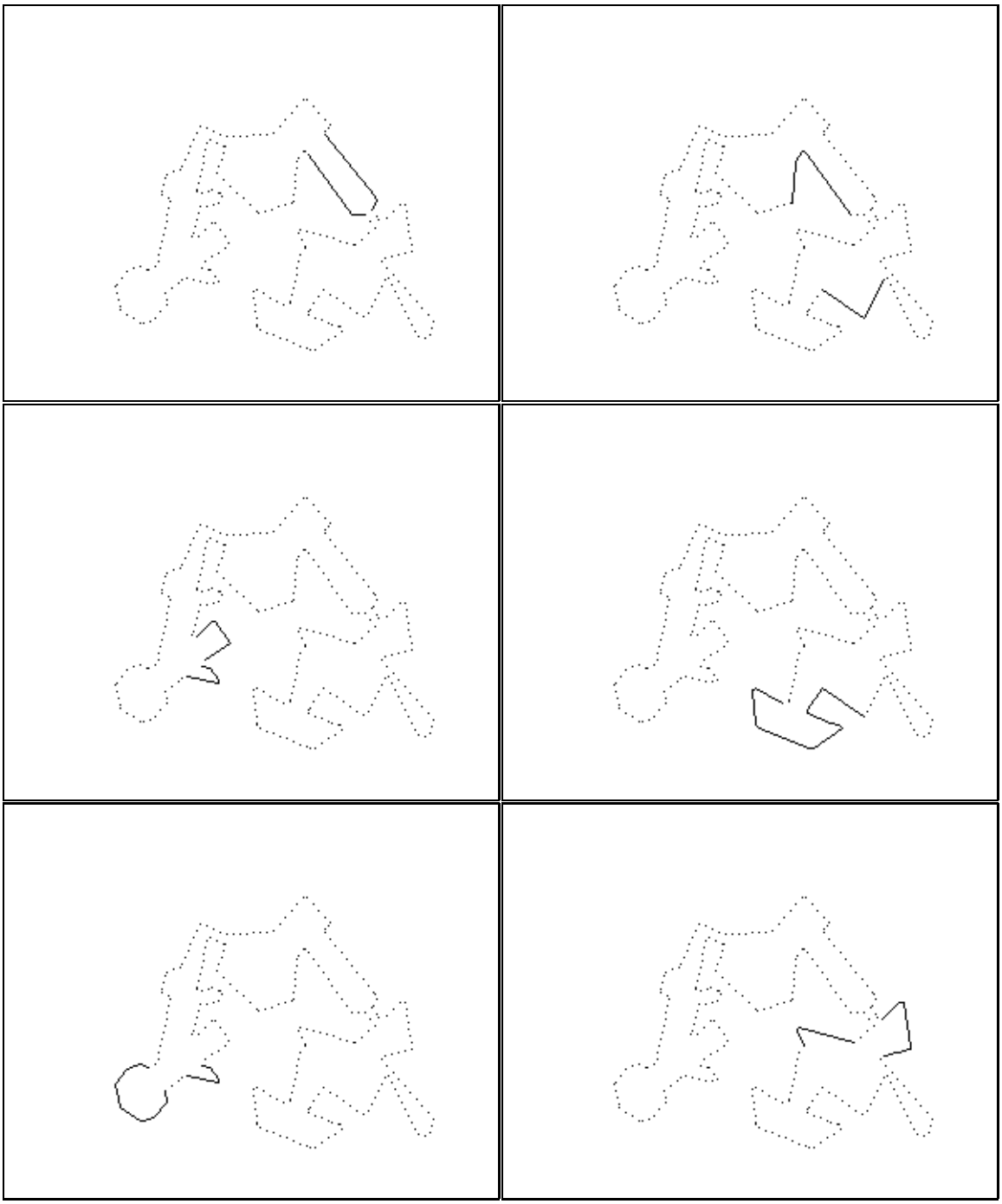

Figure 22: This figure shows the second through seventh pairs of convex groups selected by the grouping system, from the picture in Figure 21. The second and seventh pairs are correct.

Figure 23: A picture of our lab, used in the second set of three-dimensional tests. On the right are line approximations to the edges found in this image.

difference. We tried the following alternate values: For $P(d|O_1 = O_2)$, we used the distribution of lengths of occlusions caused by a circle, where originally we used $(MaxD - d)^2$. For $k$, the probability that two convex parts of different objects intersect, we used .5 and 0, instead of .2. For the maximum object diameter we tried 150 pixels in place of 300 pixels, and for the maximum diameter of a convex part we tried 75 pixels instead of 150. For the a priori probabilities that two convex parts of an object come from the same, adjacent, or non-adjacent sections, we had originally used $\frac{1}{3}$ for all three values. Instead, we used $\frac{1}{2}$ for each probability, with $\frac{1}{4}$ as the two corresponding probabilities. We also tried setting $P(int_i|adj, d, O_1 = O_2) = (MaxP - d)^2$ normalized into a probability distribution, instead of $P(int_i|adj, d, O_1 = O_2) = MaxP - d$. Finally, instead of attempting to verify hypotheses when indexing turned up 4 matches, we used 15 matches as the threshold.

We varied these parameters one at a time, and then all at once. For this final test we used $k = .5$ and the original values for the same, adjacent or non-adjacent part probabilities. We ran GROPER with all these parameter settings on the the second set of images used in recognition tests. We used error bounds of three pixels for distance, $sin\frac{\pi}{15}$ for angles, and we required that GROPER account for 50% of an object's perimeter before accepting a match. We chose these values since they allowed SEARCHER to perform accurately. We reasoned that if poor choices of parameters harmed GROPER it would still perform accurately, but would require more computation. This proved true, simplifying the comparison between different choices.

We also ran GROPER with some of its grouping constraints disabled. We ran GROPER without a distance contraint, without the intersection part of its orientation constraint, and without any orientation constaint. Finally, we ran GROPER without any of these constraints. This final system differed from SEARCHER because it still used small convex groups of nearly connected edges as its primitives in a search, instead of single lines.

Figure 24 shows the results of these tests. For the most part, alternate parameters do not make much difference in GROPER's performance. Disabling any of GROPER's grouping constraints had a much greater effect.

The greatest change comes from setting $k$ to .5 instead of .2. Since GROPER sets $P(d|O_1 \neq O_2) = k * P(d|O_1 = O_2) + (1 - k) * \frac{2d}{MaxDiameter^2}$, increasing $k$ has the effect of diminishing the

| Parameters with values different from the defaults | Average number of nodes explored |
|---|---|
| All default values | 30.6 |
| Distance distribution for circles | 30.2 |
| Distance to intersection constraint squared instead of linear | 36.2 |
| Probability parts intersect = 0 | 31.6 |
| Probability parts intersect = .5 | 45.8 |
| Max number of hypotheses we try to verify = 15 | 28.8 |
| Max object diameter = 150 | 28.8 |
| Max part diameter = 75 | 29.2 |
| Probability groups same convex section = .5<br>probability adjacent = .25, probability non-adjacent = .25 | 42.2 |
| Probability group same convex section = .25<br>probability adjacent = .5, probability non-adjacent = .25 | 26.6 |
| Probability group same convex section = .25<br>probability adjacent = .25, probability non-adjacent = .5 | 32.0 |
| Distance distribution for circles<br>Distance to intersection constraint squared instead of linear<br>Max number of hypotheses we try to verify = 15<br>Max object diameter = 150 Max part diameter = 75<br>Probability parts intersect = .5 | 29.0 |
| | |
| Null distance constraint | 78.8 |
| Null Intersection constraint | 63.6 |
| Null Orientation and intersection constraints | 64.8 |
| No grouping constraints | 459.8 |

Figure 24: The results of varying GROPER's parameters.

strength of the distance constraint (for $k = 1$, $P(d|O_1 \neq O_2) = P(d|O_1 = O_2)$ and there is no distance constraint). So, as $k$ increases too much the distance constraint becomes less effective. Since the system works well with $k = 0$ it seems that $k$ adds an unnecessary complication to the system.

These results also tell us that each of GROPER's grouping constraints played an important part in its success. Disabling either the distance or orientation contraints makes the system significantly slower. Disabling both constraints produces a much slower system, although searching through the space of convex sections of curves produces faster results than searching through collections of single edges.

# 6 Conclusions

This research has aimed at producing a grouping system that can assist in the recognition of three-dimensional objects in real scenes. Lowe has successfully done this by forming small groups of edges that have a special relationship such as parallelism or co-termination. We have attempted to use his basic approach to build a grouping system that orders the space of all sets of edges. We have done this mainly by adding new orientation constraints that allow us to estimate the likelihood that any set of edges all came from a single object.

We have built a grouping system, GROPER, using these constraints and found that it performs well on real, complex scenes of three-dimensional curved objects, and that comparable performance on two-dimensional scenes can result in a dramatic improvement in the performance of a full recognition system. We have also shown that grouping can improve the robustness of a recognition system. In situations where it is difficult to determine accurate error thresholds, many correct and incorrect object matches may pass our error bounds. This can occur because error thresholds are hard to determine a priori, and also because in some images some incorrect matches may account for more of an object's perimeter than some correct matches, for any error bounds. In these situations, grouping can provide a means of reducing mistakes, by leading a recognition system to discover correct matches that pass our error thresholds before it discovers incorrect matches that also meet our error bounds.

GROPER's grouping system arises from an attempt to understand the way that the image formation process produces constraints on the location and orientation of image edges. This analysis has led to some intuitions about what types of constraints can form an accurate grouping system. But it is the success of our implemented grouping system that provides evidence that we have understood some aspects of the way that the world makes grouping possible. This success suggests that the simplified world we have analyzed captures the most important aspects of the real world.

# A Making Random Objects

We built random objects out of random convex polygons. We wanted particularly to make sure that we constructed shapes with a random size, because this seemed like the factor most likely

to influence the size of the occlusions produced by the object. So we selected the area of the convex shape from a uniform random distribution from 0 to an arbitrary constant.

To determine the number of sides of a convex shape we picked a number between three and seven, with each number equally probable.

Then, to construct a random convex polygon, we started with a randomly chosen triangle, and added edges, one at a time, until we had enough. To make a random triangle, we chose a base with a random length. We then picked a number between 0 and $\pi$ for the angle between the base and one edge, and a number between 0 and $\pi$ minus the first angle for the angle between the base and the second edge. To add an edge, we first picked one of the polygon's edges at random, and removed it. We connected the polygon again by adding two randomly chosen edges, subject to the constraint that they must maintain the convexity of the polygon. We then scaled the polygon so that it would have the appropriate area.

To form an object with a specific number of convex parts, we created each convex part separately. To connect a convex part to an object, we randomly located each of them in an image-sized space until they intersected. We then joined them at the points of intersection, and removed all overlapping material.

# B Occlusions Caused by Circles

This appendix derives the probability distribution of the lengths of occlusions caused by randomly located circles. We assume that the radius of a circle is chosen from a uniform random distribution between 0 and 1. We then generate occlusions by randomly locating two random circles in the plane. If they occlude, then the distance from the beginning to the end of the occlusion is the random variable we describe.

Call the radius of circle one, $r_1$, and the radius of circle two, $r_2$. Without loss of generality, we can assume that circle one is centered at the origin, and that the center of circle two is no further than 2 from the origin. Let $d$ be the distance of the occlusion, and let $a$ be the distance separating the centers of the circles.

We wish to know $P(d = D)$, the density function of $d$, given that an occlusion occurs. We proceed by finding $P(d > D | r_1 = R_1, r_2 = R_2)$. Then we integrate over all values of $R_1$ and $R_2$, and divide by the probability that an occlusion occurs at all.

Suppose $r_1 = R_1$ and $r_2 = R_2$. Without loss of generality, assume that $R_1 > R_2$. Note that when $a \geq R_1$, $a = \sqrt{R_1^2 - \frac{d^2}{4}} + \sqrt{R_2^2 - \frac{d^2}{4}}$. This follows from simple geometry. When $a \leq R_1$, $a = \sqrt{R_1^2 - \frac{d^2}{4}} - \sqrt{R_2^2 - \frac{d^2}{4}}$. So, when $a = \sqrt{R_1^2 - \frac{D^2}{4}} + \sqrt{R_2^2 - \frac{D^2}{4}}, d = D$, and when $a = \sqrt{R_1^2 - \frac{D^2}{4}} - \sqrt{R_2^2 - \frac{D^2}{4}}, d = D$. In between these points, $d > D$. Therefore, $d \geq D$ iff $\sqrt{R_1^2 - \frac{D^2}{4}} + \sqrt{R_2^2 - \frac{D^2}{4}} \geq a \geq \sqrt{R_1^2 - \frac{D^2}{4}} - \sqrt{R_2^2 - \frac{D^2}{4}}$. So $P(d > D | r_1 = R_1, r_2 = R_2)$ equals the probability that $a$ falls in this range.

The area where the center of circle two can fall, creating a value of $a$ in this range is $\pi(\sqrt{R_1^2 - \frac{D^2}{4}} + \sqrt{R_2^2 - \frac{D^2}{4}})^2 - \pi(\sqrt{R_1^2 - \frac{D^2}{4}} - \sqrt{R_2^2 - \frac{D^2}{4}})^2$, and the total area in which the center of the second circle can lie is $4\pi$. Therefore $P(d > D | r_1 = R_1, r_2 = R_2) = \sqrt{R_1^2 - \frac{D^2}{4}}\sqrt{R_2^2 - \frac{D^2}{4}}$.

$$P(d > D) = \int_{\frac{D}{2}}^{1} \int_{\frac{D}{2}}^{1} \sqrt{R_1^2 - \frac{D^2}{4}} \sqrt{R_2^2 - \frac{D^2}{4}} dR_1 dR_2$$

Note that $R_1$ and $R_2$ must be greater than $\frac{D}{2}$ or an intersection of length $D$ can not occur. So,

$$P(d > D) = (\frac{1}{2}\sqrt{1 - \frac{D^2}{4}} - \frac{D^2}{8} cosh^{-1} \frac{2}{D})^2$$

To find $P(d = D)$ we take the derivative of one minus this figure, which is:

$$P(d = D) = -2(\frac{1}{2}\sqrt{1 - \frac{D^2}{4}} - \frac{D^2}{8} cosh^{-1} \frac{2}{D})(\frac{1}{4} \frac{1}{\sqrt{1 - \frac{D^2}{4}}}(-\frac{D}{2}) - \frac{D}{4} cosh^{-1} \frac{2}{D} + \frac{1}{4} \frac{1}{\sqrt{\frac{4}{D^2} - 1}})$$

This is the probability that $d = D$ when the center of circle two falls within 2 of the center of circle one. But only some of that time will an occlusion occur. So we must divide the above figure by the probability that an occlusion occurs to find the probability that $d = D$ given that an occlusion has occurred.

An occlusion occurs when: $R_1 - R_2 < a < R_1 + R_2$. That is, when the center of circle two falls somewhere in an area of size:

$$\pi(R_1 + R_2)^2 - \pi(R_1 - R_2)^2 = 4R_1R_2$$

So the probability of an occlusion occurring is: $R_1R_2$, for a given $R_1$ and $R_2$. Since the radii can range from 0 to 1, the total probability of an occlusion is:

$$\int_0^1 \int_0^1 R_1R_2 dR_1 dR_2 = \frac{1}{4}$$

# C The Likelihood of a $type_2$ Orientation

This appendix derives the probability of a $type_2$ orientation occurring between infinitesimally small groups with uniform random orientations.

We will call the two groups A and B. A and B appear on points in the plane, which we will also call A and B. If infinite, we will call the angle of A's projection $\alpha$, and of B's projection $\beta$. Two ray's will bound A's projection. We will call them $a_1$ and $a_2$. $b_1$ and $b_2$ will refer to the rays that bound B's projection. Without loss of generality, we can assume that A lies at the origin, and B lies on the x axis. We will call the angle between $a_1$ and the x axis $\theta$, and the angle between $b_1$ and the x axis $\psi$. Figure 25 illustrates these variables.

If either group has a finite projection, we can easily produce an answer. If group A has a finite projection, then the two groups' projections intersect only when point A lies inside B's projection. Since B's projection has a uniform, random orientation, it will have a probability of $\frac{\beta}{2\pi}$ of covering point A. Similarly, if B has a finite projection there will be a probability of

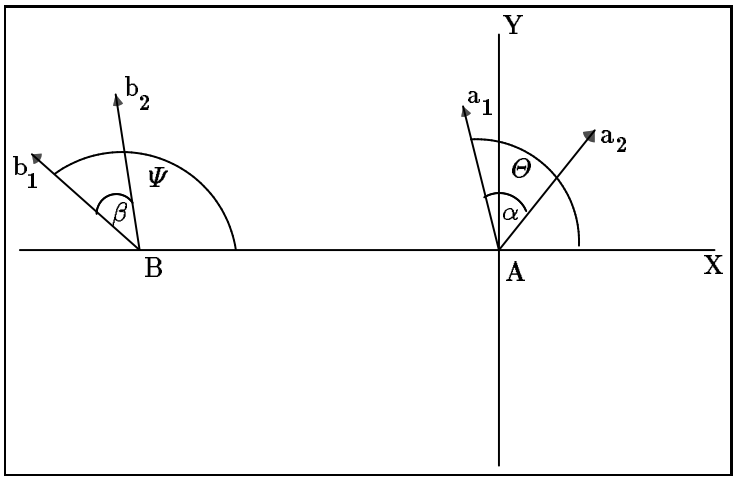


Figure 25: Two tiny groups with infinite projections.

$\frac{\alpha}{2\pi}$ of a $type_2$ orientation occurring. And if both groups have finite projections, they can not intersect.

Assume that both groups have infinite projections. We will determine the probability that these projections intersect by dividing the problem into eight different parts. First of all, we will consider separately the cases when $\alpha < \frac{\pi}{2}$ and when $\alpha > \frac{\pi}{2}$. Then for each of those cases, we will divide the possible orientations of A's projection into four groups, depending on the quadrant in which the ray $a_1$ falls.

First, suppose $\alpha < \frac{\pi}{2}$.

Suppose that $\frac{\pi}{2} < \theta < \pi$. The projections intersect only when either A falls in B's projection, or when $b_2$ intersects $a_1$. This happens if $\psi - \beta < \theta$, i.e. if $\psi < \theta + \beta$. This occurs whenever $\psi$ has a value between 0 and $\frac{\pi}{2} + \beta$. Furthermore, since $\theta$ has a uniform distribution between $\frac{\pi}{2}$ and $\pi$, the projections will also have a fifty percent chance of intersecting if $\psi$ has a value between $\frac{\pi}{2} + \beta$ and $\pi + \beta$. So the total probability of intersecting projections is:

$$\frac{\frac{\pi}{2} + \beta + \frac{\pi}{4}}{2\pi}$$

Suppose that $\theta < \frac{\pi}{2}$. A's projection may fall in one quadrant. This happens when $\alpha < \theta$. Or, A's projection may fall in two quadrants. In the first case, the projections will intersect whenever $0 < \psi$ and $\psi - \beta < \theta$. The chances that $\psi$ achieves an appropriate value are: $(\beta + \frac{\alpha + \frac{\pi}{2}}{2}) * \frac{1}{2\pi}$. Furthermore $\alpha < \theta$, occurs with probability $1 - \frac{\alpha}{\frac{\pi}{2}}$. So, the total probability of $\alpha$ being less than $\theta$, and the projections intersecting, is:

$$(\beta + \frac{2\alpha + \pi}{4}) * (1 - \frac{2\alpha}{\pi}) * \frac{1}{2\pi}$$

In the second case, when $\theta < \alpha$, the projections intersect only when $b_1$ or $b_2$ lie in between $a_1$ and $a_2$. This has an $\frac{\alpha+\beta}{2\pi}$ probability of occurring. So, over all, the probability that $\theta$ will be less than $\alpha$, and that the two projections will then intersect is:

$$\frac{(\alpha+\beta)}{2\pi} * \frac{2\alpha}{\pi} = \frac{2\alpha^2 + 2\alpha\beta}{2\pi^2}$$

Combining these two results, we find that, if $\theta < \frac{\pi}{2}$, the probability of projections intersecting is:

$$(\beta + \frac{\pi}{4} + \frac{\alpha^2}{\pi})\frac{1}{2\pi}$$

Next, suppose $a_1$ falls in the quadrant where x is positive and y negative. In this case, $a_2$ makes an angle of $\alpha + (2\pi - \theta)$ with the x axis. An intersection occurs if B's projection extends anywhere within this range. This happens if $\theta - \alpha < \psi$. If $\psi < \beta$, then A falls inside B's projection. So, the total chances of projections intersecting are:

$$(\frac{\pi}{4} + \alpha + \beta)\frac{1}{2\pi}$$

Finally, suppose $a_1$ falls in the quadrant where x and y are both negative. First of all, B may fall in A's projection, with probability $\alpha\frac{2}{\pi}$. Secondly, if this does not happen, an intersection occurs if $\psi > \theta - \alpha$, or if $\psi < \beta$. Given that B does not fall in A's projection, $\theta$ will randomly range from $\frac{3\pi}{2}$ to $\frac{3\pi}{2} - \alpha$. So an intersection occurs when $\psi$ falls in a range that varies between $\pi$ to $2\pi$ and $\frac{3\pi}{2} - \alpha$ to $2\pi$, or $\psi$ may fall in the range from 0 to $\beta$. Overall, the chances of this happening are:

$$\frac{(\pi+\beta) + (\frac{\pi}{2} + \beta + \alpha)}{2} * \frac{1}{2\pi} * \frac{\frac{\pi}{2} - \alpha}{\frac{\pi}{2}}$$

When we add in the chances of A's projection including B, we find that the total probability of the projections intersecting is:

$$(\frac{3\pi}{4} + 3\alpha + \beta - \frac{2\beta\alpha}{\pi} - \frac{\alpha^2}{\pi})\frac{1}{2\pi}$$

$a_1$ is equally likely to fall in any of the four quadrants, so, averaging the four results above, we find that the probability of projections intersecting when $\alpha < \frac{\pi}{2}$ is:

$$\frac{\frac{\pi}{2} + \beta + \alpha - \frac{\alpha\beta}{2\pi}}{2\pi}$$

Similar, tedious reasoning produces the same result when $\alpha > \frac{\pi}{2}$.

## Acknowledgements

I would like to especially thank Eric Grimson and Tomás Lozano-Pérez for allowing me to use a lot of code from their recognition system. Eric Grimson also provided a good deal of

guidance and many useful suggestions reflected in this paper. My understanding of this topic also benefitted greatly from discussions with Todd Cass, Dave Clemens, Liz Edlind, Tomás Lozano-Pérez, Jim Mahoney, Shimon Ullman, and especially Whitman Richards. The image processing was done on a hardware/software environment developed by Keith Nishihara and Noble Larson.

General Motors generously provided me with support during much of the time I worked on GROPER. This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by an Office of Naval Research University Research Initiative grant under contract N00014-86-K-0685, and in part by the Advanced Projects Agency of the Department of Defense under Army contract number DACA76-85-C-0010 and under Office of Naval Research contract N00014-85-K-0124.

## References

[1] Ayache, N. and Faugeras, O., 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.

[2] 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.

[3] Bolles, R. and Cain, R., 1982. "Recognizing and Locating Partially Visible Objects: The Local-Feature-Focus Method." *The International Journal of Robotics Research*, 1(3):57-82.

[4] Brooks, R., 1981. "Symbolic Reasoning Among 3-D Models and 2-D Images." *Artificial Intelligence*, 17:285-348.

[5] Canny, J., 1986. "A Computational Approach to Edge Detection." *IEEE Transactions on Patern Analysis and Machine Intelligence*, 8(6):679-698.

[6] 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.

[7] Cox, I., J. Rehg, and S. Hingorani, 1992, "A Bayesian Multiple Hypothesis Approach to Contour Grouping," *European Conference on Computer Vision*, pp. 72–77.

[8] Dolan, J. and E. Riseman, 1992, "Computing Curvilinear Structure by Token-based Grouping," *IEEE Conference Computer Vision and Pattern Recognition*, pp. 264–270.

[9] 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.

[10] Goad, C., 1983. "Special Purpose Automatic Programming for 3D Model-Based Vision." *Proceedings ARPA Image Understanding Workshop.*: 94-104.

[11] Grimson, W.E.L. and Lozano-Pérez, T., 1987. "Localizing Overlapping Parts by Searching the Interpretation Tree." *IEEE Transactions on Patern Analysis and Machine Intelligence*, 9(4):469-482.

[12] Grimson, W.E.L., 1987. "Recognition of Object Families Using Parameterized Models." *Proceedings of the First International Conference on Computer Vision*:93-101.

[13] Grimson, W.E.L., 1988. "The Combinatorics of Object Recognition in Cluttered Environments using Constrained Search." *Proceedings of the Second International Conference on Computer Vision*:218-227.

[14] Grimson, W.E.L. and Huttenlocher, D., 1988. "On the Sensitivity of the Hough Transform for Object Recognition." *Proceedings of the Second International Conference on Computer Vision*:700-706.

[15] Hoffman, D. and Richards, W., 1984. "Parts of Recognition." In *Visual Cognition*, edited by Pinker. Cambridge, MIT Press.

[16] Huttenlocher, D. and Ullman, S., 1989. "Recognizing Solid Objects by Alignment with an Image." Cornell University TR 89-978.

[17] Huttenlocher, D. and Wayner, P., 1992, "Finding Convex Edge Groupings in an Image," *International Journal of Computer Vision*, **8**(1):7-29.

[18] Jacobs, D., 1988. *The Use of Grouping in Visual Object Recognition.* MIT Technical Report 1023.

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

[20] Kalvin, A., Schonberg, E., Schwartz, J., and Sharir, M., 1986. "Two-Dimensional, Model-Based, Boundary Matching Using Footprints." *The International Journal of Robotics Research*, 5(4):38-55.

[21] Knoll, T. and Jain, R., 1986. "Recognizing Partially Visible Objects Using Feature Indexed Hypotheses." *IEEE Journal of Robotics and Automation*, 2(1):3-13.

[22] Kohler, W. 1959. *Gestalt Psychology.* New York: Mentor Books.

[23] Lamdan, Y., J.T. Schwartz and H.J. Wolfson, 1990, "Affine Invariant Model-Based Object Recognition," *IEEE Transactions Robotics and Automation*, **6**:578–589.

[24] Lamdan, Y. & H.J. Wolfson, 1988, "Geometric Hashing: A General and Efficient Model-Based Recognition Scheme," *Second International Conference Computer Vision*, pp. 238–249.

[25] Lowe, D. 1985. *Perceptual Organization and Visual Recognition.* The Netherlands: Kluwer Academic Publishers.

[26] Mahoney, J., 1987. *Image Chunking: Defining Spatial Building Blocks for Scene Analysis*, MIT AI TR–980.

[27] Marr, D. and Hildreth, E., 1980. "Theory of Edge Detection." *Proceedings, the Royal Society of London*, B207:187-217.

[28] 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.

[29] Pavlidis, T. and Horowitz, S., 1974. "Segmentation of Plane Curves." *IEEE Transactions on Computers*, C(23):860-870.

[30] Santalo, L., 1953. *Introduction to Integral Geometry.* Paris: Hermann & Cie Editeurs.

[31] Saund, E., 1992, "Labeling of Curvilinear Structure Across Scales by Token Grouping," *IEEE Conference Computer Vision and Pattern Recognition*, pp. 257–263.

[32] Schwartz, J. and Sharir, M., 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.

[33] 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.

[34] Thompson, D. and Mundy, J. 1987. "Three-Dimensional Model Matching from an Unconstrained Viewpoint." *Proceedings of the IEEE Conference on Robotics and Automation*:208-220.

[35] Tucker, L., Feynman, C., and Fritsche, D., 1988. "Object Recognition Using the Connection Machine." *Proceedings of CVPR*:871-878.

[36] Turney, J., Mudge, T., and Volz, R., 1985. "Recognizing Partially Occluded Parts." *IEEE Transactions on Patern Analysis and Machine Intelligence*, 7(4):410-421.

[37] Ullman, S., 1986. "An Approach to Object Recognition: Aligning Pictorial Descriptions." M.I.T. AI Memo 931.

[38] Van Hove, P., 1987. "Model-Based Silhouette Recognition." *Proceedings, IEEE Workshop on Computer Vision*:88-93.

[39] Wallace, A., 1987. "Matching Segmented Scenes to Models Using Pairwise Relationships Between Features." *Image and Vision Computing*, 5(2):114-120.

[40] Wayner, P.C., 1991, "Efficiently Using Invariant Theory for Model-based Matching," *IEEE Conference Computer Vision and Pattern Recognition*:473-478.

[41] Weiss, I., 1988, "Projective Invariants of Shape," *DARPA IU Workshop*, pp. 1125–1134.

[42] Witkin, A. and Tenenbaum, J., 1983. "What is Perceptual Organization for?." *Proceedings, IJCAI-83*:1023-1026.

[43] Witkin, A. and Tenenbaum, J., 1983. "On the Role of Structure in Vision." In *Human and Machine Vision*, edited by Beck, Hope and Rosenfeld. New York: Academic Press.

[44] Zucker, S., 1983, "Cooperative Grouping and Early Orientation Selection," In *Physical and Biological Processing of Images*, edited by Braddick and Sleigh. Springer-Verlag, Berlin.