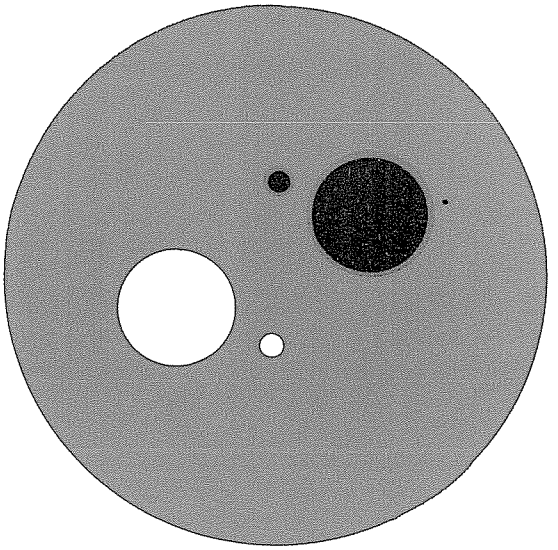
COMPUTER SCIENCES
DEPARTMENT

University of Wisconsin-
Madison



OBJECT RECOGNITION USING HOUGH PYRAMIDS

by

Charles F. Neveu
Charles R. Dyer
Roland T. Chin

Computer Sciences Technical Report #576

January 1985

Object Recognition Using Hough Pyramids

Charles F. Neveu
Sandia National Laboratory
Livermore, CA 94550

Charles R. Dyer
Computer Sciences Department
University of Wisconsin
Madison, WI 53706

Roland T. Chin
Electrical and Computer Engineering Department
University of Wisconsin
Madison, WI 53706

Index Terms: 2-D modeling, multi-resolution model-based matching, Hough transforms, pyramids

---

This work was supported in part by the National Science Foundation under Grants ECS-8352356 and ECS-8301521, and in part by General Motors Foundation, Inc., Detroit, MI.

Abstract

A multi-resolution modeling technique is described for coarse-to-fine model-based object recognition. Each two-dimensional object is modeled as a directed acyclic graph. Each node in the graph stores a boundary segment of the object model at a selected level of spatial resolution. The root node of the graph contains the coarsest resolution representation of the boundary of the complete object, leaf nodes contain sections of the boundary at the highest resolution, and intermediate nodes contain features at intermediate levels of resolution. Arcs are directed from boundary segments at one level of resolution to spatially-related boundary segments at finer levels of resolution. A generalized Hough transform is used to match the model nodes with regions in the corresponding level of resolution in a given input image pyramid. First, the root node of the model graph is matched with the coarsest level of the input image pyramid and an ordered list of hypothesized positions and orientations for the object is generated. These hypotheses limit the area in which the search for sub-objects (children nodes) should be conducted. If the sub-objects of one of the hypotheses are not found, the next best hypothesis for the location and orientation of the object at the coarsest level is tried. Advantages of this approach include the use of multi-resolution descriptions to model different parts of an object at different scales, the ability to detect partially occluded objects, the ability to control dynamically over the coarse-to-fine matching process, and the increase in recognition speed over conventional model-based recognition algorithms.

1. Introduction

Most of the techniques developed to date for object recognition use a set of image features computed at a single level of detail (resolution). Multi-resolution, or pyramid, image representations make image features explicit at multiple scales. Such representations are important for several reasons. First, different objects in a given scene often have different scales at which they are most readily apparent. For example, a surface sampled at too coarse a resolution may be too blurred or miss important surface structure, while a surface sampled at too fine a resolution may be noisy and contain fine details which confound the recognition of the dominant structure. Second, features which are large and extended at one resolution can often be more simply defined and detected as local features at a coarser level of resolution. Finally, by directing the search for objects based on coarse resolution results, we can speed up recognition time.

Relatively little work has attempted to use a pyramid as a multi-resolution representation for modeling objects, and to serve as the basis for *model-based object recognition*. (In contrast, many researchers have studied pyramid-based image processing and feature detection methods; see, for example, [1].) Several authors, including Kelly [2] and Yachida and Tsuji [3], have defined simple two-level procedures for implementing coarse-fine model descriptions. Crowley and Sanderson [4] used a symbolic representation of shape and a probabilistic matching algorithm for multiple resolutions.

In this paper we define a representation called the Hough pyramid, because it uses a pyramid representation combined with the Hough transform matching algorithm [5]. A hierarchical model of each two-dimensional object is constructed as a directed acyclic graph which contains at each node a selected segment of the object boundary at one of the pyramid levels of resolution. Each boundary segment is stored at a node in the form of a generalized Hough R-table [6] at a particular level of spatial resolution. Arcs are directed from boundary features at one level of resolution to spatially-related boundary features at finer levels of resolution.

The boundary segments of the entire object are stored at the root node at the coarsest level of resolution and are matched first with the corresponding resolution level of the input image. The results of this match are used to hypothesize a list of approximate positions and orientations of the object. Next, the higher resolution nodes in the model graph are matched to sub-windows of the larger, higher resolution images to verify the presence of finer boundary features and to determine more precisely their locations and orientations. The sub-windows are chosen according to the rough estimates provided by the coarse level match positions.

In the remainder of this paper we present the details of the Hough pyramid algorithm. Section 2 describes the detection of edges in the pyramid representation. Section 3 details the method of constructing the model graph. Section 4 presents the hierarchical Hough matching algorithm using the feature pyramid and model graph. Results are presented in Section 5. Concluding remarks are summarized in Section 6.

2. Feature Detection

One drawback of many object recognition algorithms is that objects are represented by connected "chains" of boundary pixels such as ribbons, skeletons, or concurves. These representations are intuitively appealing and may require less memory than the edge point image from which they were derived, but they are time consuming to generate because of the serial operations of linking and curve fitting. For example, the time to compute Perkins' concurve representation [7] is considerably greater than the time used for matching.

In order to avoid this serial image processing bottleneck, we represent objects as simple sets of unconnected edge points in an image window. The features stored in the model graph are vectors (stored in an R-table) corresponding to the edge points in selected windows of the zero-crossing (edge point) pyramid

The zero-crossing pyramid is created using the Laplacian pyramid [8] as follows. First, a Gaussian pyramid is created by convolving the original image with a two-dimensional mask approximating the Gaussian probability distribution function to create a blurred, low-pass filtered version of the original image. This blurred image is then sampled 2:1 to obtain the next coarser level of the Gaussian pyramid. This process is repeated until a 16 by 16 image is created as the apex of the pyramid. By subtracting two adjacent levels of the Gaussian pyramid (after appropriate expansion of the lower resolution image), the result is a bandpass filtered image similar to the original image convolved with a Laplacian operator. Performing this operation on all pairs of adjacent Gaussian pyramid levels results in the Laplacian pyramid.

The zero-crossing pyramid is created from the Laplacian pyramid by detecting at each level those pixels with positive Laplacian value which have at least one negative neighbor. Thus each zero-crossing image is a binary edge map of its corresponding Gaussian image. Because all steps can be performed as iterative, local operations, the zero-crossing pyramid can be created very efficiently.

Figure 1 shows an edge map of some keys and a washer. Figure 2 shows the zero-crossing pyramid constructed from the original grayscale version of Figure 1.

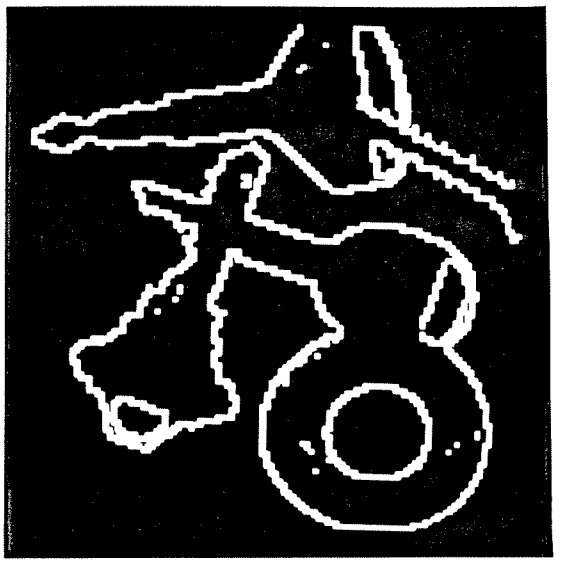


Figure 1 A scene containing three keys, a washer, and a screw.

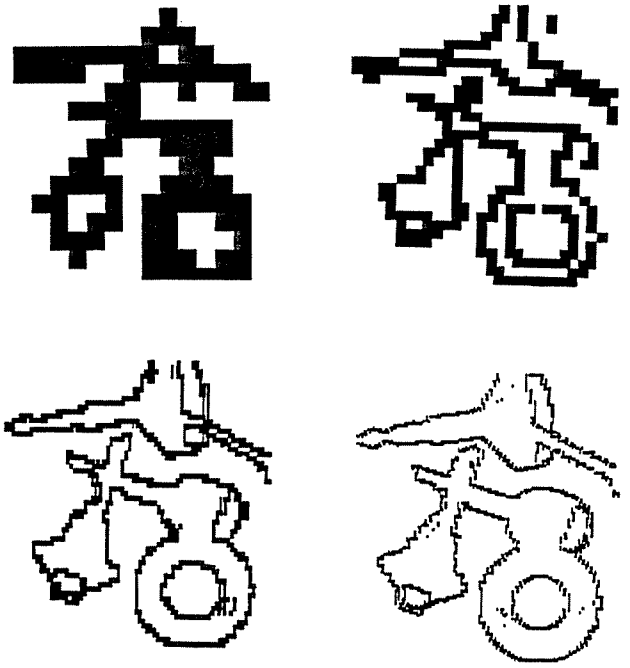


Figure 2 A four-level zero-crossing pyramid of Figure 1. Each level has been zoomed to a common size for display.

3. Model Graph Construction

Given a set of zero-crossing pyramids, one for each prototype object to be modeled, the goal of this phase is to construct a *models feature graph*, i.e., a graph of local features in which subgraphs specify individual object models. For simplicity of exposition, we assume there is only a single object being modeled in the remainder of this section. The merging of object graphs into a single composite models feature graph is straightforward.

Currently the user interactively selects at each level of the zero-crossing pyramid a set of windows containing features of interest for the object being modeled. From each window, a node in a directed acyclic graph is constructed. Each node stores the level of the pyramid from which the window was taken, a the edge points in the window, pointers to child nodes, if any, and a weight value used to indicate the significance of the feature contained in the window. The root node of the graph always contains the entire low resolution (16 by 16) edge map of the complete object. Each

node in the graph may have zero or more child nodes, but the feature stored in each child node is restricted to be contained in a sub-window of its parent's window. Thus, beginning at the root node and moving from parent to child, the feature stored in each node must be from a higher resolution image of the object than its parent node.

It is not necessary that a child node contain a feature from the next higher resolution image of the object, i.e., it is possible to skip levels of image resolution between adjacent nodes in the model graph. This is not recommended, however, for the following reason. The hierarchical Hough matching algorithm described in the next section uses the maximum peak in the accumulator array for the current node to estimate the location of the window where the child node is expected to be found. This estimate is based on the location of the centroid at the current node and is always integral, having an accuracy of plus or minus one-half pixel. Assume that the location of the peak of the accumulator array for the current node is the correct location (rounded to the nearest integer coordinates). Since the resolution of each pyramid level is twice that of the previous level, the location of the window where the child node should be found is only accurate to plus or minus one pixel. Should the peak location in the parent's accumulator array be slightly in error, say by one pixel, the location of the child's window will be off by two pixels, plus or minus one. In this case, our estimate will be off by up to three pixels. If the child node were associated with a window at resolution higher than twice its parent's resolution, then these errors are compounded even further. Large errors in window position will often cause the matching algorithm to fail because the error size becomes a significant percentage of the size of the window itself.

Each node in the graph contain the following fields:

R-table description of edge points in window
vector from window center to object centroid
pointers to child nodes
weight defining significance of current feature
resolution level of window

Each node stores the set of edge points in its associated window as an R-table [6], using the window center as the reference point of the R-table. To relate the location of the entire object to the sub-

object contained in the current node, a vector from the center of the window to the centroid of the object is also stored in the node. The centroid of the object is calculated from the edge points in the highest resolution zero-crossing image, with the centroid coordinates at each lower resolution level derived directly from this highest resolution level. Each node in the model has the same structure, and any node can be used as the root node of a subgraph if so desired.

The fine feature matching process uses the Hough peak at a parent node to locate the position and orientation of the sub-objects in each of its child nodes. To determine parent-child absolute spatial relationships, each node stores its position and orientation relative to the centroid of the entire object and the orientation of the object's major axis. For example, assume that the object we are modeling is a triangle. Say we want to store each corner as a node with the vertex of a corner as the reference point. First, every edge point associated with the corner is stored as a vector with its head at the reference point and its tail at the edge point. Next, a vector from the reference point to the triangle's centroid is stored in the node. Finally, all vectors are normalized by rotating them by $-\theta$ degrees, where θ is the angle of the major axis of the triangle.

Also associated with each node is a weight indicating the significance of this feature for the overall recognition of the object. Static weights are chosen by the user so that during the matching process if there exists any connected sub-graph of nodes with total weight greater than 1, then sufficient evidence exists to indicate the detection of the object. Thus each such match-set of nodes contains a sufficient number of features to uniquely distinguish the object from other possible objects. Specification of match-sets is important not only to speed the recognition process, but also to correctly detect objects which are partially occluded or noisy.

For example, if the object we are modeling is a room key, and the other objects of interest are nuts and washers, one of the key's match-sets need only contains the root node and a low resolution representation of the shaft of the key. However, if the other objects are also keys and are distinguishable only by the pattern of their teeth, then the match-sets need to contain enough high resolution detail of the teeth in order to discriminate this key from all other types. Currently, all

match-sets must include the root node due to restrictions in the matching algorithm. This condition could easily be removed, however, in order to deal with images in which occlusion prevents matching the root node with the coarsest resolution image.

The nodes in a match-set usually contain enough fine sub-features so that the high-resolution nodes can verify the matches of their low-resolution ancestors. Sometimes, however, match-sets include nodes that are not sub-features of their parent nodes. These nodes usually have two or more parents, and contain a sub-feature of one of the parents but not of the other, so that if one parent is occluded but the other is not, the fine node can still be reached through the unoccluded node.

3.1. An Example

In this section we present in detail the process of building the model graph for a room key. We will assume that four others objects are also to be recognized — two others kinds of keys, a washer, and a screw. The zero-crossing pyramid of the key is assumed to contain four levels of resolution: 16 by 16, 32 by 32, 64 by 64, and 128 by 128.

The first step in the modeling process is to compute the centroid of the key's edge points in the 128 by 128 image and the orientation of the major axis of these points. The corresponding centroid at each of the three lower resolution levels is then computed.

The root node of the key model graph is simply the 16 by 16 level of the zero-crossing pyramid. This is shown in Figure 3 as node 0. In each of the remaining three levels of the zero-crossing pyramid, the user defines nodes as the children of lower resolution nodes by choosing a rectangular window over the feature he wishes to include. Figure 4 shows two examples of an interactive modeling session in which the operator has defined nodes at the appropriate resolution levels of the pyramid displayed on the screen. In the current implementation, all nodes' windows are 16 by 16. All the points within the window are then stored in the R-table of the node, with the center of the window as the reference point.

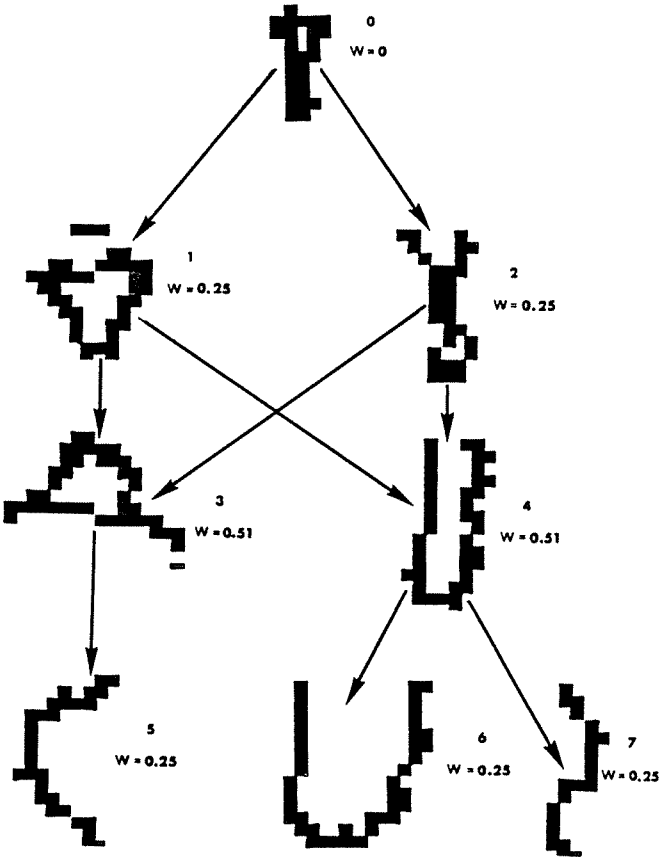


Figure 3 The model graph of a room key.

Figure 4 Interactive modeling. Node 1 of the 32 x 32 level and node 5 of the 128 x 128 level are shown. The nodes are chosen interactively by the operator by windowing the areas of interest using 16 x 16 windows.

For example, after the root node is stored, the 32 by 32 zero-crossing image is displayed and the user selects windows which become children of the root. Figure 3 shows two nodes which have been selected, corresponding to the head and shaft of the key represented at the root. In general, it is important to choose higher level nodes which quickly disprove false matches. In this case, however, the 32 by 32 level is still too coarse to distinguish between the given key and the other two key types, although it does contain enough detail to distinguish between the key and the washer and the screw.

After the user is finished with the 32 by 32 level, the 64 by 64 zero-crossing image is displayed. All previously stored nodes, starting with the root node, are displayed, giving the user the opportunity to create and link a new child node to any node at the levels above the current level. Figure 3 shows two nodes containing windows at the 64 by 64 level. Node 3 corresponds to the hole at the top of the head of the key and node 4 is a finer resolution view of the shaft. Note that in this case both new nodes have been linked from both nodes at the 32 by 32 level. This was done for the reason mentioned above, namely, to define match-sets which skip certain coarse resolution nodes (i.e., the nodes at the 32 by 32 level) in case occlusion or noise prevents a successful match at that level.

The same procedure is repeated at the 128 by 128 level. In Figure 3, nodes 5, 6 and 7 correspond to this level, defining boundary segments for the side of key head, the tip of the shaft, and a section of teeth, respectively.

Finally, weights are assigned to each node in the graph as shown in Figure 3. The root node has been assigned a weight of 0 because we assume in the current implementation that it must be part of every match-set and must always be successfully matched in order to recognize the key. This restriction is not crucial, however, and can easily be removed with appropriate modifications in the matching algorithm. With this assignment of weights there are eight match-sets implicitly defined. Figure 5 shows each of these sub-graphs.

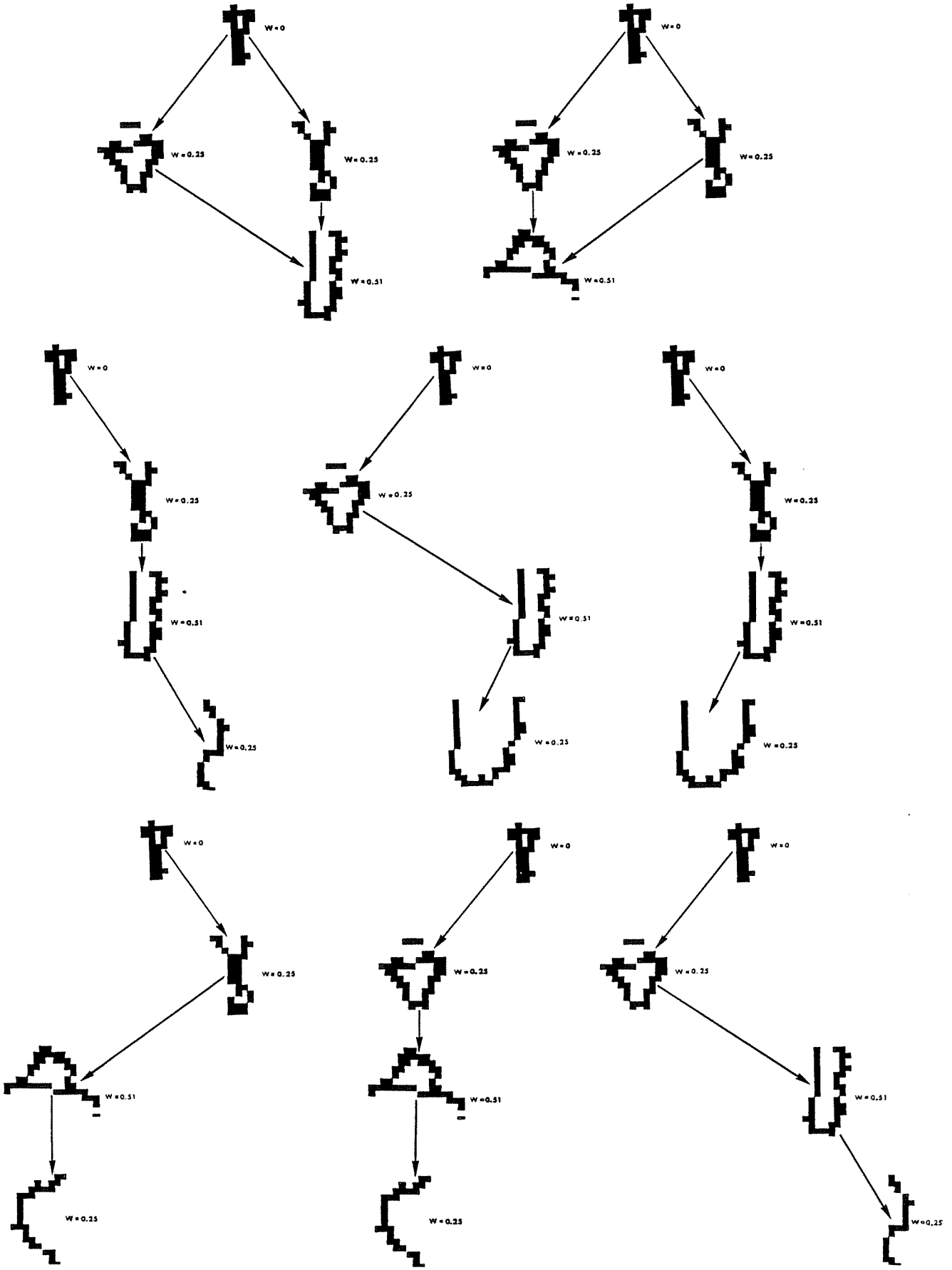


Figure 5 The set of match-sets for the room key. One of the eight match-sets has to be successfully match for recognition.

4. Hierarchical Hough Matching

Our matching algorithm employs a top-down, coarse-to-fine search strategy to recognize objects in a given input image. We consider here the particular problem of detecting one instance of a given object specified by its model graph. Our method may be briefly summarized as follows. The initial best estimate of the location and orientation of the object is determined from information at the coarsest level of resolution, and this estimate controls the search for higher resolution features. If enough higher resolution features are found to uniquely identify the object, the matching succeeds. If enough higher resolution features are not found, the previous estimate is discarded and the next best estimate at the coarsest level is chosen and used to control the search for sufficient confirming higher resolution features. This process continues until the all the nodes in some match-set are matched (success) or until no more coarse level estimates remain (failure). In the remainder of this section we present the details of this matching procedure.

4.1. Coarse Level Hypothesis Formation

The first step in the matching process is to match the model graph's root node with the coarsest resolution level (16 by 16) of the input image's zero-crossing pyramid. The generalized Hough transform is used for this purpose [6], resulting in a three-dimensional accumulator array of size 16 by 16 by 16 of match values. (Thus we estimate orientation at this level in increments of 360/16 = 22.5 degrees.)

The peaks in the accumulator array indicate the best estimates of the location and orientation of the object. Non-maximum suppression is performed on the accumulator array using a 3 by 3 by 3 neighborhood to aid in peak detection. Next, all peaks over a pre-determined threshold are sorted in descending order. Each peak, beginning with the highest, is now hypothesized to be the actual position and orientation of the object in the image.

4.2. Fine-Feature Verification Matching

Given a peak from the coarse matching process, the fine-feature matching process calculates the position and orientation of the children of the root node. For each child of the root, a 16 by 16 window centered on its estimated position is extracted from its level in the input image's zero-crossing pyramid. Another generalized Hough transform is now applied using this window and the R-table associated with the current model node. The resultant accumulator array in this case is 16 by 16 by 11. That is, orientation is now quantized to 11 values at multiples of 4.5 degrees centered around its parent's estimated orientation angle (which was quantized in steps of size 22.5 degrees).

The time required by the generalized Hough transform is proportional to the number of edge points in the window times the size of the node's R-table times the number of orientations. Consequently, our use of small 16 by 16 windows and small R-tables saves considerable time over the brute-force method of trying all possible positions and orientations at the current level.

Following the completion of the Hough transform, if the maximum value in the accumulator array divided by the size of the node's R-table is greater than a predefined threshold, then the window matches the current node. Otherwise, the match fails. If the match succeeds, then the weight associated with the current node is added to the current match-set total. If that total now is greater than 1, then the object has been detected; otherwise, the same procedure is used recursively to try and match all children of the current node (using the updated position and orientation associated with the maximum peak in the current node's accumulator array). It should be noted that the match threshold used here can be chosen fairly conservatively because "false positives" at one node are quickly disproved by finer resolution children nodes.

In general, the coarse resolution nodes help to rapidly disprove false hypotheses and to make fine adjustments in the hypothesized position of the object's true position and orientation. There is not usually enough information in the coarser nodes to verify the presence of the object in the image, and coarser nodes may match similar features of the wrong object. A coarse representation of the shaft of a key, for example, may not match well with a washer in the image, thus quickly disproving

this hypothesis, but it may match well with the shaft of a bolt, and so cannot be used to verify the presence of the key in the image. If an hypothesis is correct, the coarse nodes reduce the error in the object's centroid and orientation estimates.

The fine resolution nodes generally serve to verify an hypothesis. Fine nodes rarely give false matches; in order for a fine node to match at all, the feature must be small (relative to the size of the image) and at roughly the correct orientation. If the features in the fine nodes are chosen judiciously, this is unlikely to happen except with the correct hypothesis. As a result of these observations, the weights associated with nodes in a model graph should be chosen so that match-sets include mostly coarse and intermediate level nodes plus a few fine nodes to complete the verification of at least selected parts of the object.

In our prototype system, we have used a breadth-first strategy to search the model-graph, but any graph search algorithm can be used. If a node has been correctly matched, its children are placed at the end of a queue and the next node in the queue is tested. Child nodes inherit the centroid and orientation estimates of their parents, appropriately scaled to the resolution level of the child. When all the nodes in a match-set are correctly matched, their estimates of the centroid and orientation are compared to make sure they agree to within a predefined error tolerance. If they agree, the object is successfully detected.

If no match-set is successfully matched, the current hypothesis fails and the next highest peak from the root node's list becomes the new hypothesis. Hypotheses that have failed are never retried. This sequence of trying and discarding hypotheses continues until one hypothesis results in the matching of a match-set, or until there are no more hypotheses.

4.3. Other Strategies

4.3.1. Failure Recovery

As our system is currently implemented, there is no recovery from failure. That is, if the matching process fails to find a match-set for any of the hypotheses, the system reports failure and

quits. Alternatively, the system could keep track of all nodes successfully matched by each hypothesis. The hypothesis that correctly matches nodes deepest in the graph (at the finest resolution nodes) would be considered the best guess. This modification would be useful when an object is so occluded that no match-set succeeds but enough fine features are detected to indicate that the object is probably present.

There is nothing inherent in structure of the root node that distinguishes it from any other node. Therefore we could also use each of the root node's children as new root nodes, and try to match each to the complete image at the appropriate resolution. The peaks from these matches would then be used as before as hypotheses which are confirmed or denied by finer resolution nodes.

4.3.2. Speed-ups

In the current implementation, a node is either correctly matched or it is not. That is, if the match measure exceeds a predefined threshold, then it matches. Alternatively, it may be advantageous to make match-sets dependent upon the combined measure of match (i.e. Hough peak size) of some or all its member nodes. If the match measure of a coarse-level node is only slightly above the threshold, it may not indicate that the feature is necessarily present. However, if it is significantly greater than the threshold, it should increase the likelihood that the hypothesis is correct. In this case, it may not be necessary to match nodes that are sub-features of this node, and the match-set could be modified dynamically to take this into account. This dynamic pruning of the match-set could save considerable time because fewer nodes would need to be matched in order to verify the presence of simple, unoccluded objects.

A second speed-up could be obtained by modifying the matching algorithm used at non-root nodes. At each of these nodes an hypothesized position and orientation of match is already known from its parent's match. This information can be used to minimize the number of points needed in the accumulator array of the current node to confirm the location and orientation of the match. Since the number of candidate points is relatively small, we could in fact replace the Hough transform algorithm with a simple template matching procedure at each of the candidate positions and

orientations.

Of course, this increase in speed is at the cost of increased errors in the estimated position and orientation computed at each node. If the actual position of the object is greater than the estimated worst case distance, the match may either fail, or succeed but estimate inaccurately the feature's position and orientation. This modification is most appropriate for fine-level nodes, where the assumption is that errors in the hypothesis have been corrected by ancestor nodes. It is probably not appropriate for coarse-level nodes, where location errors must be corrected.

5. Results

The following table shows the results of the procedure on three test images containing keys and washers. Image 1 is shown in Figure 6. Image 2 is shown in Figure 7, and Image 3 is shown in Figure 1. The model graph shown in Figure 3 was used to recognize the room key in each image (the lower-left key in Figure 1). The total size of all R-tables in the graph is 295. With both Image 1 and Image 2, the first hypothesis selected was the correct one, and the matching process examined five nodes, with one failure, to find the key. The failure node in both cases contained a feature which was occluded. In the more complicated image, Image 3, the system had to try ten different hypotheses before finding the correct one. The strongest hypothesis had a match measure of 0.84 while the correct hypothesis had a match measure of 0.77. For Image 3, 28 nodes were tested before a match-set was completely matched.

Image	# hypotheses tested	# nodes tested	Avg. # nodes to disprove a false hypothesis
1	1	5	0
2	1	5	0
3	10	28	3.1

To roughly estimate the speed-up of this method over a single-level generalized Hough approach, consider Image 3. There were about 100 edge points in every window selected from Image 3's pyramid. Thus the total computation time for this example was proportional to $295 * 11 * 100 \approx 3 * 10^5$. In contrast, using the generalized Hough transform with the finest resolution image (128 by 128) and a fine resolution R-table for the key, the time would be proportional to 1400 (the number of edge points in the image) times 312 (the size of the R-table) times 80 (= 360/4.5 possible orientations), or about $3 * 10^7$.

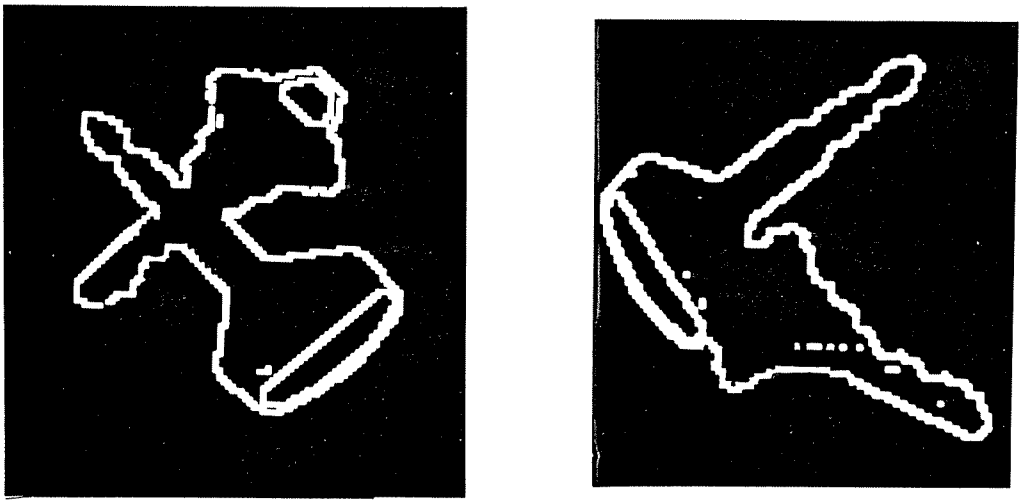


Figure 6 and 7 Two test images

6. Concluding Remarks

A model-based object recognition procedure has been presented which models 2-D objects in terms of parts, each part represented at its most appropriate level of resolution. The top-down matching algorithm using this model is shown to effectively identify objects which are substantially occluded in a scene and are represented by noisy edge point data in an image.

In conclusion, the advantages of using a Hough pyramid for model-based object recognition include the following:

- *Hierarchical object modeling.* In general, no one level of resolution is best for describing all aspects of an object to be modeled and recognized. By computing multiple scales of description in the pyramid, we can select the most appropriate level for representing each model feature. For example, a model of a "key" may describe the "shaft", "head", "hole" and "teeth". The shaft and head may best be described at a relatively coarse level in which the shaft is described as a rectangle and the head as a circle. The hole may be appropriately defined at an intermediate level of resolution and the teeth at a fine level. Trying to create a model which

includes each of these features at a single level of resolution would not only be less natural, but would also require more space since the finest resolution would be used for describing all features.

- *Increased control.* The graph model structure enables the algorithm to locate partially occluded objects by verifying the presence of only a sufficient subgraph of distinguishing features.

- *Coarse-to-fine matching speeds recognition.* Matching can be performed by first comparing coarse features to quickly determine candidate models and their locations; successively finer features can then be used to verify the model. Using a Hough approach at the finest level of resolution, the time required is $O(nro)$, where n is the number of edge points, r is the size of the Hough R-table, and o is the number of possible orientations. Using a pyramid, we can drastically reduce at each level the size of n and o by using only a small sub-window of the image and a restricted number of candidate orientations based on the predicted orientation obtained at a coarser resolution. We can also restrict the size of r by matching large features at coarse resolutions and small features at fine resolutions. Because n, r, and o are small compared to what they would be if a single template of the entire object boundary was matched at the highest resolution and all possible angles, the total time is drastically reduced. For example, given a 128 by 128 image with $r = 350$, $n = 700$, and $o = 80$, the single level Hough algorithm requires time proportional to 10^7. Using a Hough pyramid model graph containing 8 nodes, with $r = 50$, $n = 60$, and $o = 16$, the total time required is proportional to 10^4.

- *Common features are stored uniquely.* Common intra- and inter-model features are stored uniquely in a single node of the composite graph describing all of the model objects. This provides a concise description of multiple models when features are common (e.g. holes and corners, especially at coarse resolutions).

- *The Hough transform is a robust matching algorithm.* One of the advantages of the Hough transform compared to other pattern matching algorithms is that it is relatively insensitive to noisy or broken contours. Edge points can be missing and it will still give a good match as

long as most of the points are present.

References

1. S. L. Tanimoto and A. Klinger, eds., *Structured Computer Vision*, Academic Press, New York, 1980.

2. M. D. Kelly, Edge detection in pictures by computer using planning, in *Machine Intelligence 6*, Elsevier, New York, 1971, 397-409.

3. M. Yachida and S. Tsuji, A versatile machine vision system for complex industrial parts, *IEEE Trans. Computers* **26**, 1977, 882-894.

4. J. L. Crowley and A. C. Sanderson, Multiple resolution representation and probabilistic matching of 2-D gray scale shape, *Proc. Workshop on Computer Vision: Representation and Control*, Annapolis, MD, 1984, 95-105.

5. R. O. Duda and P. E. Hart, Use of the Hough transform to detect lines and curves in pictures, *Graphics and Image Processing* **15**, 1972, 11-15.

6. D. H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, *Pattern Recognition* **13**, 1981, 111-122.

7. W. A. Perkins, A model based vision system for industrial parts, *IEEE Trans. on Computers* **27**, 1978, 126-143.

8. P. J. Burt and E. H. Adelson, The Laplacian pyramid as a compact image code, *IEEE Trans. Communications* **31**, 1983, 532-540.