RECOGNIZING PARTIALLY VISIBLE OBJECTS USING FEATURE INDEXED HYPOTHESES

Thomas F. Knoll

Ramesh Jain

Computer Information and Control Engineering
and
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, Michigan 48109

June 1985

CENTER FOR RESEARCH ON INTEGRATED MANUFACTURING

Robot Systems Division

COLLEGE OF ENGINEERING
THE UNIVERSITY OF MICHIGAN
ANN ARBOR, MICHIGAN 48109-1109

TABLE OF CONTENTS

Abstract

A basic task in computer vision is to recognize the objects in an image. Most computer vision systems do this by matching models for each possible object type in turn, recognizing objects by the best matches. This is not ideal, as it does not take advantage of the similarities and differences between the possible object types. The computation time also increases linearly with the number of possible object types, which can become a problem if the number of types is large. This paper describes a new recognition method, the feature indexed hypotheses method, which takes advantage of the similarities and differences between object types, and is able to handle cases where there are a large number of possible object types in sub-linear computation time. A two-dimensional occluded parts recognition system using this method is described.

1. Introduction

A common task in computer vision is, given an image of an unknown object, determine its identity, position and orientation. The unknown object is usually assumed to be identical to one of a set of possible objects that the vision system has been taught to recognize, usually by showing it one or more sample views of each object. The most straightforward way to recognize the objects would be to match models for each possible combination of identity, position and orientation to the image. Besl and Jain [BeJ85] show that such an exhaustive search is hopeless for the case of three-dimensional objects. Because of the complexity of this task, researchers have instead turned to recognizing objects by their features. In a limited domain with only a few objects, features can be manually selected; in a more complex application, the system should select the most pertinent features for recognition.

The recognition task can be classified based on difficulty in several ways. One way is based on the degree of uncertainty allowed in the object's position and orientation. This can range from no uncertainty, to uncertainty only in its two-dimensional position, to uncertainty in both its two-dimensional position and rotation, all the way up to uncertainty in its three-dimensional position and orientation. This paper will concentrate initially on the class of problems which are basically two-dimensional, where objects are assumed to lie in a known plane, in one of at most a few stable attitudes. Possible extensions to three-dimensional cases will be discussed later.

The task can be further classified based on the complexity of interactions allowed between objects. In the simplest case, each object to be recognized must be completely visible and surrounded by background. In a more complex case, objects are allowed to

touch but not overlap. In the most general case, which is considered in this paper, objects are allowed to touch or partially occlude one another.

A local feature is a feature that depends only on a subset of the object. If an object in the scene is only partially visible, it can still be recognized using local features that happen not to depend on the hidden portions of the object. To reliably recognize partially occluded objects, several local features should be used, so, with luck, at least one will depend only on the object's visible portion.

Many methods have been proposed to recognize partially visible objects. Most operate by selecting local features, or combinations of local features, and searching for these features in the image [Per78], [BoC82], [TMV85]. They vary in the type of feature they use, how the features are chosen, and how the features are searched for in the image. The next section gives an overview of some of the previous work in object recognition.

Most work on the occluded objects problem has concentrated on the case where there is only one object type in the scene. This is known as the bin-of-parts problem, where identical parts are stacked in a bin, and the program has to determine the position and orientation of the top-most part so that a robot can pick it up. Algorithms developed for this problem can be extended to cases of multiple objects types, simply by running the algorithm multiple times, once for each possible object type. This is not ideal, as it does not consider or take advantage of the similarities and differences between possible objects. Also the recognition time increases linearly with the number of possible objects, after a fixed time for preprocessing. See Figure 1. This can become a problem as number of possible object types increases.

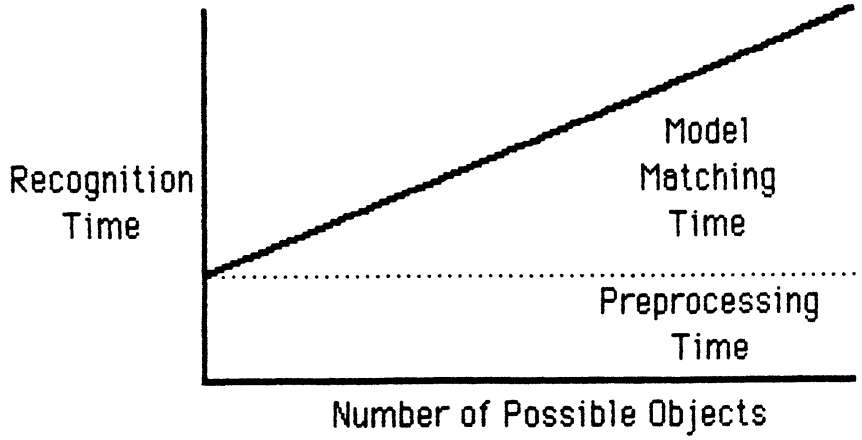

Figure 1: Using the usual methods, the recognition time grows linearly with number of possible objects, after a fixed preprocessing time.

This is how most vision systems operate, however. Models for each possible object are matched to the image in turn. A score is computed for each possible object and the unknown object is recognized by the highest score.

Previous recognition methods have suffered the problem of linear growth of the recognition time, and do not perform optimally because they ignore the similarities and differences between the possible object types. This paper describes a new method to recognize objects, the feature indexed hypotheses method, which avoids these problems. By taking advantage of the similarities and differences between the possible object

types, the recognition time growth rate is reduced to as the square root of the number of possible objects. The feature indexed hypotheses method breaks the recognition process into two phases: hypothesis generation, and hypothesis verification. Features are selected so that the total time spent is minimized. This paper describes how such features can be selected, and presents an automatic selection algorithm. The method's feasibility is demonstrated by the results of a prototype two-dimensional occluded parts recognition system.

2. Literature Survey

2.1. Global Techniques

The classic technique for object recognition is pattern recognition using global feature vectors [Kan74]. The idea is to describe the object with a list of numerical values which are invariant to translation, rotation, and possibly scaling. In the system's learning stage, feature vectors for each possible object type are stored. When recognizing an unknown object, its feature vector is compared to the feature vectors of each object type. The object is usually recognized using a nearest neighbor or likelihood ratio classifier.

One set of features that can be used are moment invariants [Hum62], [DBM77]. The moments of an object are given by

$$m_{pq} = \int_{-\infty}^{\infty} \rho(x,y) x^p \; y^q \; dx \; dy \quad (p,q = 0,1,2,\cdots)$$

where $\rho(x,y)$ is the density function of the object, which can take on the values of one

or zero in the case of binary silhouettes. These moments are then algebraically combined to produce a sequence of moment invariants. This sequence of moment invariants is then used as a feature vector.

An ad hoc set of features is used by the SRI vision module [GlA79]. The module includes a camera which sees a binary silhouette of the object. From this silhouette several global descriptors are computed. These include area, perimeter, number of holes, area of holes, maximum and minimum radii from the object's centroid, the ratio of these two radii, and others. These descriptors are formed into a feature vector, and the object is recognized using a nearest neighbor classifier.

Another feature set is the normalized Fourier descriptor. These have been used to recognize aircraft from silhouettes [WaW80]. An object's Fourier descriptor is found by taking the Fourier transform of its boundary curve, treating the boundary as a periodic complex function with real and imaginary parts corresponding to the x and y coordinates. The descriptor is then normalized to a standard location, rotation angle, size, and boundary curve trace starting point. The unknown object's normalized descriptor is compared to each normalized descriptor in the object library. The object is identified by the closest match.

The above methods work by computing the entire feature vector and then comparing it to each possible object model one by one. This can be slow if the feature vector is complex and time consuming to compute, or if there are a large number of possible object types to match with the feature vector. One way to speed up the recognition is to use a binary decision tree [Bel78]. Beginning at the root node, a single global feature value is computed and compared to a threshold. Different branches of the tree are then taken depending on the result. Any possible objects that have this

feature value on the opposite side of the threshold are thus eliminated from consideration. One by one, more features are computed and compared to thresholds, reducing the possible object set, until only single possible object is left. See Figure 2 for an example binary decision tree. Decision trees can operate faster than nearest neighbor classifiers because often only a subset of the features need be computed, and because whole groups of possible objects can be eliminated at once, instead of checking each one separately. Yachida and Tsuji [YaT77] used a similar method, where the system proposes the next feature to look for, and where to look for it, based on the current possible object set.

2.2. Local Techniques

Global techniques are unable to recognize objects that are only partially visible, which can happen either when the object only partially in the field of view, or when it is occluded by another object. Global features computed for part of an object have in general no relation to those computed for the entire object. To avoid this problem, many researchers have instead used local features, which depend only on portions of objects, to perform recognition.

Perkins [Per78], [Per80] describes a system which is able to handle cases of overlapping objects. First edge points are detected in a gray scale image. These edge points are linked and stored as chain codes. These edges are approximated by straight lines and circular arcs by fitting lines to the chain code data in Θ–s (angle-arc length) space. In the learning stage, the system is shown each possible object, and the system stores the detected curves as object models. When recognizing objects, image curves are matched against model curves. Potential matches are suggested by comparing the curve's general features such as length, total angular change, etc. Potential matches

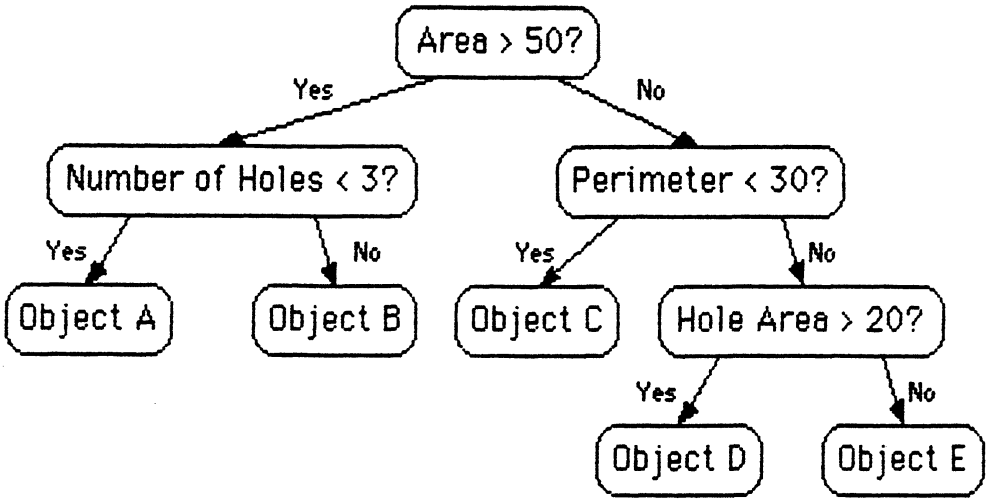

Figure 2: An example binary decision tree. Global features are computed and compared to thresholds one by one until the object is recognized. This can reduce the feature computation time; for example, to recognize Object B, only two of the four features need be computed.

are then checked using cross-correlation in Θ–s space. Matches here are then verified by computing a transformation from model coordinates to image coordinates, and checking for edges in the expected directions at a list of points spaced along the model's perimeter. McKee and Aggarwal [McA77] constructed an earlier system using similar methods.

Wallace, et al. [WMF81] describe a system that recognizes aircraft silhouettes using local shape descriptors. The system assumes that only one object appears in the image and that it is completely visible. First the boundary of the object is traced and is processed to reduce quantization error effects. Peaks in the boundary's curvature function are found, along with valleys of low curvature separating these peaks. The local shape descriptors consist of the arc length between peaks, and the angle change between valleys. Objects are recognized by matching the unknown object's list of lengths and angles with the lists for each model object. A distance measure was defined between descriptor lists, and the unknown object is recognized by the closest match. Because of the simple matching procedure used, the system is not able to handle the case of overlapping objects, in spite of the use of local features.

Several systems have been described that use relaxation techniques to match two-dimensional shapes [Dav79], [RPR81], [Rut82], [BhF84]. These systems work by first extracting a set of local features in both the images of the model objects, and the image of the unknown object. Typically these are arc length and angle features similar to those used by Wallace, et al., described above. All the features in the unknown image are matched to all the features in the model images. A vector is stored for each feature in the unknown image giving the estimated probability that it corresponds to each feature in the model images. These vectors are then updated using a relaxation technique, with nearby compatible labelings supporting each other. Each feature in the unknown image is recognized as its vector entry with the highest probability at the end of the labeling process.

The Hough transform can be generalized to detect arbitrary two-dimensional shapes [Bal81]. The Hough transform uses a multi-dimensional accumulator array, one

dimension for each of the object's degrees of freedom. Each cell in the array corresponds to a combination of object parameters. For each edge point in the image, all possible parameter sets of object position and orientation that are compatible with the edge location and direction have their accumulators incremented. After all the edge points have been processed, the accumulator array is scanned for the largest value. If this value is above a threshold, the object is recognized with the given location and orientation parameters. Multiple possible objects are handled by repeating this process, once for each possibility, and recognizing the object with the largest maximum accumulator value.

Turney, et al. [TMV85], describe an algorithm to recognize and locate partially visible two-dimensional objects using a subtemplate based version of the Hough transform. Instead of using edge points as in the normal Hough transform, overlapping segments of boundary are used. For each segment, a vector is stored pointing to the object's centroid. Whenever a subtemplate matches in the image, an accumulator at the position pointed to by the subtemplate's vector is incremented. See Figure 3. After all the object's subtemplates have been matched against the edges in the image, the accumulator with the largest value, if the value is above a threshold, is considered to be the position of the object.

One problem with Hough transforms is that false peaks may be generated. If a subtemplate contains a feature that occurs several times in the object, the subtemplate will match several times in the image, incrementing several accumulators, only one at the correct location. The problem is compounded when two or more of the possible objects contain many common features. Turney, et al., reduce this problem by assigning each subtemplate a weight based on its saliency. See Figure 4 for an

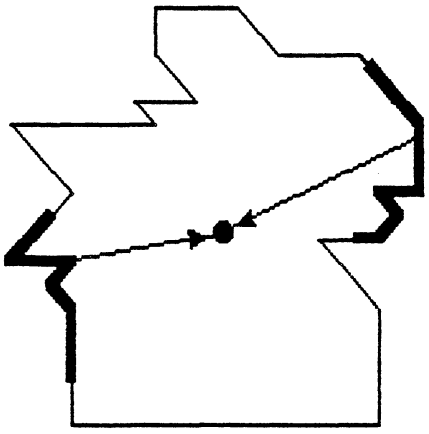

Figure 3: Matched boundary segments vote for an object identity and location in Turney, et al.'s, system.

example of salient features. Subtemplates that match many times are given low weights, while subtemplates that match less often are given higher weights. This weighting scheme increases the importance of the unique features of an object, and decreases the importance of its more common features.

Grimson and Lozano-Perez [GrL83] describe a system that is able to recognize objects from sparse range and surface normal data. The system assumes that only one object is in the field of view. Object models are stored in a CAD-type data base, and are assumed to have discrete faces. The system works by pairing range data points to

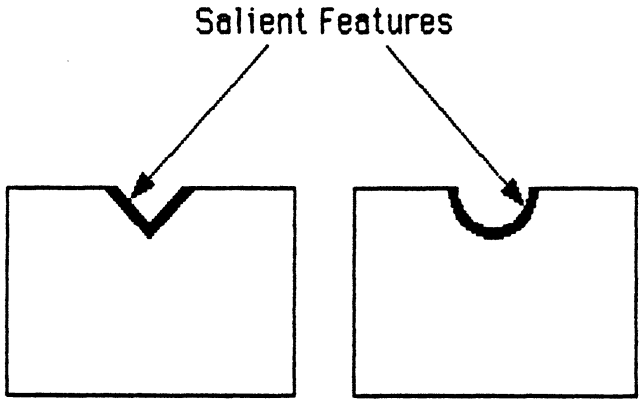

Figure 4: An object's salient features are those which distinguish it from other objects. The indicated segments are the salient features for these two possible objects. These segments would be given a higher weight in Turney, et al.'s system.

faces in the object models. If there are m known objects, with nj faces each, and s range data points, there are

$$\sum_{j=1}^{m} (n_j)^s$$

combinations of range point-model face pairings. The system tests these combinations

using a tree search. Not all these combinations need be tested because face distance and normal constraints are used to prune almost all the combinations. Still, the number of combinations that need to be tested grows rapidly with object complexity. For those combinations not ruled out, the system tries to find an object position and orientation consistent with the face assignments. If a consistent transformation is found, the object is recognized.

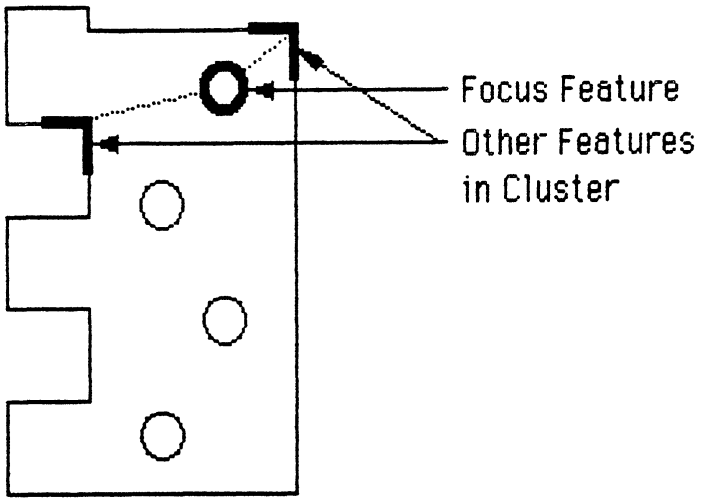

Figure 5: An example feature cluster searched for in Bolles and Cain's local-feature-focus method. The combination of the round hole and the two corners uniquely identifies and orients the object.

Bolles and Cain [BoC82], describe the Local-Feature-Focus Method, which is an algorithm to recognize and locate partially visible two-dimensional objects. First the image is scanned to detect local features. In the example presented in their paper, three types of features were detected in the binary images: round holes, convex ninety degree corners, and concave ninety degree corners. The position of each feature is recorded, along with its size and orientation, if any.

When each object was being learned, the algorithm searched for a cluster of local features in a relative configuration that did not occur elsewhere in the object or in any other possible object. One feature in the cluster was selected as the "focus" feature. See Figure 5 for an example feature cluster. When searching for this object in the image, the focus feature is first searched for. If it is found, its neighborhood is searched for the remaining features in the cluster. If they are found, and their relative configuration is consistent with the configuration found in the learning stage, the object is hypothesized to exist at the location. A template for the object is then translated and rotated by the required amounts and is matched to the image. If the match is good enough, the algorithm declares that the object exists at the location.

2.3. Decomposition and Analysis Techniques

Object recognition may be aided by decomposing objects into simpler parts. One method of decomposition uses graphic-theoretic clustering [ShH79], [Sha80]. Fischler and Elschlager [FiE73] matched complex objects such as human faces by first matching component parts, such as eyes, noses, mouths, etc., and then measuring how close their relative positions were to the expected values.

Another simplification technique reduces two dimensional shapes to skeletons. The first such technique was the medial axis transform, but this was sensitive to noise.

A more recent and more robust technique is smoothed local symmetries [BrA83]. A smoothed local symmetry is a sort of local centerline of the shape. After the skeleton is found, the object can be recognized by matching its skeleton to the skeletons of model objects. It is not clear how this matching should be done, however.

Many researchers have tried to construct intelligent vision systems. These systems often use world knowledge to help recognize objects, such as where in the scene an object usually is, and how it relates to other objects. For a survey of such systems,

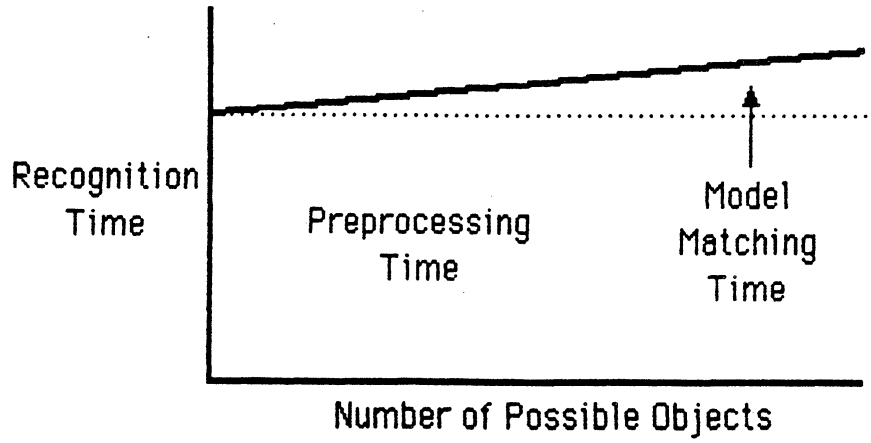

Figure 6: Using global feature based methods, more time is usually spent computing rather than comparing the feature vectors. Thus the recognition time grows only slowly with the number of possible objects.

see Binford [Bin82].

2.4. Search Strategies

Most object recognition strategies work by first detecting some features in the input image. Then object models are fit to these features to test if the object is present. If there is more than one possible object type, each model object is fitted in turn, and the object model with the best match is recognized. With this search strategy, the recognition time increases linearly with the number of possible object types, after the constant time for initial feature detection.

This is generally not a problem with global feature techniques, as most of the recognition time is spent computing rather than comparing the feature vectors. See Figure 6. Computing a global feature usually means scanning a two-dimensional image with thousands of pixels, whereas comparing two feature vectors is usually a simple Euclidean distance computation between a few numbers. Even when there are hundreds of possible objects, most of the recognition time is usually spent extracting features. Decision trees are able to speed up both the feature extraction and feature comparison stages. In the ideal case, the comparison stage time can be made to increase only as the logarithm of the number of object types. Even so, decision trees usually do more good by reducing the average number of features that have to be extracted, than by speeding up the comparison stage. Also decision trees use global features, and are not easily extended to use local features.

The linear growth rate of the recognition time may be a problem in systems that use local features. In these systems, a significant amount of time is spent matching each model, and the recognition time can become long if there are a large number of possible objects. One method to reduce this growth is to match the objects in two

stages, as in Khan and Jain's system [KhJ84]. First the objects are assigned to a class, and second to a particular object in that class. In the example shown in Figure 7, first an object could be first recognized to be 4-wheeled vehicle, and then to be car. This is faster because the system need not waste time matching the motorcycle model, if the object is already known to be some kind of 4-wheeled vehicle. It is difficult to apply this technique in many cases, as objects often cannot be assigned to easily recognized classes.

In effect, what the feature indexed hypotheses method will do is form classes based on features. This differs from Khan and Jain's method in that the objects may belong to multiple classes. There is some support for this idea from the field of cognitive psychology. A theory on how letters are recognized in the context of words by the human brain was presented by McClelland and Rumelhart [McR81]. They were trying to explain why letters are easier to recognize in the context of words than in the context of nonsense letter combinations. They proposed a model with nodes for all words, and nodes for each letter in each character position. Compatible nodes are connected by links that increase activation, and incompatible nodes are connected by links in inhibit activation. For example, each letter node has activation links with word nodes that have the letter in the correct position. Figure 8 shows these links for a simple case with a four word vocabulary and a four letter alphabet. (Links connecting incompatible nodes are not shown because they do not relate to the feature indexed hypotheses method). The activation level of a letter node is initially determined by low-level feature detectors. Active nodes then increase the activation of compatible nodes, and decrease the activation of incompatible nodes. The feedback to the letter nodes from the compatible and incompatible word nodes is why it is easier to recognize letters in the context of words. What is interesting about this theory is

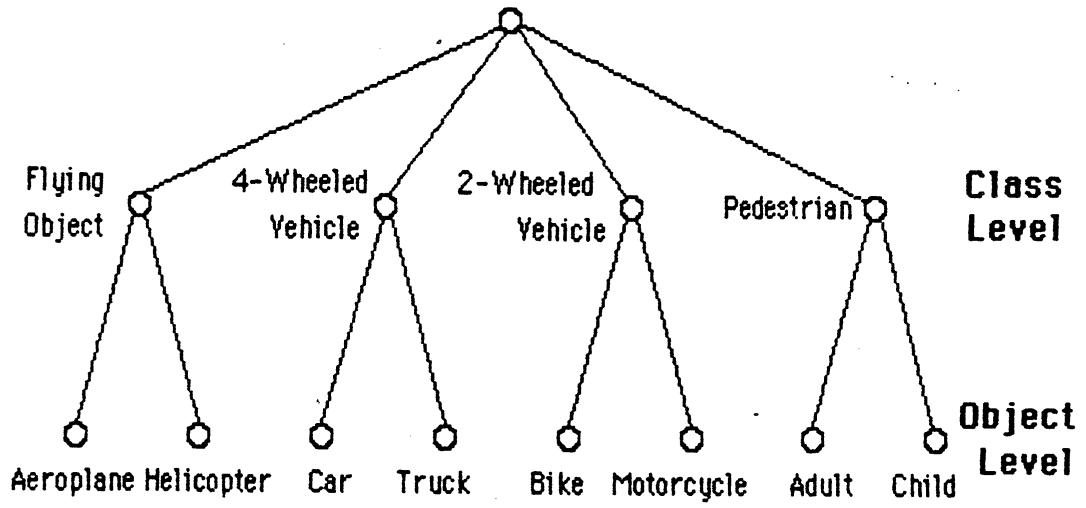

Figure 7: Khan and Jain's system recognizes objects at two levels. First the object is recognized as belonging to a class, and then as a specific member of that class.

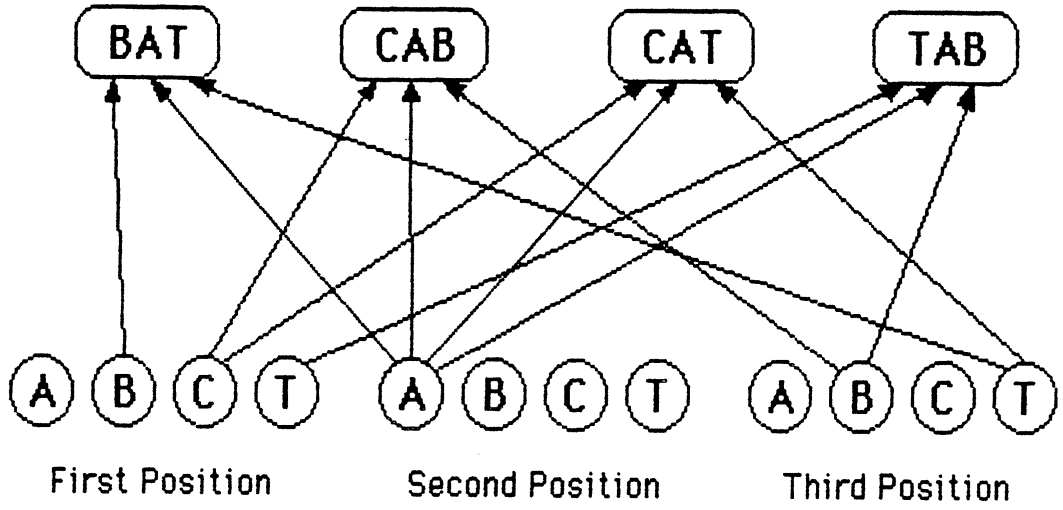

Figure 8: McClelland and Rumelhart's model of letter recognition in the context of words. Shown are the links connecting letter nodes with all compatible word nodes, for a simple case with a four letter alphabet and a four word vocabulary.

that each letter node activates a list of compatible word nodes. In the feature indexed hypotheses method, each feature will activate a list of compatible hypotheses.

3. The Feature Indexed Hypotheses Method

Previous object recognition methods do not take full advantage of the similarities and differences between objects on the possible object set. By looking for only unique features or feature clusters, the recognition time grows linearly with the number of

possible objects. These methods also tend to fall apart when the possible object set contains many similar objects, as unique features may be difficult or impossible to find.

This section describes a new method to recognize objects, the feature indexed hypotheses method, which avoids these problems. Recognition time grows only as the square root of the object set, and unique features are not required. Instead of unique features, features common to several objects in the object set are used. For each feature, a list is kept of where it occurs in each object type. When a match is found for a feature in an image, objects are hypothesized for each object identity and orientation in the feature's list. See Figure 9 for an example of the hypotheses generated by a feature match. Each of these hypotheses is then tested using a template match, to determine which, if any, are correct.

Bolles and Cain [BoC82], assumed that the cost of verifying hypotheses was high compared to the cost of generating them, and thus it was desirable to generate the best hypotheses possible. This may not be the case. The cost of searching for a feature in an image is often comparable to, or greater than, the cost of a hypothesis test. Feature matches can be slow because often the entire image must be searched, and the matching processes at each image point is non-trivial. Hypothesis tests can be faster because no search is involved. All that is required is a verification at a sampling of points that the predictions of the hypothesis are fulfilled in the image.

It may be possible to increase the speed of the system by generating many hypotheses, most of them false, and letting the hypothesis test weed out the bad ones. For example, suppose two objects contain a common feature. The image could be searched for this common feature. If found, two templates could be tested, one for each object. This may be faster than searching the image two separate unique features, one

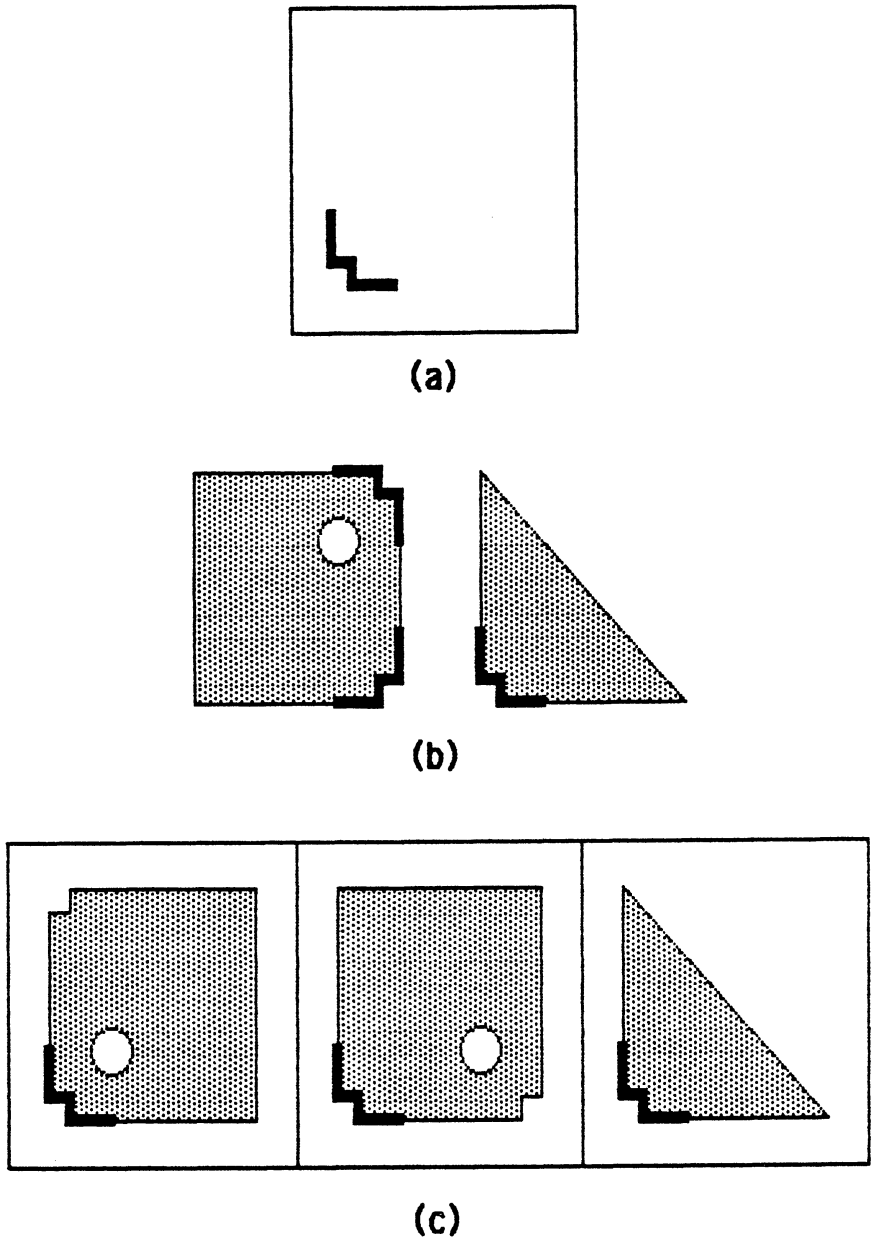

Figure 9: Example of the feature indexed hypotheses method. (a) Feature match in image of unknown object. (b) Occurrences of feature in possible object set. (c) Hypotheses generated by feature match.

for each possible object.

To show this, let us compute the total cost for recognizing an object. Let us assume for the purposes of this calculation that it is possible to select features that match exactly the desired number of times and in the desired places in the possible object set. This can only be approximated in real situations, and how well it can be approximated is beyond the scope of this paper.

Let p be the number of possible objects types, and assume that all types are equally likely. Because of occlusion, not all the features on an object may be visible. Thus it is desirable for the feature set to match more than once on each possible object. Let r be this desired redundancy. That is, each object should be matched by features in r places. The total number of matches in the possible object set is thus pr . If n is the total number of times each feature matches in the set of possible objects, then number of features that need to be searched for is pr/n .

The number of hypothesis tests made is equal to the number of features searched for, pr/n , times the average number of matches per feature per object, n/p , times the number of hypotheses generated per matched feature, n . If Cf is the cost of a feature match, and Ch is the cost of of a hypothesis test, the total cost of recognizing the object is:

$$C_t = C_f \times \frac{pr}{n} + C_h \times \frac{pr}{n} \times \frac{n}{p} \times n$$

$$= C_f \times \frac{pr}{n} + C_h \times rn$$

The feature matching portion of the cost decreases with n , while the hypothesis verification portion increases with n . There is an ideal number of matches per feature where

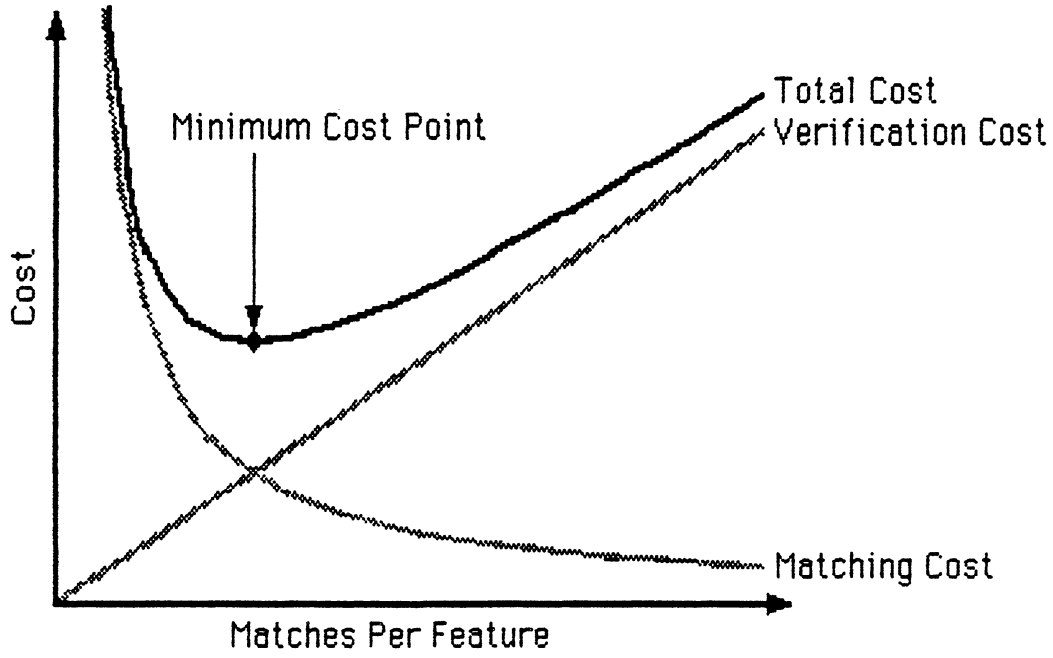

Figure 10: The cost of matching features decreases with the number of matches per feature, while the cost of verifying the generated hypotheses increases. The total cost can be minimized by selecting features that match the ideal number of times.

the total cost is at its minimum. See Figure 10. Using elementary calculus, minimizing Ct with respect to n gives:

$$C_t = \sqrt{C_f\, C_h} \times r \times \sqrt{p}$$

when

$$n = \frac{\sqrt{C_f}}{\sqrt{C_h}} \times \sqrt{p}$$

This must of course be rounded up to down to an integer, as a feature can match only an integer number of times.

Note that the ideal number of matches per feature is proportional to the square root of the number of possible objects. Note also that the recognition cost grows only as the square root of the number of possible objects, instead of linearly, as with most other occluded objects recognition systems.

The above calculation assumed implementation on a serial computer, and would have to be modified if the algorithm were implemented on a parallel computer. The feature indexed hypotheses method could be efficiently implemented on a parallel computer by assigning a processor to each feature, so all features could be matched in parallel. A processor could also be assigned to each hypothesis generated, so all hypotheses could also be tested in parallel.

The feature indexed hypotheses method has an advantage over other methods in that it does not require unique features. It becomes harder and harder to find unique features as the number of possible object types increases, especially if the objects are similar. For example, Figure 11 shows two similar objects, the only difference being that object A has a notch taken out of one corner. A unique feature for object A would be the corner with the notch taken out. But object B has no unique features. Object recognition methods that require unique features fail with this possible object set. The feature indexed hypotheses method has no problems with this object set. It

simply uses a feature common to both objects, hypothesizes both objects, and lets the hypothesis test distinguish between them.

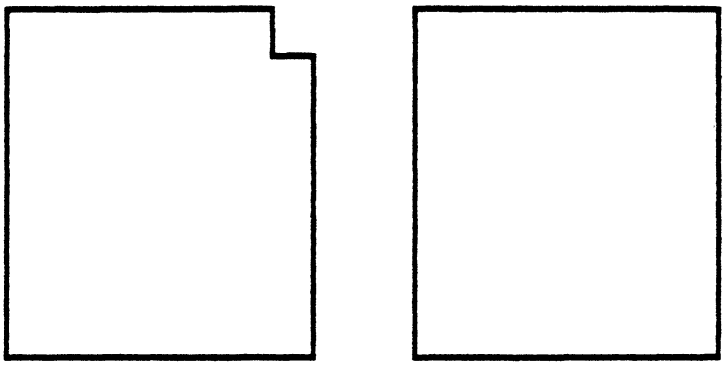

Figure 11: Object A as a unique local feature (the corner with the notch), whereas object B has no unique local features (assuming the rotation angle of the objects are unknown). Object recognition systems that require unique features fail with this object set, whereas the feature indexed hypotheses method works fine.

4. Prototype System Using Feature Indexed Hypotheses

This section describes a prototype system that illustrates the efficacy of the feature indexed hypotheses method. The prototype system recognizes two-dimensional objects of known scale in binary images. Below are descriptions of the feature matching algorithm, the automatic feature selection algorithm, and the hypothesis testing algorithm.

4.1. Feature Matching

The system's input is binary images, with black objects on a white background. Objects are assumed to be basically two-dimensional, having a fixed boundary shape each time they appear in an image. Three-dimensional objects with multiple stable attitudes are considered to be different objects for each attitude. Objects are allowed to touch or overlap one another, but the tilt caused by the overlap is assumed not to change the shape of the object significantly. Objects are also assumed to appear the same size each time they show up in an image.

Boundaries are traced using an eight neighbor chain code. Chain codes have a problem where the length of a line segment varies based on its angle relative to the sampling grid, as in Figure 12. This is known as chain code error. The system reduces this error using the methods of Wallace and Wintz [WaW80]. The coordinates of each pixel along the boundary are averaged with the coordinates of its neighbors. This reduces the ragged nature of the boundary. The boundary is then resampled at one pixel intervals along its length. These samples (which do not necessarily have integer coordinates) are stored as the boundary.

Features consist of fixed lengths of boundary. The feature length is input to the system as a parameter, l. Each feature has a characteristic position and angle. The

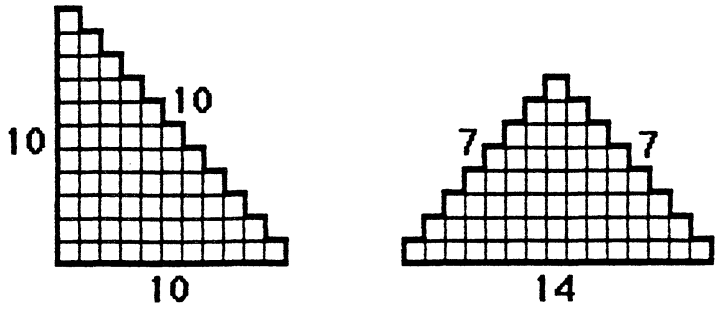

Figure 12: An example of chain code error. Line segment lengths depend on the object's angle relative to the sampling grid.

characteristic position is taken to be the centroid of the boundary segment, and the characteristic angle is taken to be the angle between the starting and ending coordinates. See Figure 13. When comparing two features, one is translated and rotated so that its characteristic position and rotation match those of the other. The features can then be compared using a simple distance measure. The distance between two features is taken to be the sum of the distances between corresponding coordinates of the two overlaid features:

$$\text{Distance} = \sum_{j=1}^{L} \left[(x_{1,i} - x_{2,i})^2 + (y_{1,i} - y_{2,i})^2 \right]^{(\frac{1}{2})}$$

See Figure 14.

This matching technique is slightly different from cross correlation in Θ–s space, as used by [Per78] and [TMV85]. Cross correlation matches in angle space, normalizing the average angles, whereas this method works in coordinate space. Because angles are derivatives of coordinates, and derivatives tend to increase noise, the method used here should be less sensitive to noise.

When matching a feature to a boundary, the feature is matched with pieces of the boundary spaced every pixel. The minima of this distance function are potential match locations. These are compared to a threshold to see if the segments are similar

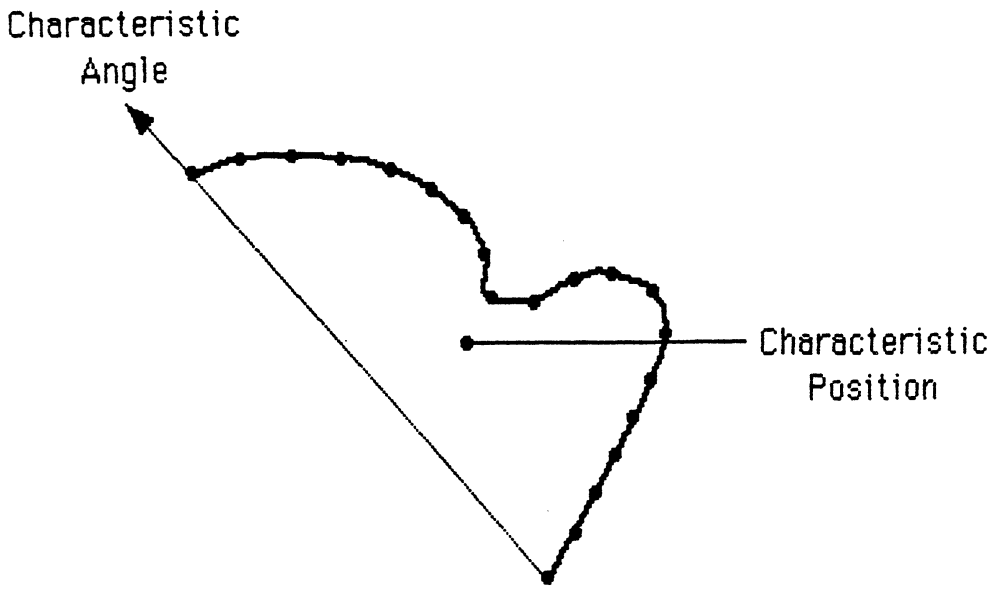

Figure 13: A boundary segment's characteristic position and angle.

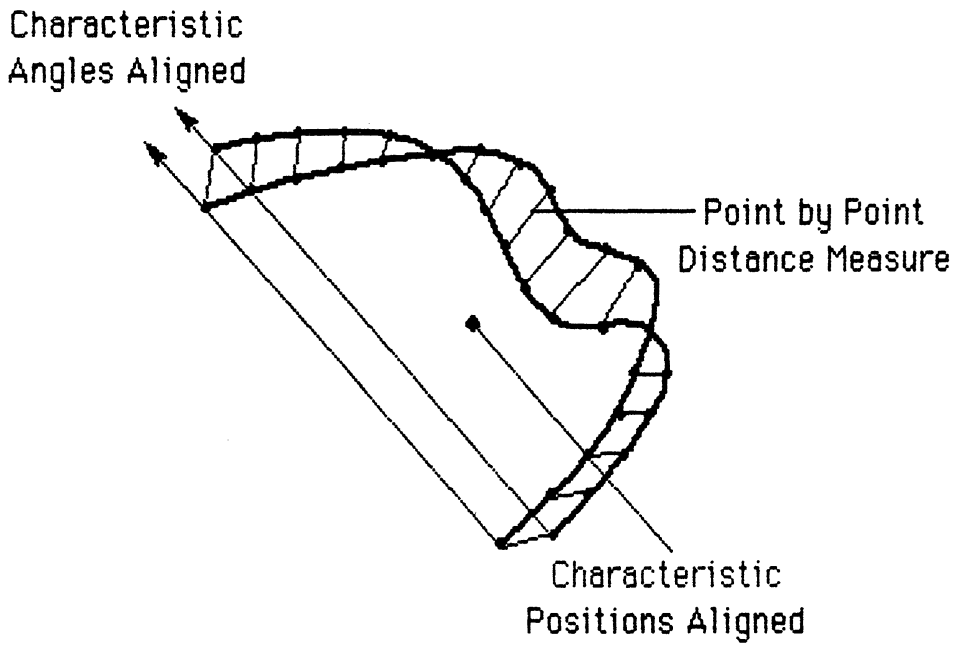

Figure 14: Distance between boundary segments is measured by aligning their characteristic positions and angles, and computing the point by point distance.

enough to be considered at match. For each match, a record is kept of the translation and rotation required to bring the segments into alignment.

4.2. Automatic Feature Selection

When the system is learning the possible object types, it must select a set of features that will be searched for when recognizing objects. This section describes how this selection takes place. In initial experiments, the perimeter of a typical object is five hundred pixels long. If potential features are sampled every ten pixels along the perimeter, there are about fifty potential features per object. If the system is to learn say, a hundred objects, there are about five thousand potential features to select from.

There are several goals to achieve when selecting features. First, as has been shown above, there is an optimal number of matches per feature. If a feature matches the possible object set too few times, the system will spend too much time matching features. On the other hand, if a feature matches the set too many times, too much time will be spent testing hypotheses. If the cost of a feature match is equal to the cost of a hypothesis test, which is not a bad assumption for the prototype system, the ideal number of matches per feature is approximately the square root of the number of possible object types.

Another goal in feature selection is to have the matches well separated on each object. Because objects may overlap or touch one another, not all of an object's boundary may be visible in the image. To allow the object to be recognized under these conditions, more than one feature match per object is desirable. The ideal number of matches per object depends on the amount of occlusion in the scenes, how reliable the system must be, and how fast it must operate. The desired redundancy is entered as a parameter, r, to the system. To be maximally effective against occlusion, the redundant matches should be well spread around each object. If all the redundant matches were concentrated on a small portion of the object's boundary, and that portion happened to be occluded, the object would not be recognized. For maximum reliability, the matches should be as far away from one another on each object as possible.

The final goal in feature selection is to have the matches well distributed across the possible object set. Ideally all objects should have the desired r number of redundant matches and no more. Additional matches beyond the desired redundancy only increase the time spent testing hypotheses, because hypotheses are generated for all

match locations in the possible object set, even if an object already has sufficient matches to be reliably recognized.

The desirability of a given feature depends on the other features in the search set. To find the optimal set of features, every subset of the potential feature set would have to be evaluated. With about five thousand potential features, the number of possible subsets is astronomical. Thus the task of selecting an optimal set of features is very complex, and is not possible to do in a reasonable time. The prototype system implements a non-optimal method to select features, using a "greedy" algorithm. In a greedy algorithm, features are added sequentially to the feature set, each time selecting the best feature based on the search set so far selected. The algorithm is non-optimal because once a feature is selected, it cannot be unselected, even if later selections make it obsolete. However, this algorithm does allow the selection of a fairly good search set, with an execution time that increases only as the square of the number of possible object types.

The system defines the **Desirability** of a feature as its **Benefit** to **Cost** ratio:

$$\textbf{Desirability} = \frac{\textbf{Benefit}}{\textbf{Cost}}$$

The cost of including a feature in the search set is the cost of matching the feature, Cf , plus the cost of testing the expected number of hypotheses it will generate. If all possible object types are equally likely, and the feature matches n times in the possible object set, the cost of including the feature is:

$$\textbf{Cost} = C_f + C_h \frac{n^2}{p}$$

Benefits are also associated with each feature. Benefits are awarded for the matches the feature makes. Features able to recognize many objects are thus given high benefit scores. The prototype system defines the benefit of a feature as:

$$\text{Benefit} = \sum_{i=1}^{n} M_i$$

where M_i is the match score of the *ith* match of the feature, which is defined below. The match score, M_i, for a match is defined as the Euclidean distance between the

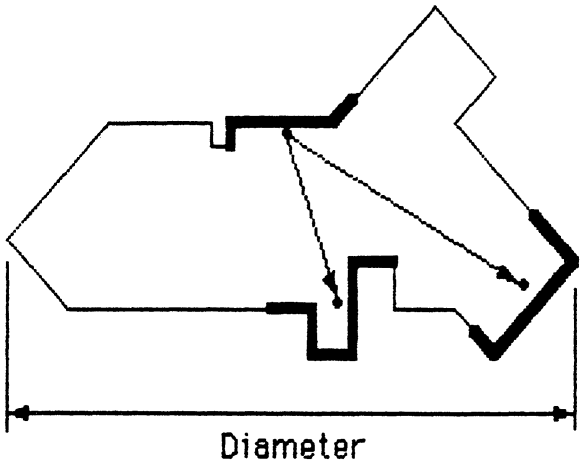

Figure 15: If an object has previously been matched at least once, but less than r times, the match score is found by dividing the distance to the nearest match by the object's diameter.

location of the match and the nearest match already on the object, divided by the object's diameter. (The diameter of an object is the distance between the two points farthest apart on the object). See Figure 15. If there are no matches currently on the object, the match score is set to one. Thus the system is encouraged to spread the matches around each object. To prevent the system from matching too many times on a object type, the match score is set to zero if the object has already been matched r or more times. When all possible objects have been matched r or more times, all match scores will be zero, and thus all features will have zero desirability.

The feature selection algorithm can be summarized as follows:

(1) Let P be the potential feature set. For each P_j in P, match against the entire possible object set. Keep a record of where each potential feature matches.

(2) Let S be the selected feature set. Set S to the null set.

(3) For each P_j in P, compute its desirability, D_j, based on the selected feature set S. Let P_{max} be the feature with the highest score, and D_{max} be its score.

(4) If D_{max} is zero, stop the algorithm. This will happen when all possible object types have been matched at least r times.

(5) Remove P_{max} from P and add to S. Go to step 3.

The system parameter, l, which controls feature length, can be adjusted if the feature selection algorithm is having trouble selecting good features. If the features are matching too many times, the feature length can be increased to make the features more specific. Similarly, the length can be reduced if the features are matching too

few times.

Whenever the possible object set is changed, the feature selection algorithm must be re-executed to find the new search set. If objects have been added to the possible object set, step 1 of the algorithm can be performed incrementally by matching the old potential features against the new objects, and the new potential features against the entire object set. Ideally the algorithm should then be started at step 2, re-selecting the entire search set. A faster method would be to start with step 3, and just add features to the current search set. How well this incremental selection works in practice, compared to the full selection algorithm, is a topic for further study.

4.3. Hypothesis Testing

The prototype system's hypothesis testing method is based on Bolles and Cain [BoC82]. The feature matching subsystem has declared that there may be an object at a given location and orientation. The job of the hypothesis testing subsystem is to see if this is true, or if it is a false alarm.

The object's perimeter is sampled periodically. At each sample, a short, directed line segment is drawn perpendicular to the boundary, from inside to outside, through and centered on the sample. These line segments are stored as the object's template. The length of the segments is a parameter, t, which controls the tolerance of the template match. See Figure 16.

When testing a hypothesis, the object's line segments are translated and rotated to the hypothesized location. The line segments are then traced in the binary image. If a segment traces a dark to light transition, this is positive evidence for the hypothesis, because an edge was found at the predicted location. If only dark is

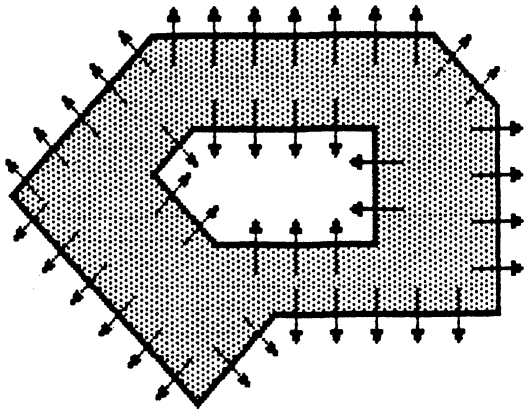

Figure 16: A template for hypothesis testing. The system checks for dark to light transitions at locations spaced around the object's perimeter.

found, this is neutral evidence; another object could be occluding the predicted edge. If a light to dark transition, or only light, is found, this is negative evidence for the hypothesis; in this case the missing edge cannot be explained by occlusion.

Positive or negative evidence is assigned a value from zero to one for each line segment. If a dark to light transition is found, positive evidence is assigned a value based on how close to the center of the line segment the transition is, with a value of one at the center, tapering linearly to zero at the endpoints. If there is no dark to light transition, negative evidence is assigned to the fraction of the first half of the

segment that is light.

Positive and negative are summed for all line segments in the template. These are then combined into a total hypothesis score by subtracting ten times the negative evidence from the positive. Negative evidence is given a much higher weight than positive, because it directly contradicts the hypothesis. Positive evidence only indicates that some edges were at the predicted locations--this may have happened by chance. A weighting factor of ten was found to work well for the test images.

If the total hypothesis score is negative, the hypothesis is rejected outright. Otherwise the result is stored until all hypotheses have been evaluated, and the one with the highest score is recognized. Additional objects can be recognized by marking the areas explained by the recognized object, and re-evaluating the remaining hypotheses, not counting positive evidence from the explained areas.

5. Results

The algorithms described above were implemented on an Apple Macintosh personal computer. (To convert execution times to equivalent VAX/780 times, divide by about five). Seven miscellaneous objects were selected. Binary 256 by 256 pixel images of the objects were taken, both as isolated objects, and overlapping each other. Some of these images are shown in Figure 17.

The feature length used was 60 pixels. Potential features were taken at ten pixel intervals along the perimeters of the sample images of the possible object. A total of 325 potential features for the seven possible objects were considered. Each of these features was matched against images for all possible objects, and a record was kept of where in the possible object set each feature matched. This cross matching took 50 minutes to complete. An example feature, and its matches in the possible object set, are

shown in Figures 18 and 19.

The remaining portion of the selection algorithm, were the features were selected using the greedy algorithm, took 17 seconds to complete. The cost of matching a feature, C_f, and the cost of testing a hypothesis, C_h, were both set to one. The desired redundancy, r, was set to five. A total of 18 features were selected, matching a total of 36 times in the possible object set. Table 1 shows on which objects each selected feature matches.

The recognition algorithm was then run on the image in Figure 17h. 7 of the 18 features in the search set matched, a total of 9 times, generating 22 hypotheses. An example match is shown in Figure 20. This feature matching stage took 30 seconds to complete. These hypotheses were then evaluated. Line segments ten pixels long, spaced every two pixels around each template's perimeter, were used. An example rejected hypothesis (resulting from the match shown in Figure 20) is shown in Figure 21. The hypothesis with the highest score is shown in Figure 22. After the areas explained by the hypothesis in Figure 22 were marked and the remaining hypotheses re-evaluated, the hypothesis shown in Figure 23 had the highest score. The time for hypothesis testing was 28 seconds for both objects, for a total recognition time of 58 seconds.

The program was also run the image shown in Figure 17i. This time, 25 hypotheses were generated, and 3 objects were recognized. 52 seconds were needed to generate the hypotheses, and 41 seconds to verify them, for a total recognition time of 93 seconds. The recognized objects are shown in Figure 24.

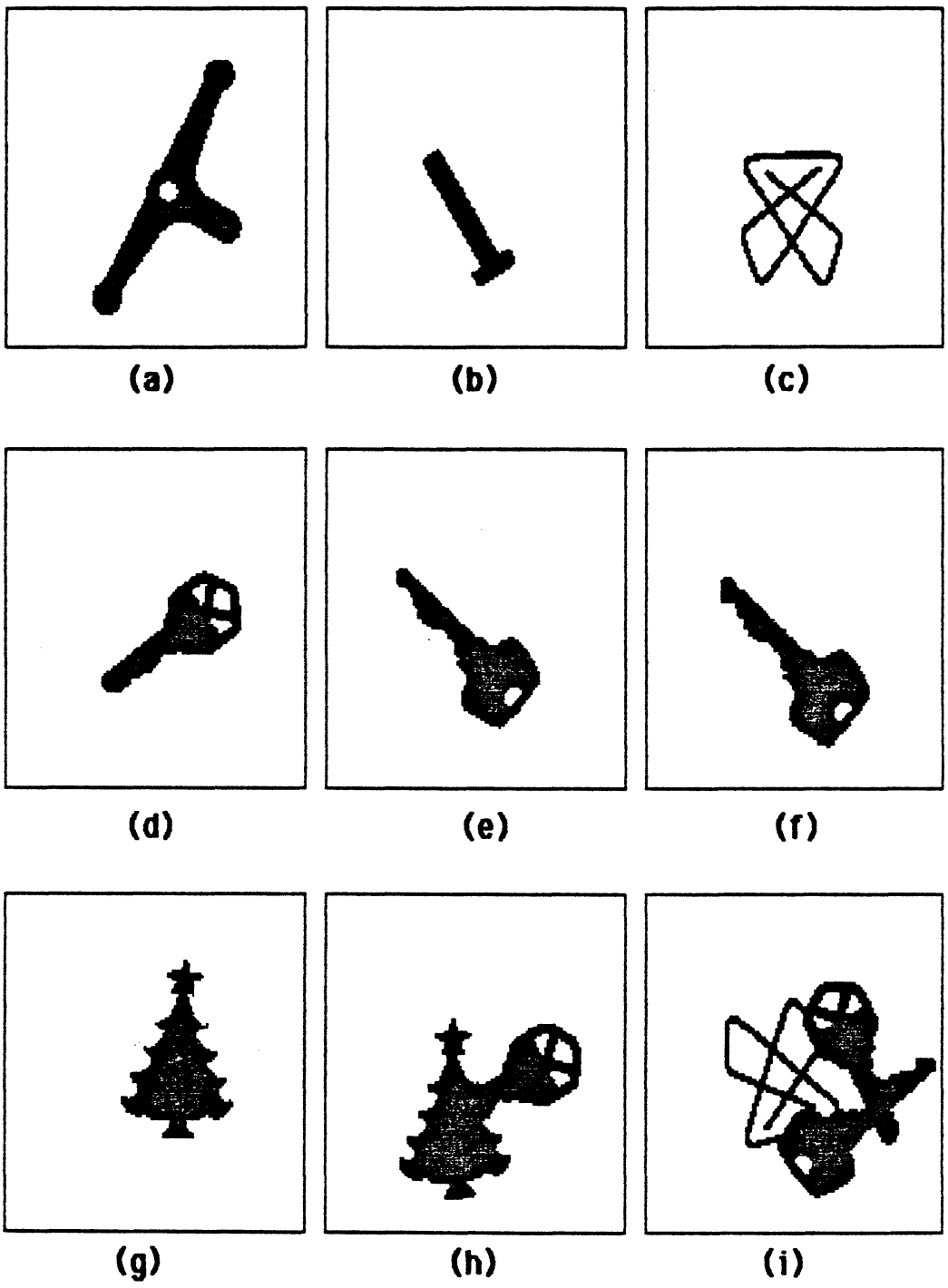

Figure 17: Input images for the recognition system. (a)-(g) Sample images of each possible object. (h)-(i) Test images.

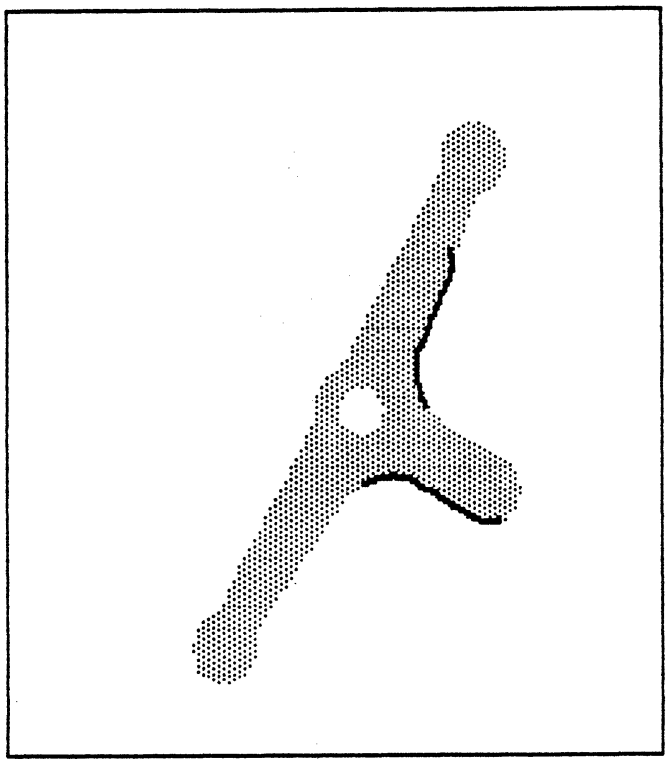

Figure 18: Features were 60 pixel long segments of boundary. This feature matched 3 times in the possible object set. Two the matches are shown here; the third match is shown in Figure 19.

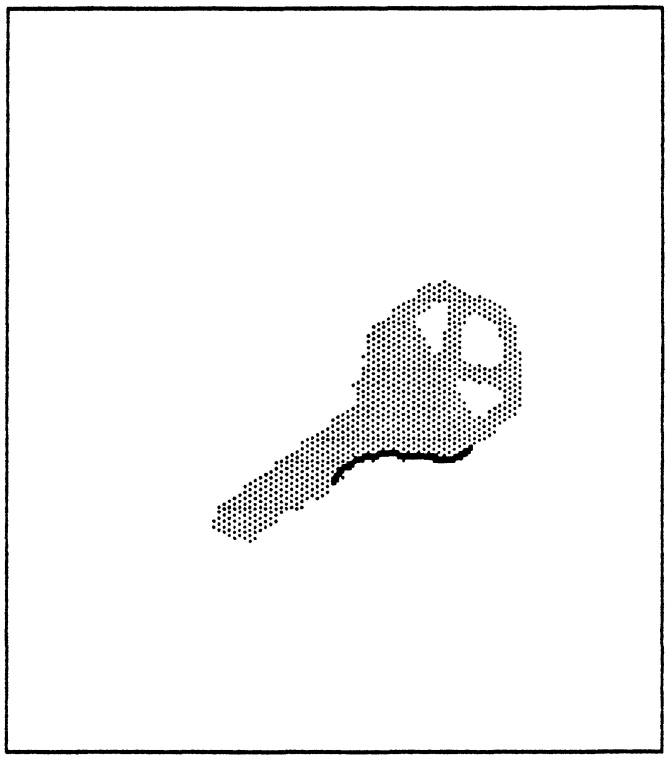

Figure 19: The third (and last) match of the feature shown in Figure 18.

Feature	Matches
1	a a a
2	a a d
3	b
4	b g
5	b
6	b
7	a b c
8	c c
9	c
10	d f
11	d d e f
12	d e f
13	c e g
14	e
15	e f
16	g g
17	g
18	g

Table 1: List showing on which object the selected features match in the possible object set. The object letters correspond to Figure 17. Feature number 2 is shown in Figures 18 and 19.

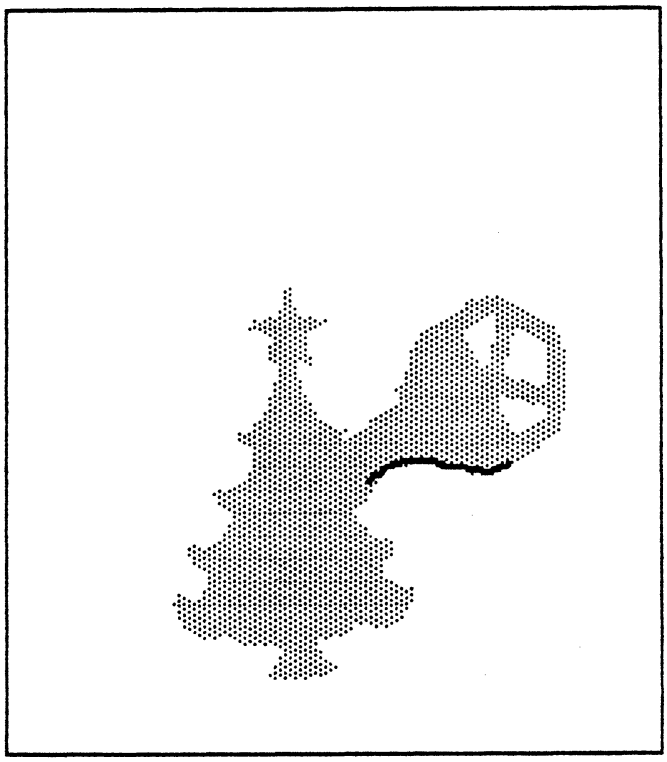

Figure 20: An example match in the test image. This match generated 3 hypotheses, one of which is shown in Figure 21.

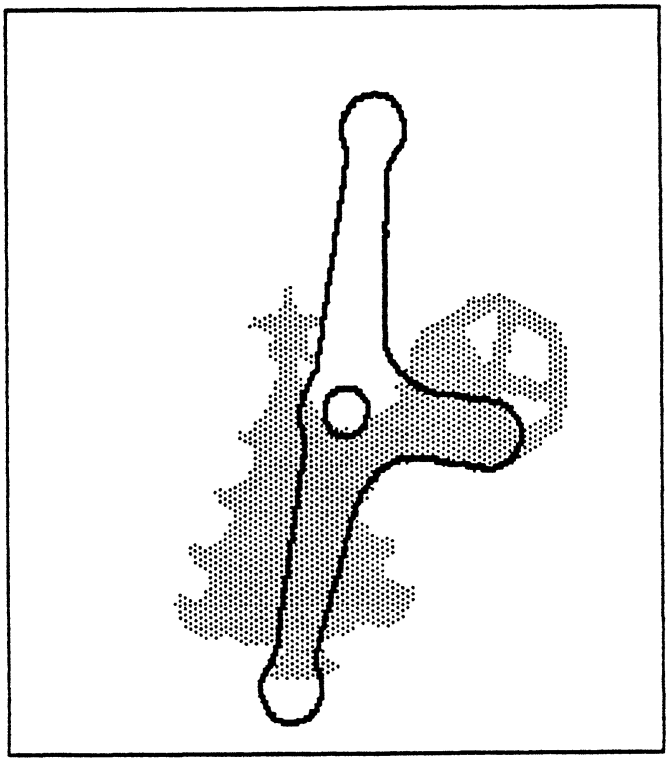

Figure 21: An example rejected hypothesis. This hypothesis is one of three generated by the match shown in Figure 20. Positive evidence: 53.6, negative evidence: 144.2, net score: -1388.4.

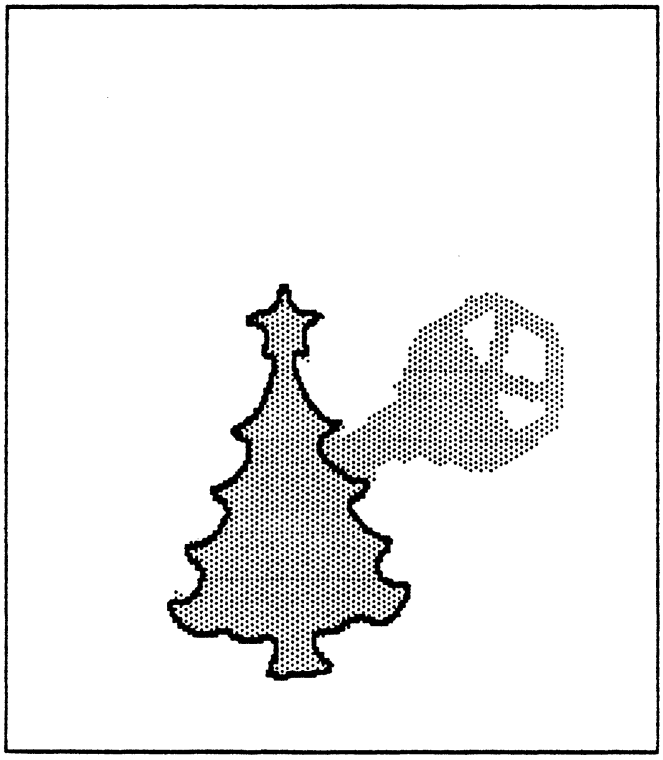

Figure 22: Highest scoring hypothesis. Positive evidence: 200.2, negative evidence: 0.8, net score: 192.2.

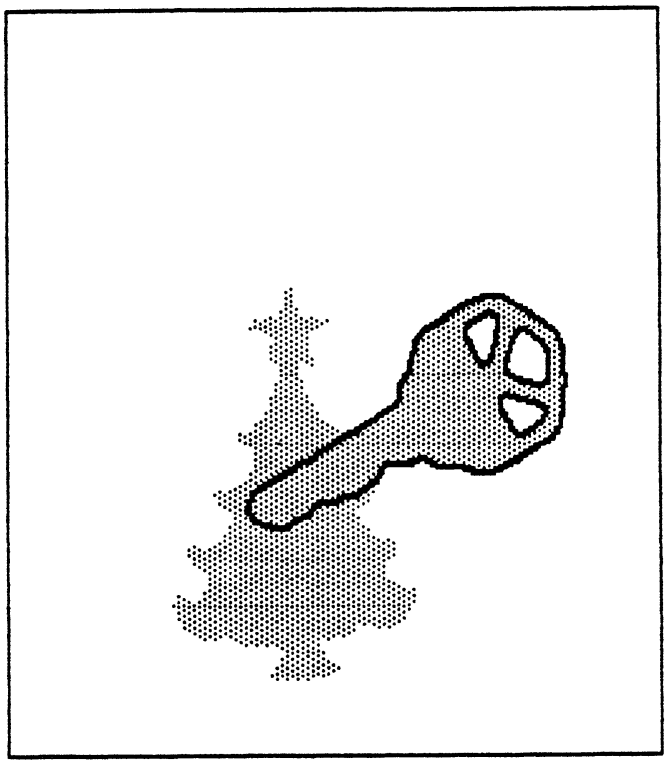

Figure 23: Highest scoring hypothesis, after the hypothesis in Figure 22 is accepted. Positive evidence: 179.2, negative evidence: 0, net score: 179.2.

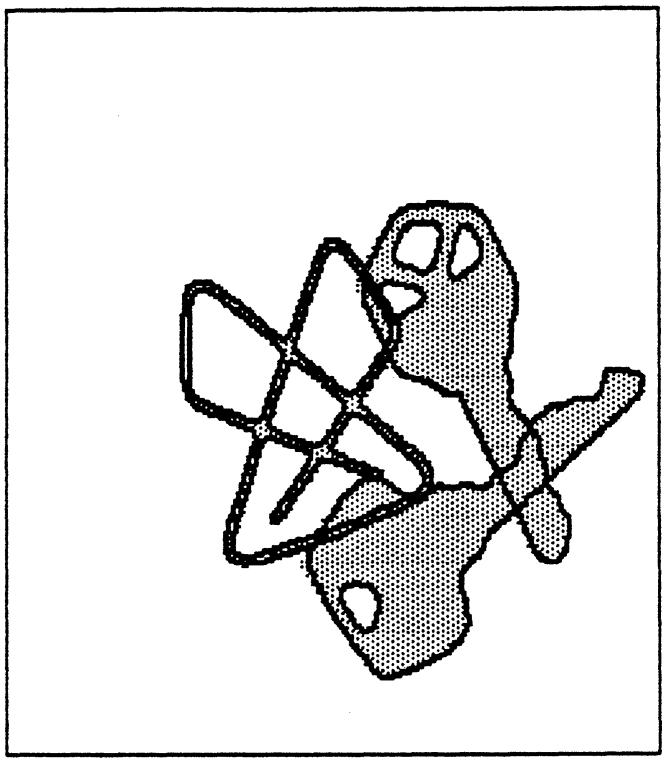

Figure 24: Shown are the three objects recognized in Figure 17i.

6. Conclusion

Most previous object recognition methods work by selecting a unique feature or feature cluster for each possible object, and searching for these in the image of the unknown image one by one. This has two main disadvantages: first the recognition time growing linearly with the possible object set, and second, unique features or feature clusters are required for each possible object.

This paper has presented a new method to recognize objects, the feature indexed hypotheses method, which solves these problems. This method uses features that occur several times in the possible object set, rather than unique features. This allows faster recognition in cases where there are a large number of possible object types--the recognition time grows only as the square root of the object set. And because unique features are not required, the method performs better in domains where the possible object set contains many similar objects, as in many industrial applications.

A prototype system, which recognizes two-dimensional objects in binary images, demonstrates the efficacy of this idea. The system implements an automatic feature selection algorithm, which selects, from a large set of potential features, an efficient set of features to use.

Future research will improve and extend the prototype system. Work is needed in the handing of rotationally symmetric objects, combining match information to improve hypotheses, and in the use of fuzzy logic in the feature matching. Additional work is also needed in alternate feature selection algorithms. A learning system, where the feature set is continually adjusted based on experience, is being studied.

The feature indexed hypotheses method could also be applied to three-dimensional object recognition, using depth maps as input. First the depth map would be scanned with three-dimensional feature detectors. These features would then be grouped into feature clusters, much as in Bolles and Cain [BoC82]. Hypothesis verification would be directly analogous to the prototype system's method. A depth map based the hypothesis is generated and compared to the actual depth map. Points where the generated map is closer, equal to, and farther than the actual map--are negative, positive, and neutral evidence, respectively.

7. References

[Bal81] D. H. Ballard, "Generalizing the Hough Transform to Detect Arbitrary Shapes," *Pattern Recognition,* Vol. 13, No. 2, 1981, pp. 111-122.

[BeJ85] P. J. Besl, R. C. Jain, "Three-dimensional object recognition," ACM Computing Surveys, Vol. 17, No. 1, March 1985.

[Bel78] D. A. Bell, "Decision Trees, Tables, and Lattices," *Pattern Recognition,* Plenum Press, New York, 1978, pp. 119-141.

[BhF84] B. Bhanu, O. D. Faugeras, "Shape Matching of Two-Dimensional Objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-6, No. 2, March 1984, pp. 137-156.

[Bin82] T. O. Binford, "Survey of Model-Based Image Analysis Systems," *The International Journal of Robotics Research,* Vol. 1, No. 1, Spring 1982, pp. 18-64.

[BoC82] R. C. Bolles, R. A. Cain, Local-Feature-Focus Method," *The International Journal of Robotics Research,* Vol 1. No. 3, Fall 1982, pp. 57-82.

[BrA83] M. Brady, H. Assda, "Smoothed Local Symmetries and Their Implementation," MIT Technical Report, 1983, 10pp.

[Dav79] L. S. Davis, "Shape Matching Using Relaxation Techniques," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-1, No. 1, January 1979, pp. 60-72.

[DBM77] S. A. Dudani, K. J. Breeding, R. B. McGhee, "Aircraft Identification by Moment Invariants," *IEEE Transactions on Computers,* Vol. C-26, No. 1, January 1977, pp. 39-46.

[FiE73] M. A. Fischler, R. A. Elschlager, "The Representation and Matching of Pictorial Structures," *IEEE Transactions on Computers,* Vol. C-22, No. 1, January 1973, pp. 67-92.

[GlA79] G. J. Gleason, G. J. Agin, "A Modular Vision System for Sensor-Controlled Manipulation and Inspection," *Proceedings of the 9th International Symposium on Industrial Robots,* Washington D. C., 1979, pp. 57-70.

[GrL83] W. E. L. Grimson, T. Lozano-Perez, "Model-Based Recognition and Localization From Sparse Range or Tactile Data," MIT A. I. Memo 738, August 1983, 46pp.

[Hum62] M. Hu, "Visual Pattern Recognition by Moment Invariants," *IRE Transactions on Information Theory,* Vol. IT-8, February 1962, pp. 179-187.

[Kan74] L. Kanal, "Patterns in Pattern Recognition: 1968-1974," *IEEE Transactions on Information Theory,* Vol. IT-20, No. 6, November 1974, pp. 697-722.

[KhJ84] N. A. Khan, R. Jain, "Matching an Imprecise Object Description with Models in a Knowledge Base," *Proceedings of the ICPR,* 1984.

[McA77] J. W. McKee, J. K. Aggarwal, "Computer Recognition of Partial Views of Curved Objects," *IEEE Transactions on Computers,* Vol. C-26, No. 8,

August 1977, pp. 790-800.

[McR81] J. L. McClelland, D. E. Rumelhart, "An Interactive Activation Model of Context Effects in Letter Perception: Part 1. An Account of Basic Findings," *Psychological Review,* Vol. 88, No. 5, September 1981, pp. 375-407.

[Per78] W. A. Perkins, "A Model-Based Vision System for Industrial Parts," *IEEE Transactions on Computers,* Vol. C-27, No. 2, February 1978, pp. 126-143.

[Per80] W. A. Perkins, "Simplified Model-Based Part Locator," *Proceedings of the 5th International Conference on Pattern Recognition,* December 1980, pp. 260-263.

[RPR81] W. S. Rutkowski, S. Peleg, A. Rosenfeld, "Shape Segmentation Using Relaxation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-3, No. 4, July 1981, pp. 368-375.

[Rut82] W. S. Rutkowski, "Recognition of Occluded Shapes Using Relaxation," Computer Graphics and Image Processing, Vol. 19, 1982, pp. 111-128.

[Sha80] L. G. Shapiro, "A Structural Model of Shape," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-2, No. 2, March 1980, pp. 111-126.

[ShH79] L. G. Shapiro, R. M. Haralick, "Decomposition of Two-Dimensional Shapes by Graph-Theoretic Clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-1, No. 1, January 1979, pp. 10-20.

[TMV85] J. L. Turney, T. N. Mudge, R. A. Volz, "Recognizing Partially Occluded Parts," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-7, No. 4, July 1985, pp. 410-421.

[WaW80] T. P. Wallace, P. A. Wintz, "An Efficient Three-Dimensional Aircraft Recognition Algorithm Using Normalized Fourier Descriptors," *Computer Graphics and Image Processing,* Vol. 13, 1980, pp. 99-126.

[WMF81] T. P. Wallace, O. R. Mitchell, K. Fukunaga, "Three-Dimensional Shape Analysis Using Local Shape Descriptors," *IEEE Transactions of Pattern Analysis and Machine Intelligence,* Vol. PAMI-3, No. 3, May 1981, pp. 310-323.

[YaT77] M. Yachida, S. Tsuji, "A Versatile Machine Vision System for Complex Industrial Parts," *IEEE Transactions on Computers,* Vol. C-26, No. 9, September 1977, pp. 882-894.