SHAPE REPRESENTATION AND RECOGNITION

Larry S. Davis
Computer Sciences Department
University of Texas at Austin
Austin, Tx. 78712

TR-100 June, 1979

This is a draft of a chapter to appear in Consistent Labeling Problems in Pattern Recognition, ed. R. Haralick, Plenum Press.

This research was supported in part by the National Science Foundation under Grant ENG 74-04986.

This report was reproduced with funds provided by the Hitachi Corporation.

1. Introduction

This chapter is concerned with structural methods for the representation and recognition of planar shapes. In particular, we will consider relational network models and syntactic models for shape representation. In general, when shapes are modeled by relational networks, then graph matching procedures are used for recognition. A graph matching procedure will be described which includes a relaxation component, and its application to recognizing pieces of shapes (which correspond to coastlines of islands of the world) will be presented. When shapes are modeled using grammars, then a parsing procedure is used to recognize an unknown shape. Syntactic models can be regarded as a generalization of relational networks which supports a hierarchical organization of relational models. A parsing procedure will be presented, called a hierarchical relaxation process, which recognizes shapes using syntactic models. Its application to the recognition of shapes of airplanes will be discussed, and it will be compared to other shape parsing procedures.

This chapter will not discuss problems related to recognizing a three-dimensional object based on the shapes of one or more two-dimensional projections of that object. The interested reader is referred to, e.g., Marr and Nishihara [1] for a discussion of this problem. We will also not consider the vast variety of shape models based on collections of scalar shape features (for example, Fourier features, central moments, etc.). The reader is referred to Pavlidis [2] for a discussion of such models and to Duda and Hart [3] for an introduction to the statistical

recognition procedures for these models.

Although a considerable amount of effort has been directed towards the problem of planar shape recognition, it is nevertheless true that no general solution to the problem has been discovered. Some of the factors to which this failure can be attributed are:

1) orientation -- One needs to be able to recognize a shape independent of its orientation. For example, in chromosome recognition, there is no way to control the orientation of the chromosomes on the slide.

2) size -- One also needs to recognize a shape over a wide range of sizes. For example, in airplane recognition based on aerial reconnaissance imagery, the size of the airplane in the image depends on several factors including the characteristics of the sensor, the altitude of the surveillance aircraft and the computational resources available for recognition (which determine the spatial resolution of the digital image). Furthermore, the appearance of objects changes with scale; the shape model should, ideally, account for those changes.

3) distortion -- It is quite possible that parts of the shape, or perhaps the entire shape, may appear distorted. "Distortion" may be due to the fact that the object is viewed from an oblique angle (as in aerial reconnaissance where we view this problem as two-dimensional since the airplanes are on the ground and so give rise to a rather limited variety of two-dimensional projections which we intuitively regard as distortions of some prototype shape), or to imperfections in the image segmentation procedures which attempt to recover the shape for subsequent recognition.

4) partial information -- Often, one must recognize a shape based on only partial information, i.e., an entire shape is not available for analysis possibly because of occlusion in the image or deficiencies in the segmentation procedures. So, for example, the camouflage of an airplane may preclude our segmenting the entire airplane outline from an image. We would hope, nevertheless, to be able to recognize the airplane based on the available information.

5) agglomeration -- A complementary problem to partial information is when an agglomeration of several shapes is presented to the recognition algorithm. This is a particularly difficult problem to deal with since so many subpieces of the agglomeration might correspond to individual objects. One is faced with such problems in chromosome analysis, where overlapping chromosomes are often extracted as one shape by the segmentation procedures.

It will be important to keep these factors in mind during the discussion of network and syntactic shape models in order to appreciate the strengths and weaknesses of the various approaches.

2. Relational Network Shape Models

A relational network model for shape representation is composed of two parts:

1) a model for decomposing the shape into pieces, and

2) a model for computing salient relations between those pieces.

So, for example, we might attempt to decompose a shape into a central piece and lobes, or into convex subsets (see Pavlidis [4]). The relations might include adjacency of pieces, collinearity of pieces (if we associate an axis with each piece -- say in the direction of elongation -- then we can compute the collinearity of axes), parallelness of pieces, etc. (See also the discussion in Section 4.1.)

If the shape is not closed, then an areal decomposition is quite hard to obtain because of difficulties associated with determining a closure for the available pieces. In such situations, one might consider decomposing the available perimeter into simple pieces such as straight lines, circular and elliptical arcs, etc. Methods of piecewise approximation of shapes are discussed in detail in Pavlidis [5]. We will present a simple example involving angle detection.

Suppose that $\{(x_i,y_i)\}_{i=1}^{n}$ is a vector representation of the boundary points of a shape, S. At each point in S we compute a number related to the curvature of S at that point:

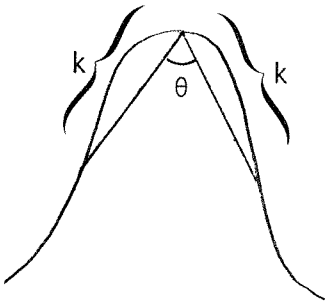
$$C_k(i) = (\vec{a}_{i_k} \cdot \vec{b}_{i_k})/|\vec{a}_{i_k} \cdot \vec{b}_{i_k}|$$

where

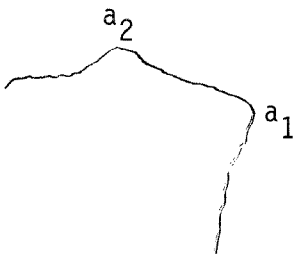
$$\vec{a}_{i_k} = (x_i - x_{i+k},\ y_i - y_{i+k})$$

$$\vec{b}_{i_k} = (x_i - x_{i-k},\ y_i - y_{i-k})$$

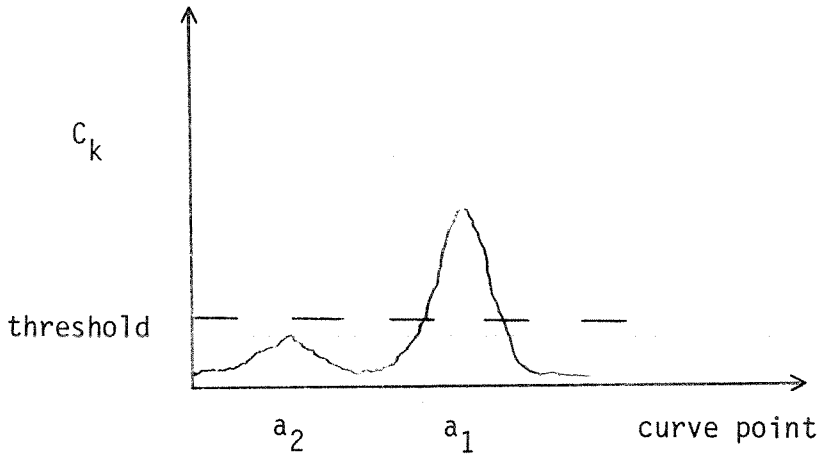
Points whose C_k values are (1) local maxima and (2) above threshold are marked as angles (see Figure 1). The pieces of the shapes are then taken to be the angles along with the segments of shape between the angles (possibly approximated by straight lines). It is precisely this representation which we shall use in our shape matching procedure based on relational networks.



a) $C_k = \cos^{-1}(\theta)$



b) shape with one sharp angle (a_1) and one shallow angle (a_2)



c) both $C_k(a_1)$ and $C_k(a_2)$ are local maxima, but $C_k(a_2) <$ threshold, so it is not marked as an angle.

Figure 1. Angle detection base on C_k.

2.1 Relation Network Matching Using Relaxation

In this section we present a shape matching procedure based on the representation of a shape as a network of angles (and sides). This procedure is invariant to rotation and a wide range of scale changes; it will construct and manipulate a structure called an association graph. An important property of this approach is its ability to deal with incomplete information (point 4 in the introduction), sometimes called the segment matching problem in shape recognition (see Freeman [6]). Shapiro [7] describes a shape matching procedure similar to the one described below.

Let $O = O_1, O_2, \ldots, O_n$ be the polygonal path representation of some object curve in a data base of curves. Each O_i is a triple, $\langle X_i, Y_i, MAG_i \rangle$, where X and Y are the coordinates of O_i, and MAG_i is the magnitude of the angle. Similarly, let $T = T_1, T_2, \ldots, T_m$, $m < n$, be the polygonal representation of an unknown curve segment that we wish to match against O. Informally, a match of T to O has two components:

1) a function providing an association of elements in T with object elements, with proper regard to the sequencing of the elements (called an association function);

2) a global coordinate transformation from the coordinates of T to the coordinates of O that provides a spatial registration of T with O.

The matching process to be described is a simplification of the relaxation scheme used in Davis and Rosenfeld [8]. There, the goal was to recognize noisy upright squares on a noisy background. The template for the square was made of four subtemplates -- one for each corner of the square. A node was entered into a network corresponding to each

picture point at which some local evaluation function designed to detect corners detected a match with one of the four corners. Each node in the network had a fixed label set associated with it. These labels included four labels representing the interpretation of that node as one of the four corners of the square and one label corresponding to that node being a false alarm. Each label was assigned an estimate of the probability of it being the correct label for that node based on the results of the local evaluation function. Pairs of nodes that corresponded to image points that might form part of a single square were connected in the network. The probability of a particular node having a particular label was then iteratively updated on the basis of the probabilities of labels at neighboring nodes in the network. Each iteration of this updating occurred in parallel at every node in the network.

The process has been simplified in the following way. The local evaluation function will determine, for each angle in T, a subset of object angles that the angle in T may be associated with. Each association of an angle in T with a single matching object angle will correspond to a node in a graph, called the association graph. Thus, each node in the graph corresponds to an association of an angle in T with a single angle in O, rather than with all of the angles in O (i.e., all of the corners of the square). Each node will be assigned a weight reflecting how well the object angle and the angle in T match according to some local evaluation function.

Connections between nodes are determined based on our relation models. Denote a node in the graph by $\langle O_i, T_j \rangle$. Suppose $\langle O_i, T_j \rangle$, $\langle O_{i'}, T_{j'} \rangle$ are two such nodes, and suppose that prior knowledge is available concerning the

scale of $\mathcal{O}$ relative to T. Then the two nodes can be connected if $d(\mathcal{O}_i,\mathcal{O}_{i'})/d(T_j,T_{j'})$ is consistent with our prior knowledge of scale (here $d(\mathcal{O}_i,\mathcal{O}_{i'})$ is the distance between $\mathcal{O}_i$ and $\mathcal{O}_{i'}$).

Nodes are pruned from the graph if some evaluation function applied to the neighborhood of the node is below threshold. For example, consider the following evaluation function. Let G = <N,V> be the association graph with node set N and arc set V. Suppose M(n), for $n \in N$, is the weight, or merit, of node n. Then the function

$$E(n) = \sum_{\substack{n' \text{ with} \\ (n,n') \in V}} M(n')$$

simply sums the weights of all the neighbors of node n. The node deletion process using a particular threshold t on E(n) will define a sequence of association graphs $G = G_0,G_1,\ldots,G_s$, where G_i is obtained by G_{i-1} as follows:

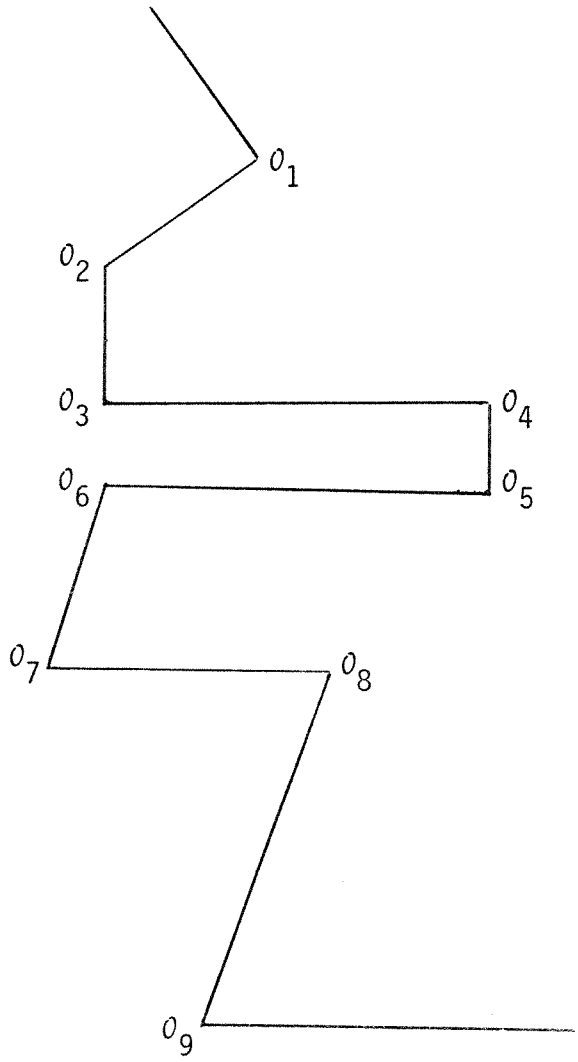
1) $N_i = N_{i-1} - \{n: \ n \in N_i \text{ and } E(n) < t\}$

2) $V_i = \{(n,n') | n,n' \in N_i\} \cap V_{i-1}$

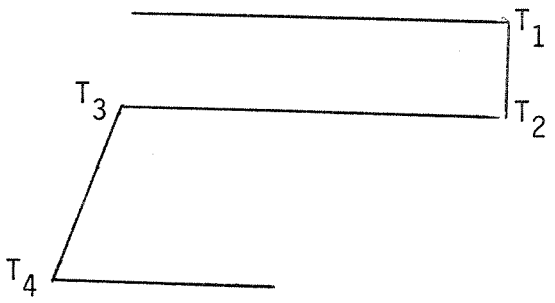
When $G_i = G_{i-1}$, the process terminates. We will let G^∞ denote the fixed point of this process. Figure 2 shows a simple example of this process.

Proposition: For some finite s, $G_s = G^\infty$.

Proof: Immediate, since G_0 is finite, and each iteration which does not reach the fixed point reduces the size of the network (and the empty network is a fixed point).



a) a shape from a data base.

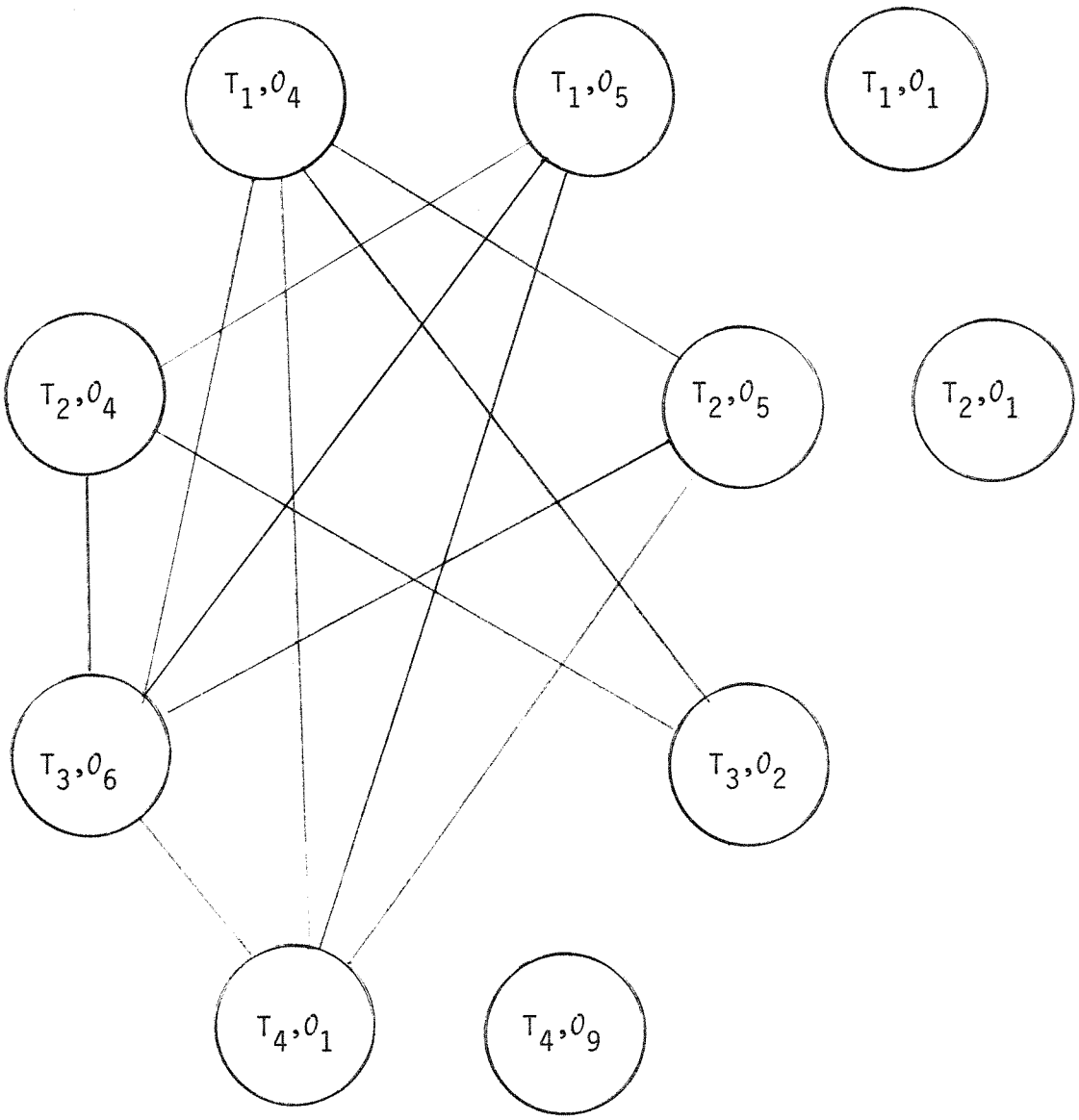


b) an unknown shape.

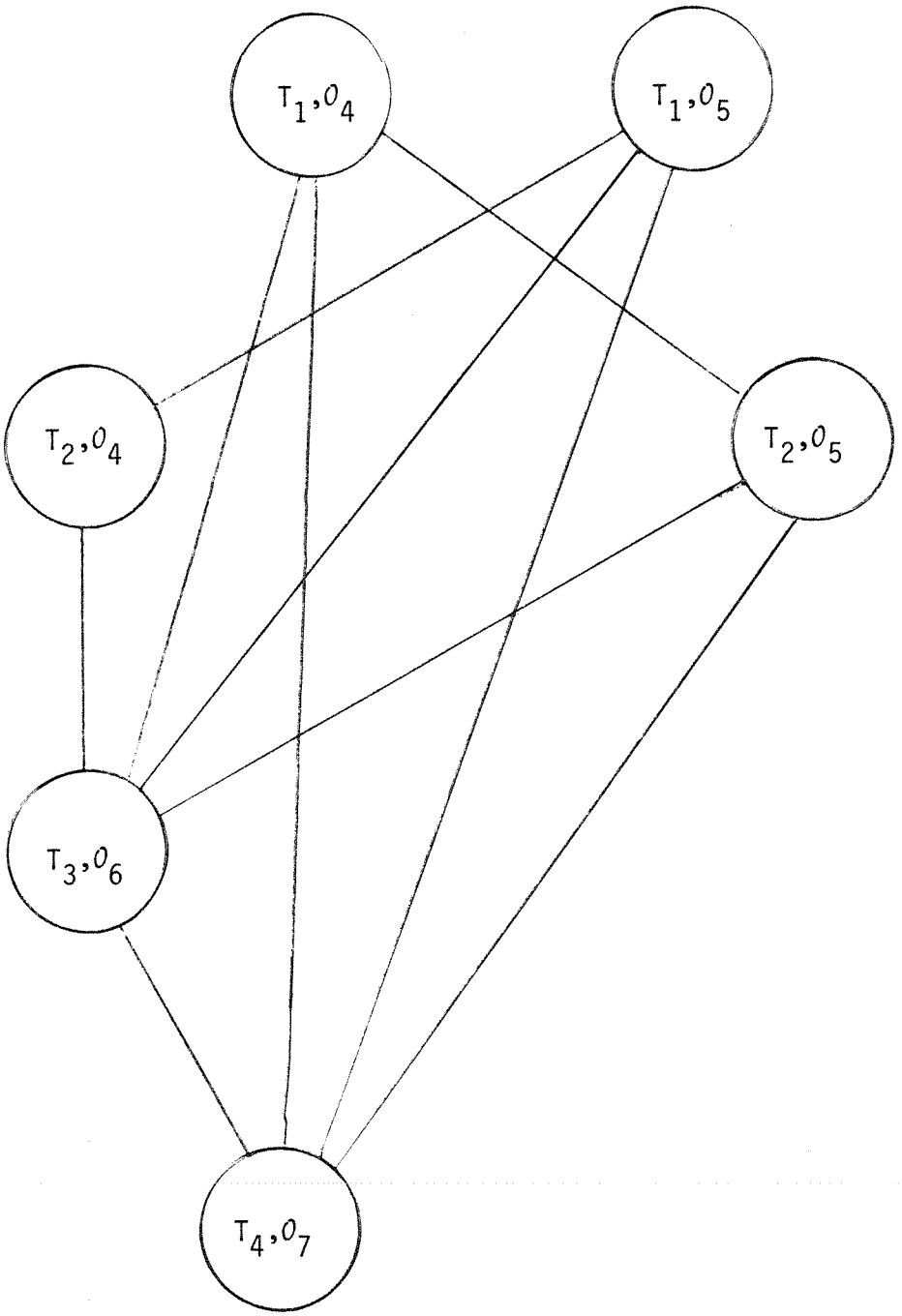
T_i	Matching angles in O (exact match)
T_1	O_1, O_4, O_5
T_2	O_1, O_4, O_5
T_3	O_2, O_6
T_4	O_7, O_9

c) angle match table

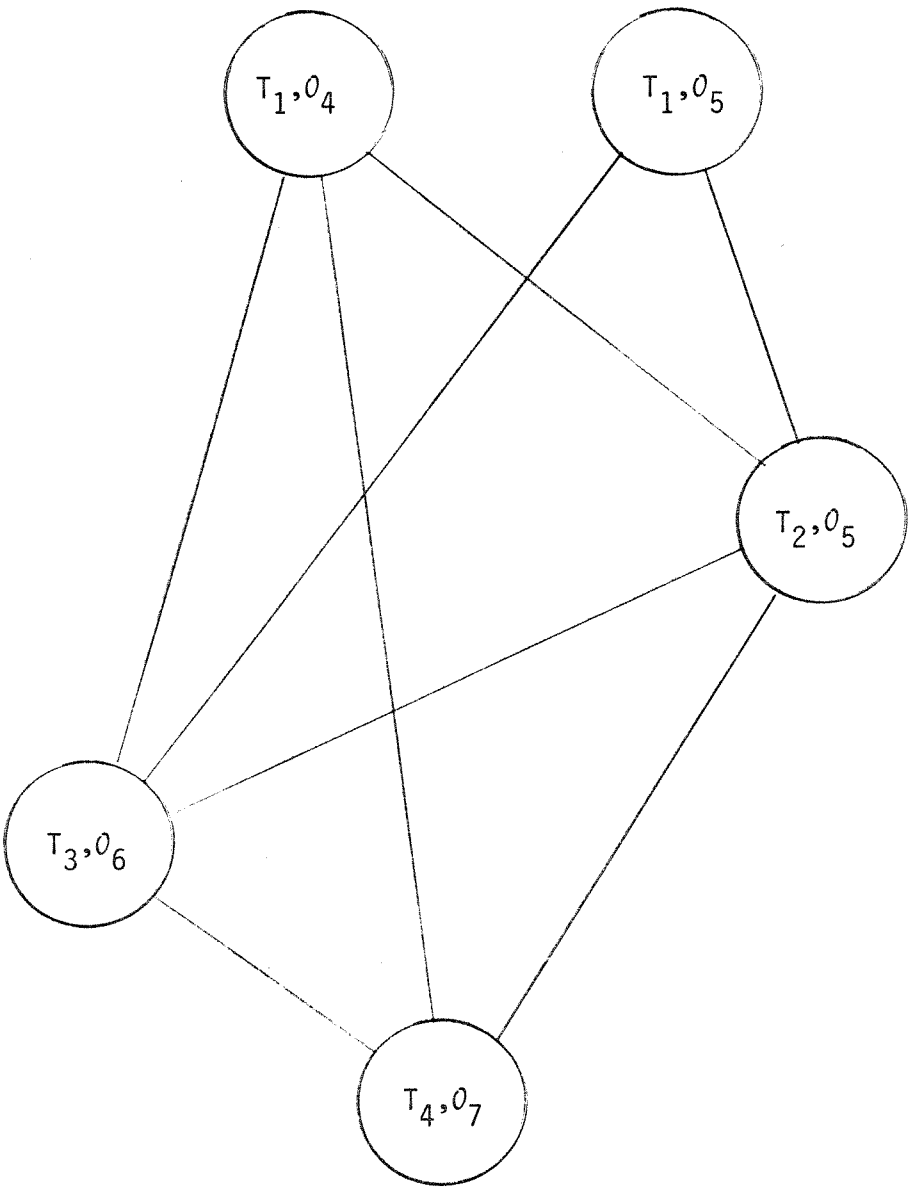
Figure 2. Example of node deletion process.



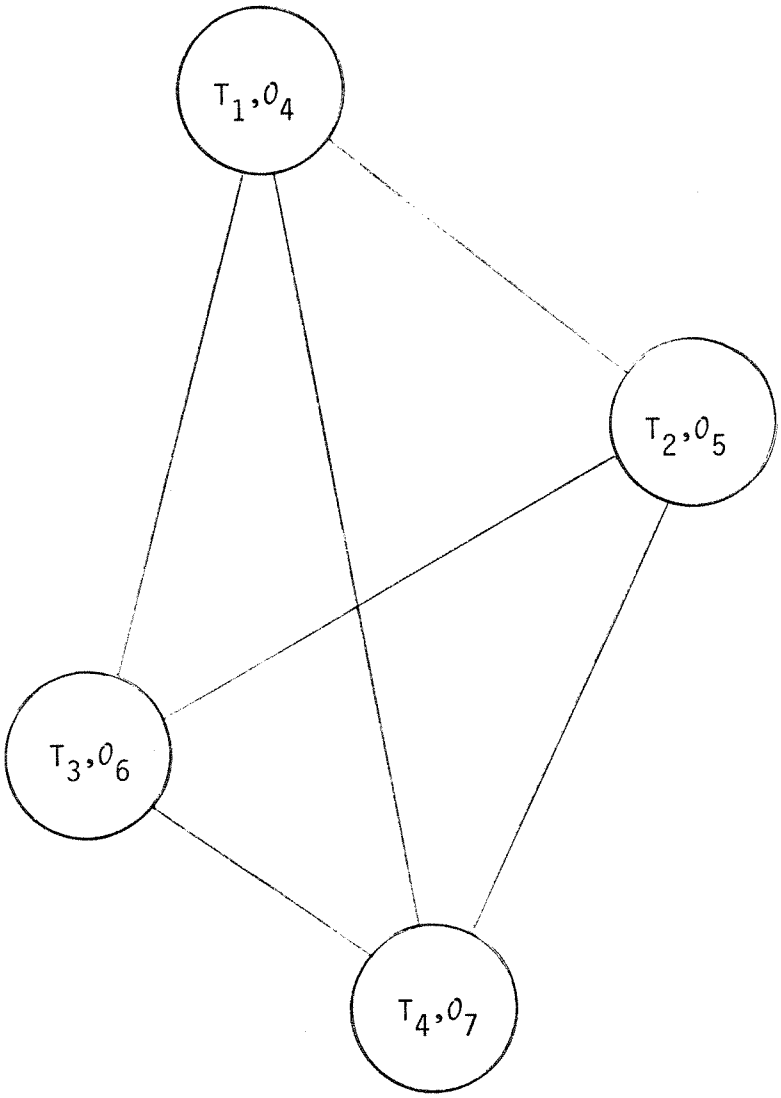
d) Association graph to which we will apply the node deletion criteria: "Delete all nodes of degree < 3."



e) Result of applying one iteration of node deletion to Figure 2d.



f) Result of applying one iteration of node deletion to Figure 2e.



g) Result of applying one iteration of node deletion to Figure 2f. The resulting network is now stable.

Thus we have a "discrete" relaxation operation. It is discrete because it is a label discarding process (deleting a node is really eliminating one possible interpretation of an unknown angle). On the other hand, it does not require that a node be absolutely consistent with all of its neighbors, but only that it be adequately consistent with enough of its neighbors.

We will now attach information to the edges of the graph, and introduce an edge filtering process which will further enhance the results of the local evaluation function based on local similarity transform equivalence between pieces of the unknown and of the object.

An edge in the association graph connects two associations $<O_i,T_j>$ and $<O_{i'},T_{j'}>$. The similarity transformation that maps the line segment joining T_j to $T_{j'}$ onto the line segment joining O_i to $O_{i'}$ is unique, and corresponds to a local hypothesis of the best coordinate transformation that minimizes the mean squared distance between T and O given that $<O_i,T_j>$ and $<O_{i'},T_{j'}>$ are part of the best overall association. The parameters of this transformation will be associated with the edge joining $<O_i,T_j>$ and $<O_{i'},T_{j'}>$. If these associations are in fact part of the best association function from T to O, then many other edges in the network should be labeled with similar parameters. In particular, these transformations should be found attached to the other edges that are incident on $<O_i,T_j>$ and $<O_{i'},T_{j'}>$.

This suggests that the following new graph (the line graph), G', be built from the given association graph, G.

Definition: Let $G = \langle N,V \rangle$ be an undirected graph with node set N and edge set $V \subseteq N \times N$. Then the line graph, G', of G is the graph (N',V') where

1) for each $v \in V$ we have a node $n' \in N'$;

2) $(n'_1,n'_2) \in V'$ iff n'_1,n'_2 correspond to edges in V that have one common endpoint.

Then, a node can be deleted by a parallel filtering algorithm if, say, the value of some evaluation function applied to that node is less than some threshold. This can be done in the following way. Let $n'_1,n'_2 \in N'$. Suppose n'_1 corresponds to the edge joining $\langle O_{i1},T_{j1} \rangle$ to $\langle O_{i2},T_{j2} \rangle$ in the original association graph, G. Then the similarity transformation labeling n'_1 is the 4-tuple (a,b,c,d) where

$$\sum_{r=1}^{2} \left[X_{ir} - (aX_{jr} + bY_{jr} + c)\right]^2 + \left[Y_{ir} - (aY_{jr} + bX_{jr} + d)\right]^2 = 0$$

Then node n'_2 must correspond to an edge in G connecting some $\langle O_{i3},T_{j3} \rangle$ to either $\langle O_{i1},T_{j1} \rangle$ or $\langle O_{i2},T_{j2} \rangle$. Suppose it is $\langle O_{i1},T_{i1} \rangle$. If both n'_1 and n'_2 were part of the same perfect match of T to O, then it would be the case that

$$s(n'_2|n'_1) = \sum_{r=1,3} \left[X_{ir} - (aX_{jr} + bY_{jr} + c)\right]^2 + \left[Y_{ir} - (aY_{jr} + bX_{jr} + d)\right]^2 = 0$$

If the match were not perfect, or if n'_1 and n'_2 were not part of a single match, then, in general, $s(n'_2|n'_1) > 0$. We can see that $s(n'_2|n'_1)$ measures the support of the pair of associations corresponding to n'_2 for the pair of associations corresponding to n'_1.

We can use the evaluation function

$$E'(n') = \sum_{n''} s(n''|n')$$

with $(n',n'') \in V'$

to create a sequence of graphs $G' = G'_0, G'_1, \ldots, G'_s$, where G'_1 is related to G'_{i-1} by

1) $N'_i = N'_{i-1} - \{n' \in N'_{i-1} | E'(n') < t\}$;

2) $V'_i = \{(n',n'') | n',n'' \in N'_i\} \cap V'_{i-1}$.

By introducing the line graph, the same system of programs that performed node deletion can now be applied to the edge deletion process. Also note that if G' is the fixed point of this process, then it is reached after a finite number of steps.

Deleting a node from the line graph is equivalent to erasing an edge of the original graph. If all of the edges of a node in the original graph are erased in this way, then that node can be deleted. This process can disconnect the association graph, and this has important consequences for the search processes described in the next section.

If the association graph is examined after both filtering operations have been applied to it, it will usually be the case that it contains at most a few connected components of nodes with highly consistent similarity transformations attached to the edges of these components. Searching for a best association function should almost always be trivial. We have also constructed powerful clues as to the best global similarity transformation between T and *O*.

2.2 Searching the Pruned Association Graph

This section discusses methods for extracting a "best" association function from the filtered graph. These techniques would be applicable to the original graph. However, since they have a sequential component, it would be computationally more costly to directly process the unfiltered graph.

Let G_f represent the final association graph produced by the node and edge deletion processes; G_f represents a reduced and more informed set of guesses as to which O_i should be associated with a particular T_j. We will discuss the use of ordered search techniques to find the best association function contained in G_f. The combination of state-space search and "discrete" relaxation processes is very similar to the incorporation of probabilistic relaxation with heuristic search in MSYS (Barrow and Tenenbaum [9]).

The states in the search space will be described by association functions and contexts (see below) and the search process will consist of choosing the state S with minimal cost, and extending the association function described by that state by adding an additional association to it chosen from the context of that state.

Given a state, S, let F_s denote the state association function for S and let C_s denote the context for S.

F_s is a set of ordered pairs $\{(O_{i_1},T_{j_1}),(O_{i_2},T_{j_2}),\ldots,(O_{i_n},T_{j_n})\}$ satisfying the properties that

1) $O_{ip} = O_{iq} \Rightarrow p = q$

2) $T_{jp} = T_{jq} \Rightarrow p = q$

It represents the partial development of some association function from T into O.

Each C_s is a subgraph of G_f. The nodes of C_s represent the possible associations which may be used to construct extensions to F_s; these extensions will become the state association functions of the immediate descendents of S. There are three important points to note about these contexts. The first concerns the connectedness of any given context, while the second two concern the generation of contexts during the expansion of a state during search.

First, the context of any state must be a subgraph of a single connected component of G_f. This is because a pair of nodes are contained in different connected components of G_f because they did not satisfy the relation model.

The second point involves the way that the context of a state is constructed from the context of its father. Suppose that state S is the father of state S', and that $F_{s'}$ is obtained by adding the association (O_i,T_j) to F_s. Then $C_{s'}$ is constructed from C_s in the following simple way (Algorithm H-Hypothesize):

1) Set $C^0_{s'} = C_s - \left\{\{(O_{i'},T_j)\,|\,i' \neq i\} \cup \{(O_i,T_{j'})\,|\,j' \neq j\}\right\}$, i.e., delete from C_s any associations involving either O_i or T_j except (O_i,T_j).

2) Apply the node and edge deletion procedures iteratively to $C^0_{s'}$ to obtain $C^1_{s'},C^2_{s'},\ldots,C^k_{s'}$, where $C^k_{s'}$ is the fixed point. Set $C_{s'} = C^k_{s'}$, provisionally.

There are logically three relations that $F_{s'}$ might have to $C_{s'}$:

1) Some of the associations contained in $F_{s'}$ are deleted by the

filtering process. This indicates that $F_{s'}$ was constructed from incompatible associations, and does not describe the best association function.

2) $C_{s'}$ is disconnected by the filtering process, and the associations of $F_{s'}$ are split among the various connected components of the $C_{s'}$. Once again, $F_{s'}$ was constructed from incompatible associations.

3) The associations of $F_{s'}$ are all contained in a single connected component of the $C_{s'}$. In this case, that connected component becomes $C_{s'}$.

The only allowable extensions will be those that fall in the third category. The association functions for such states will be called constraint compatible. Any state S that cannot be extended to an S' because all attempts at extension result in one of the first two conditions will be marked with very high cost. The association functions for such states will be called constraint incompatible. In this way, these states need not be considered during the remainder of the search process.

The third point involves refining C_s in order to reduce the number of sons that might be generated from S. Basically, the question we must answer is: Given that we have generated one son of S by extending F_s with some association (O_i,T_j) in C_s, is it necessary to consider all other nodes in C_s as other possible extensions for F_s? The answer to this question is that in general we do not. In fact, the set of associations that must be considered, given that (O_i,T_j) has already been used to generate a son, can be computed by the following process (Algorithm S-Suppress):

1) Set $C_s'^0 = C_s - (O_i,T_j)$

2) Iteratively apply the node and edge deletion processes to obtain the sequence of graphs $C_s'^1, C_s'^2, \ldots, C_s'^k$, with $C_s'^k$ the fixed point. Set the refined context of S, $C_s' = C_s'^k$.

The following propositions demonstrate that it is sufficient to consider only elements of C_s' as other possible extensions to F_s.

Lemma: C_s is independent of the order of acquisition of the association in F_s.

Proof: Can be found in [19]. The Lemma assures us that there is, effectively, only a single way to reach any state.//

Proposition: Let $S = (F_s,C_s)$. If F_s is constraint-compatible, then all constraint-compatible extensions of F_s can be constructed along a path constrained to contain state S.

Proof: Immediate, from the lemma and the definitions of the node and edge deletion operators.//

Proposition: Let $S = (F_s,C_s)$, and let (O_i,T_j) be a node in C_s. Let C_s' be constructed by applying Algorithm S to C_s and (O_i,T_j). If $(O_{i'},T_{j'})$ is a node in C_s, but is not contained in C_s', then any constraint compatible extension of F_s which does not contain (O_i,T_j) cannot contain $(O_{i'},T_{j'})$.

Proof: Since deleting (O_i,T_j) from C_s and applying the node and edge deletion operator caused $(O_{i'},T_{j'})$ to be deleted from C_s, then postponing the deletion of $<O_i,T_j>$ until some later stage (which must occur since, by hypothesis, (O_i,T_j) will not be part of any constraint compatible extension to F_s) obviously will only delay the deletion of $(O_{i'},T_{j'})$.//

In summary, the state-space contains states S that are ordered pairs (F_s,C_s). The start space is $(\emptyset,G)$ and a final state is a pair (F_s,C_s) where the nodes in C_s are identical to the associations in F_s (i.e., there is no way to expand F_s). Given some cost function C and a method for computing a lower bound estimate for each state of the cost of a minimal cost association function constrained to include that state, we can describe the search algorithm as follows (Algorithm F):

(0) Put $(\emptyset,G)$ on the list of OPEN states.

(1) Choose that state S from OPEN that has minimal cost estimate. If C_s contains only the associations in F_s, stop with F_s being the optimal association function.

(2) Pick some $<O_i,T_j>$ from C_s to build S':

(2') $F_{s'} = F_s \cup <O_i,T_j>$

(2") Obtain $C_{s'}$ by Algorithm H

(2"') Refine C_s to C_s' by Algorithm S

(3) Start again at step (1).

Ordered search techniques can be used both to direct search towards the best terminal state and also to terminate search along a path that could not conceivably be optimal. See [10] for a discussion of a general cost function and cost estimating procedure which can be used to guide the search.

3. Experimental Study

A matching experiment was performed using the coastlines of several islands as the input shapes. The source of the data was World Data Bank I, a digital representation of the coastlines and boundaries of the world specifically designed for automated mapping systems [11]. World Data Bank I is in the form of (longitude, latitude) coordinates. The coastlines were transformed to Cartesian coordinates using the Lambert normal conical projection based on a standard parallel. The data base of objects was created using one set of standard parallels, while the unknowns were produced using a different set of standard parallels. This introduced systematic distortions between the objects and unknowns.

After the projection, each individual coastline was scaled to fit into a 127×127 array so that gross differences in the physical sizes of the coastlines would not serve as a cue for the matching process.

Five coastlines were chosen -- Baffin Island, Cape Breton Island, Cuba, the Dominican Republic, and Guadeloupe. The unknowns were segments chosen from the same coastlines, but projected at different latitudes. Polygonal path representations for the objects and the unknowns were constructed using the angle detection procedure described previously.

In an attempt to provide a difficult matching environment for the matching processes, the unknown shapes were degraded according to the following procedure: Given shape $T = \{(X_i, Y_i)\}_{i=1}^{m}$, choose an arbitrary scale factor, a, and a noise factor, q. Each element of T is multiplied by a + s, where s is chosen from the normal distribution with mean 0 and variance q. That is, we construct a $T' = \{(X_i', Y_i')\}_{i=1}^{m}$ where

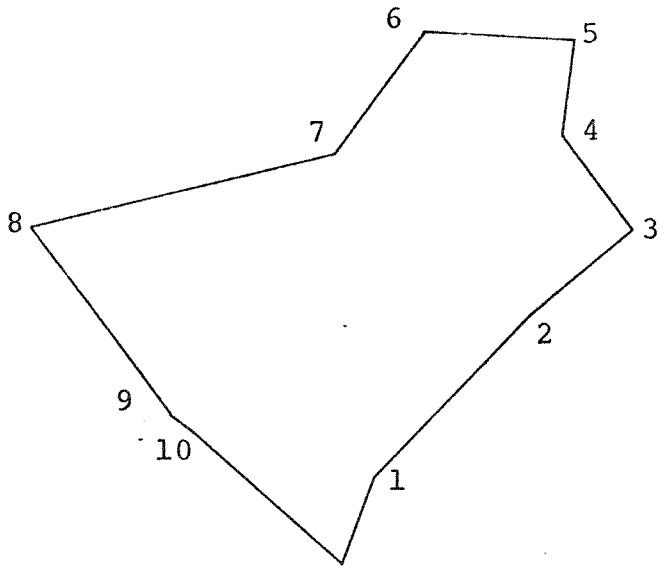
$$X_i' = X_i(a+s), \text{ and}$$

$$Y_i' = Y_i(a+s)$$

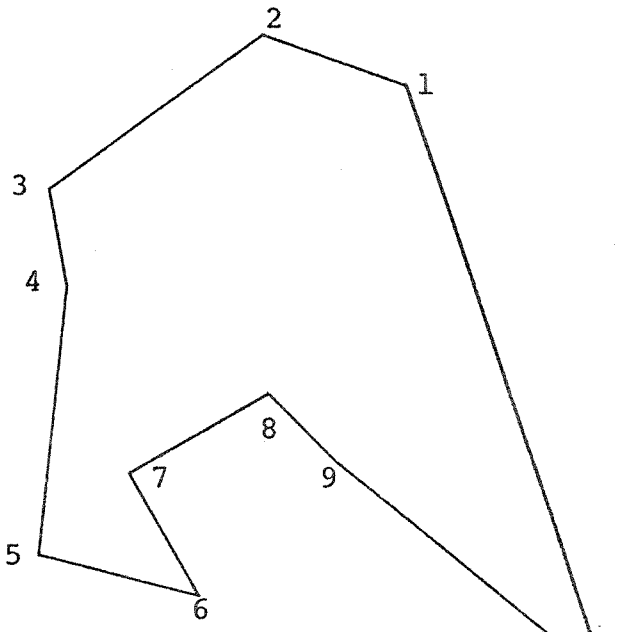
Figure 3 contains the polygonal approximations of the data base for k=10 and Figure 4 shows the 10 scaled, noisy unknown shapes.

The threshold for connecting nodes in the association graph was defined so that two nodes were connected if the associated distances, d_1 (for the template elements) and d_2 (for the object elements), satisfied $.4 < ABS(d_1 - d_2)/d_1 < 2.5$, allowing for a wide range of scale changes. The values of these thresholds determine the degree of scale invariance of the procedures. It should be pointed out that allowing such a wide range of scale changes will clearly place most of the burden on the edge deletion process, and that the node deletion process will only be able to delete the grossest mismatches.

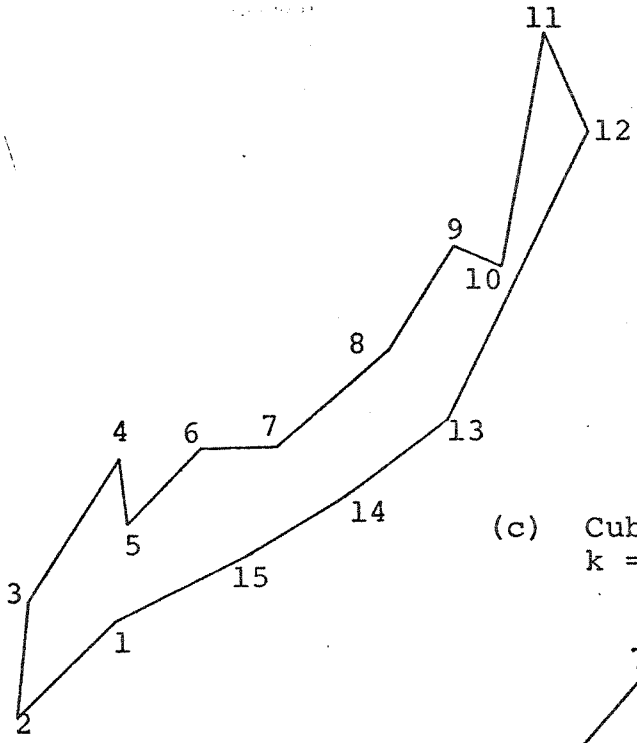
The node deletion criterion was to delete a node when the sum of the weights on its neighbors was less than a threshold, t_d. If the unknown were an exact match to a segment of one of the objects, then we would use a threshold of $t_d = (m-1)c$, (c=minimum value of local angle matches allowed for introducing a node into the network), since every correct association $<O_i,T_j>$ would be connected to m-1 other correct associations. However, since we do not have exact matches, but only approximate matches, we have adopted the following rule which was used consistently: set t_d equal to 2/5 of the number of prominent angles in the unknown shape, where a prominent angle was taken to be one with a magnitude less than 60 degrees. In this way, we allow for matches that only approximately account for the most salient features.



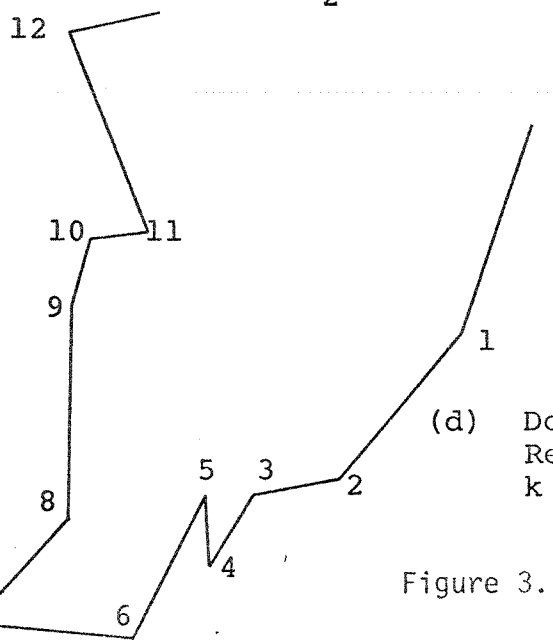
(a) Baffin Island
k = 10



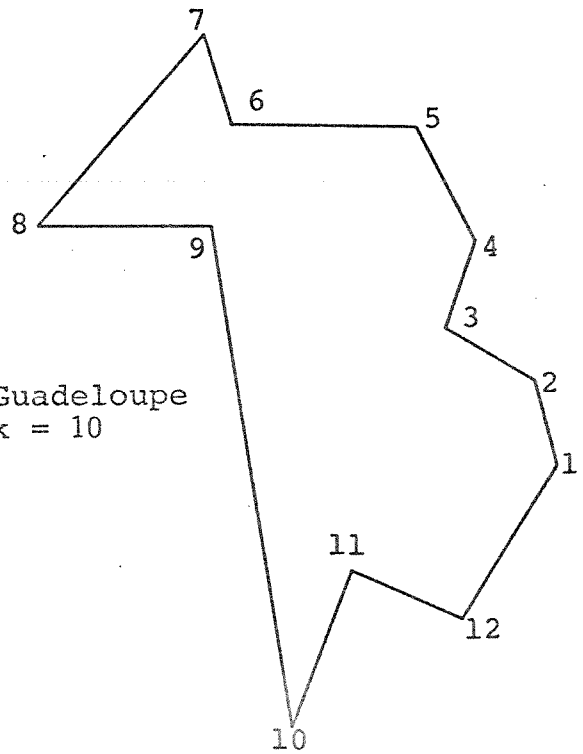
(b) Cape Breton
k = 10



(c) Cuba
k = 10

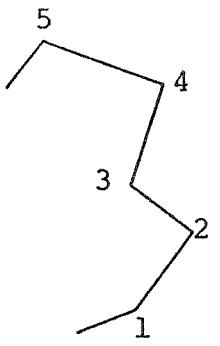


(d) Dominican
Republic
k = 10

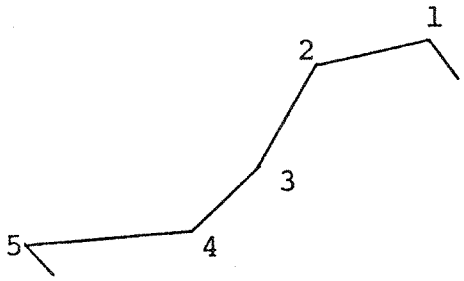


(e) Guadeloupe
k = 10

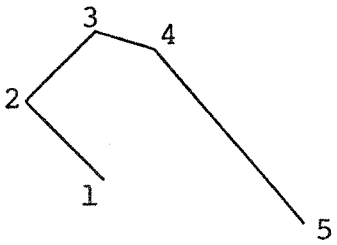
Figure 3. Data base.



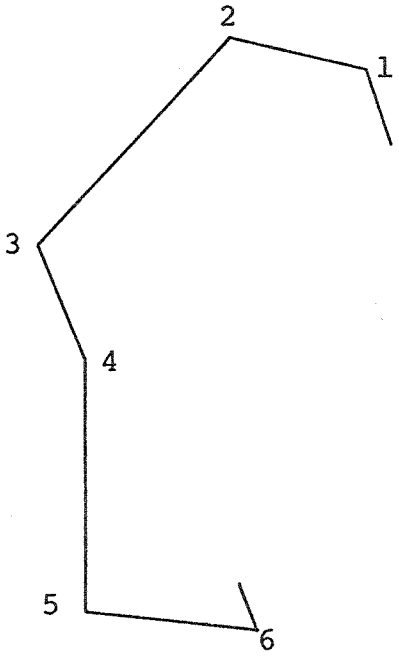
(a) Baffin Island T_1
Scale 1:2



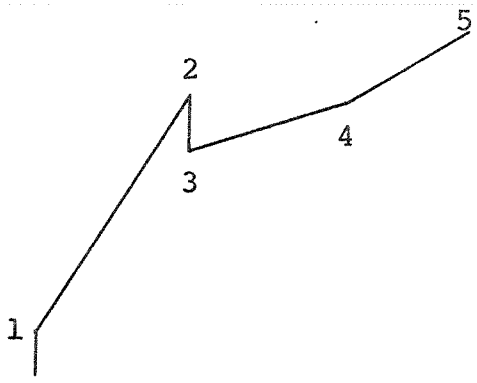
(b) Baffin Island T_2
Scale 1:1



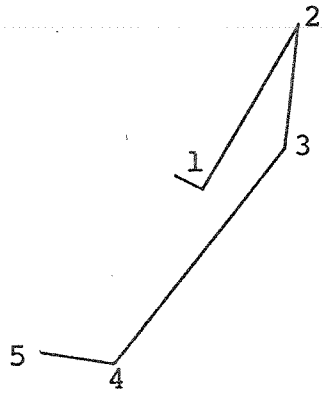
(c) Cape Breton T_3
Scale 1:2



(d) Cape Breton T_4
Scale 1:2

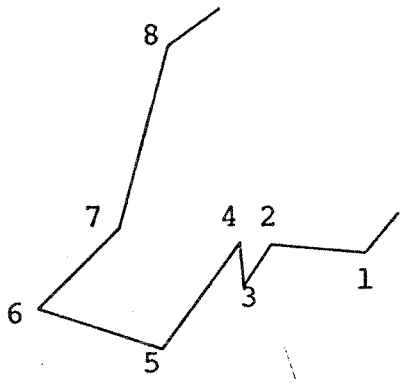


(e) Cube T_5
Scale 1:2

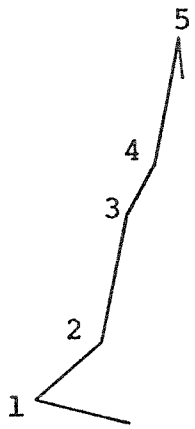


(f) Cuba T_6
Scale 1:1

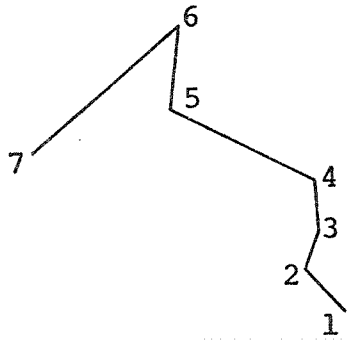
Figure 4. Noisy, scaled templates.



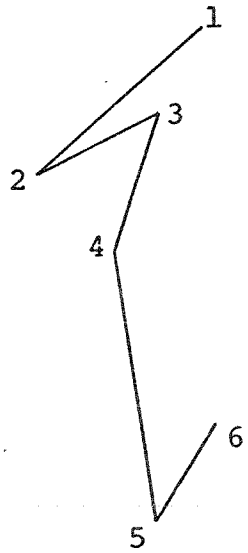
(g) Dominican Republic T_7
Scale 1:2



(h) Dominican Republic T_8
Scale 1:2



(i) Guadeloupe T_9
Scale 1:3



(j) Guadeloupe T_{10}
Scale 1:1

Figure 4. (continued)

The edge deletion process is defined as follows: Consider the triple of nodes shown in Figure 5, and suppose we are evaluating edge a. The similarity transformation associated with edge a is applied to $O_{i''}$, and the distance between the transformed $O_{i''}$ and $T_{j''}$ is computed. If this distance is less than some threshold, one piece of evidence is counted for edge a.

In 47 out of the 50 match trials, the edge deletion process deleted all mismatch associations, while preserving all of the match associations, thus eliminating the need to search the association graph. In the remaining three cases where this situation did not occur, only nominal searching was required (see [10] for more details).

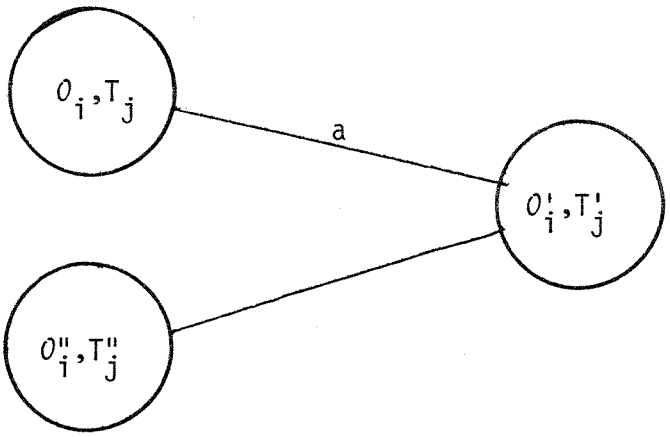


Figure 5. Edge deletion.

4. Syntactic Models

In this section we will turn our attention to syntactic pattern analysis. After a brief review of syntactic pattern analysis we will discuss how relaxation techniques can be integrated with syntactic techniques.

As described by Fu [12], a syntactic pattern analysis system can ordinarily be regarded as consisting of three stages:

1) a preprocessing stage
2) a pattern description stage, and
3) a syntax analysis stage.

The goal of the preprocessing stage is to transform the input pattern into a form from which the description of the second stage can be most easily and reliably computed. The goal of the second stage is to segment the pattern into a set of primitives and to discover the relations between these primitives. The representation constructed by the pattern description algorithm is then examined by the syntax analyzer. It is the responsibility of this last process to determine if the representation is syntactically well-formed, and if so, to produce as a side effect of that decision a parse tree of the representation. This tree is often used for the actual subsequent recognition of the pattern -- i.e., the goal of the syntax analysis is not to recognize the pattern, but rather to impose a natural organization on the pattern using the vocabulary of the grammar so that later processes can recognize the pattern based on this organization. As an example, consider the parsing of a shape which is the outline of an airplane -- the goal of the parse might be to decompose

this outline into natural pieces such as wings, tail, body, etc. Once this is accomplished, then the specific type of the airplane might be determined on the basis of physically meaningful measurements made on that outline, such as ratio of wingspan to body length, etc.

One of the difficulties in applying the syntactic approach is the reliable extraction of primitives from the original pattern. Methods for preprocessing patterns will not be discussed, since they are quite varied and do not, in most cases, transform the pattern into a form where the desired primitives can be discovered with complete reliability. Instead, we will first discuss general procedures for segmenting a shape into pieces which should often be appropriate for the subsequent organization of that shape according to a grammar. We will then turn our attention to the design and use of such grammars for shape analysis.

4.1 Primitive Detection

This section briefly reviews the relevant literature on shape segmentation. The literature on the subject is quite large; Pavlidis [5] contains a more complete review of shape segmentation schemes.

It should be stressed that the specific choice of primitives will depend on the particular set of patterns to be analyzed. However, for a wide variety of shapes, segmentation of the border of the shape into pieces described by low order polynomials (of degree less than or equal to 2) will be sufficient. Procedures for segmenting a shape into pieces based on the distribution of area of the shape will not be considered because one of our principle goals is to be able to recognize a shape even if the entire shape is not available (possibly due to difficulties in segmenting the images containing the shapes, or because of occlusion in these images). In such situations, it is difficult to decompose the shape into areal segments because of the uncertainty associated with defining the closure of the available pieces.

Perhaps the most frequently used shape boundary representation is the piecewise linear approximation of the boundary. Many algorithms have been proposed for computing various piecewise linear approximations -- see, e.g., Pavlidis [13], Ramer [14], Rosenfeld and Johnston [15], and Davis [16]. These procedures can be categorized as being either one of two types -- those that attempt to find the lines directly and those that attempt to find the breakpoints between the lines directly. The first class of procedures search for boundary segments which are well fit by lines, while the second class search for boundary points which have locally high curvature -- they are angle detection procedures. Of all these algorithms,

the split-and-merge algorithm proposed by Pavlidis [13] seems to be the most robust (i.e., it has very slight sensitivity to small changes in the underlying shape) yet is at the same time computationally efficient. One can obtain higher order approximation using this split-and-merge algorithm with higher degree polynomials, but as Pavlidis discusses [17], the computational cost increases dramatically as one raises the degree of the approximating curves. Furthermore, the algorithms become numerically less stable. Pavlidis suggests using the results of the piecewise linear approximation to selectively guide the application of higher order approximation procedures to pieces of the shape boundary, or to actually parse the piecewise linear approximation using a regular grammar which describes quadratic arcs as sequences of linear segments.

Once the segments are obtained, the relations between the segments can be computed. The specific relations computed depend both on the dimensionality of the grammar (e.g., string, tree, graph, see below) and on the semantics associated with the symbols of the grammar. If the grammar is a string grammar, then the relationship of concatenation between primitives must be computed. Notice that for a set of primitives corresponding to linear segments obtained by computing several piecewise linear approximations of the border of the shape (which is how the hierarchical relaxation processes to be described compute primitives), it is not straightforward to compute even the simple concatenation relationship, since we would not expect that segments obtained from different approximations would exactly coincide at their endpoints, rather than overlap slightly. For relations more complicated than adjacent, such as "left of", "right of", "inside", the correct

definitions become more elusive (see Freeman [18] for a survey of models for computing spatial relations and a discussion of the difficulties associated with making such computations).

For the shape grammars to be defined, the only relations that need to be computed between the primitives are relations of concatenation. As the syntax analysis proceeds, more complex relations, such as collinearity and symmetry, need to be computed, but their computation is heavily directed by the results of the syntax analysis.

4.2 Syntax Analysis

Once the representation of the pattern in terms of primitives and relations between the primitives has been formed, that representation is presented to the syntax analyzer (this strict separation of stages 2 and 3 is a simplification -- see the discussion of top-down parsing strategies below). The complexity of the syntax analysis is a function both of the type of the grammar, and the dimension of the representations which it is designed to analyze. The simplest grammar is a finite state string grammar. Here, the only relations between primitives allowed are string concatenation and the parser can analyze its input in time linear in the number of symbols in the input. Pavlidis [17] has suggested that a wide variety of shape analysis problems can be solved using what are essentially finite state grammars. It should be pointed out that few of the grammars described in the literature for shape analysis are parsed using pure syntactic analyzers, but that all have substantial semantic components. This is necessary because the primitives used do not ordinarily constitute a finite set, but rather are described by a few real numbers along with a type -- e.g., a straight primitive might have the semantic attributes of length, slope, etc. To attempt to directly account for the variability of possible straight lines by independent syntactic entities would lead to an explosion in the size of the grammar.

As the generality of the grammar is increased from finite state to context free and context sensitive, the computational complexity of the parsing procedure necessarily increases. Context sensitive grammars have almost never been used to describe pattern classes. Context free grammars, on the other hand, have been used extensively. Examples include grammars

for chromosome analysis (Ledley [19]), English characters (Narasimhan [20]), and fingerprints (Moayer and Fu [21]); Fu [12] contains an extensive list of other applications. In the next section we will suggest the use of stratified context free grammars for shape description. They represent a restriction on the class of context free grammars which allows for the computation of local constraints on the spatial arrangement of shape pieces at many levels of description. These constraints are what allow the hierarchical relaxation process to eliminate "dead-end" hypotheses about the shape while at the same time producing a parse of the shape.

The syntax analysis can also be made more complex by increasing the dimensionality of the representations. One is ordinarily forced in this direction because the one dimensional relation of concatenation is not powerful enough to allow for the natural description of two dimensional patterns. Several generalizations to string grammars have been introduced to overcome these problems. Shaw [22] increased the number of concatenation relations between primitives by associating a "head" and a "tail" with each primitive. Feder [23] introduced the notion of a "plex" grammar where the number of attachment points for syntactic entities was arbitrary. The languages described by plex grammars are very much like graph languages, and in fact string grammars have also been generalized to graph languages (see Pfaltz and Rosenfeld [24]). If one restricts the class of graphs to trees, then one can construct tree grammars (see Moayer and Fu [21]).

Parsing structures generated by higher dimensional grammars can present severe computational difficulties. The worst case analysis for parsing a graph grammar is that the parsing problem is in the class of NP-complete problems, since rule matching requires solving the subgraph

isomorphism problem.

Parsing algorithms used to analyze patterns can generally be classified as being either bottom-up or top-down. Aho and Ullman [25] contain an extensive treatment of parsing procedures for string languages.

Top-down parsers have often been used in syntactic pattern analysis (see, e.g., [26]). One of the advantages of a top-down parsing algorithm is that it naturally allows for the selective application of the primitive detection procedures to the pattern, since these procedures are invoked only when a primitive is predicted to occur at a specific location in the pattern.

However, in a situation where the primitives cannot be reliably detected, a pure top-down parsing procedure may be particularly ineffective since the pattern locations at which predictions are made are determined by a particular pattern scan procedure (usually left-to-right) built into the parser, rather than through investigation of the pattern itself. Thus, the top-down parsing procedures can be expected to perform a significant amount of backing up during the syntax analysis, since their predictions about particularly ambiguous sections of the shape will often be in error. This situation is analogous to the thrashing behavior of backtrack programming (see Mackworth [27]). There, the situation can often be alleviated by introducing certain simple constraint application operators (see Haralick and Gordon [28]). The hierarchical relaxation procedures attempt to solve the parsing problem in a similar way -- i.e., by the application of a set of local contextual constraints to hypotheses about the correct descriptions of shape pieces.

5. Hierarchical Relaxation Using Syntactic Models

In this section we will present the design of a hierarchical constraint analysis procedure which can apply a syntactic model to an unknown shape. Section 5.1 describes the class of syntactic models, called stratified context-free grammars. Section 5.2 is devoted to a discussion of the hierarchical relaxation system, and Section 5.3 describes the application of this system to airplane recognition.

5.1 Grammatical Shape Models

Grammatical models for shape analysis have been developed and investigated by Fu [12]. With some simple modifications these kinds of models can be integrated in a natural way with relaxation techniques. An extension of the geometrical grammars of Vámos [29] and Gallo [30] will be used to model shapes.

Definition: A stratified context-free grammar, G, is a quadruple (T, N, P, S), where

T is the set of terminal symbols,

N is the set of non-terminal symbols,

P is the set of productions, and

S is the set of start symbols.

Let $V = (N \cup T)$ be the set of vocabulary symbols. Associated with every symbol $v \in V$ is a level number, $\ell n(v)$: $V \rightarrow \{0,1,\ldots,n\}$. For every $v \in T$, $\ell n(v) = 0$.

1) T -- corresponds to relatively large pieces of the shapes to be modeled, e.g., straight-edge approximations to the boundary of the shape.

2) N -- consists of a set of symbols, each of which has a level number from 1 to n associated with it. A start symbol has level number n, and in any rule $v := a$ (the rewrite part of a production), if $\ell n(v) = k$, $1 \leq k \leq n$, then every symbol in the string a is at level k-1. Furthermore, for every $v \in V$,

   v: = <name part> {attachment part} [semantic part], where

   a) <name part> is a unique name by which the symbol v is known,

   b) {attachment part} is a set of attachment points of the symbol,

   c) [semantic part] is a set of predicates which describe certain aspects of the symbol.

3) P -- consists of productions of the form $(v := a, A, C, G_a, G_s)$, where

   a) $v := a$ is the rewrite part that indicates the replacement of the symbol v by the group of symbols a, where

      $v \in N$, and

      $a = v_1, v_2, \ldots, v_k$ ($v_i \in V$ and $\ell n(v_i) = \ell n(v)-1$), $i = 1,k$)

   b) A -- set of applicability conditions on the syntactic arrangement of the v_i, $i = 1,k$.

   c) C -- semantic consistency of the v_i, $i = 1,k$. C consists of various predicates describing geometric relations and other constraints that must hold between the v_i.

   d) G_a -- rules for generating the attachment part for v.

   e) G_s -- rules for generating the semantic part of v.

As an example of the productions of the grammar, consider how engines are formed (see Figure 6):

<engine> $\{e_1, e_2\}$ [a,span]: =

<engine side> $\{e_1', e_2'\}$ [a'] +

<engine front> $\{e_1'', e_2''\}$ [a"] +

<engine side> $\{e_1''', e_2'''\}$ [a"']

A: [Join (e_1' or e_2', e_1'') and Join (e_1''' or e_2''', e_2'')
or Join (e_1''' or e_2''', e_1'') and Join (e_1' or e_2', e_2'')]

C: [Parallel (a', a"') and Length (a') = Length (a"')
and Perpendicular (a', a")
and Parallel (a", Vector (Midpt (a'), Midpt (a"')))]

G_a: [e_1: = Unjoined (e_1', e_2') and e_2: = Unjoined (e_1''', e_2''')
or e_1: = Unjoined (e_1''', e_2''') and e_2: = Unjoined (e_1', e_2')]

G_s: [a: = (a' + a"')/2 and span: = a"]

This rule specifies that an "engine" is composed of two "engine side" symbols and an "engine front" symbol. A, C, G_a and G_s can be viewed as a program for producing "engine" from symbols on the right-hand side of the rewrite rule. A specifies the physical connections of the symbols on the right-hand side, i.e., that each end of the "engine front" has an "engine side" attached to it, but the "engine side" symbols are not connected to each other. C indicates that the two "engine side" symbols should be parallel, of the same length, perpendicular to the "engine front" symbol, and on the same side of the "engine front". G_a and G_s describe the

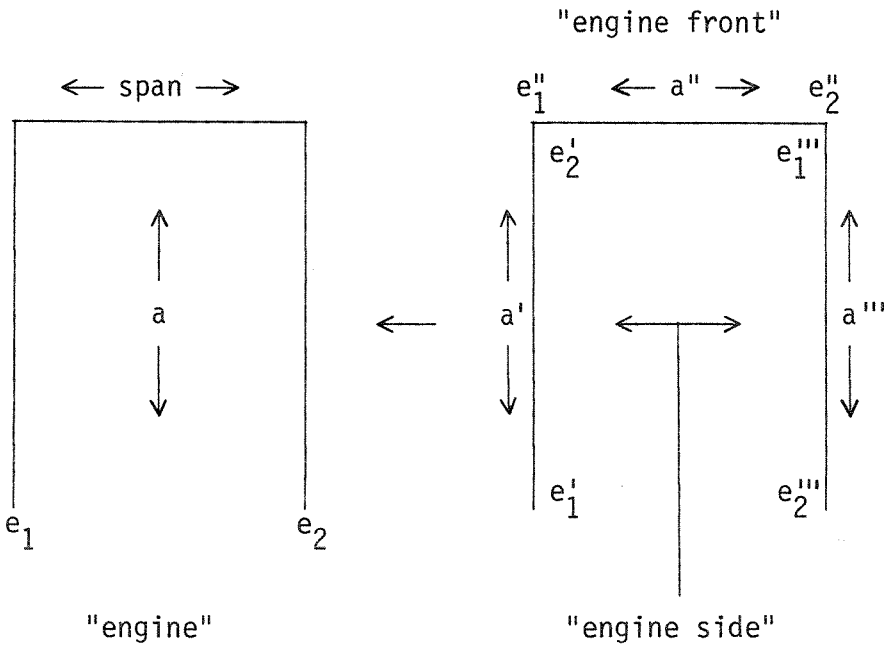


Figure 6. Example of a production.

derivation of real values for the attachment points and semantic features for "engine"; the unjoined end points of the "engine side" symbols can be given either attachment point name due to the symmetry of the symbol. The main axis is the average of those of the "engine side" symbols, and the span is exactly that of "engine front".

Stratified grammars naturally give rise to a large set of contextual constraints on the organization of a shape. It is these constraints which the hierarchical relaxation will utilize to analyze shape.

5.2 Hierarchical Relaxation

As discussed previously, a major problem of any syntactic pattern recognition scheme is the segmentation of the object into pieces which correspond to the terminal symbols of the grammar. A high false alarm rate implies that many primitives will be generated, and correspondingly many terminal symbols hypothesized from them, thus implying a large search space. In order to overcome these difficulties, a hierarchical relaxation process (HRP) uses hierarchical models of objects and model derived constraints to eliminate inconsistent hypotheses at each level of the model. In particular, using the stratified context-free grammars already described, syntactic (e.g., spatial concatenation) and semantic (e.g., symmetry, collinearity, etc.) constraints can be automatically generated to guide the analysis of the shape.

Primitives for the grammatical analysis are generated by computing several piecewise linear approximations to the boundary of the shape. A modified split-and-merge algorithm fits straight edges to the boundary using the cornerity measure proposed by Freeman and Davis [31] to choose break points. Primitives are generated at various error thresholds, and stricter thresholds are applied to segmentations already generated to generate more segments. By computing several segmentations, it is hoped that almost all of the necessary primitives will be found. The search will be made feasible by the constraints implicit in the grammar and imposed by the relaxation techniques.

HRP operates by iteratively applying a set of local contextual constraints to a hierarchical network of hypotheses about a pattern. The application of the constraints causes nodes to be removed from the network.

If a node is removed from the n^{th} level of the network, then this may lead to other nodes at levels n-1, n or n+1 being deleted from the network. When the network is stable, HRP builds the next layer of the network corresponding to hypotheses about the next level of description of the pattern. When the top-most level is constructed, all nodes at that level represent alternate (although not necessarily independent) parses of the original pattern.

We will first describe how the local constraints can be compiled from a stratified grammar, and then discuss how HRP utilizes these constraints. We will restrict our attention to conventional string grammars. The extension to shape grammars is presented in Davis and Henderson [32].

Let V be the vocabulary of our grammar, and let $s \in V$. Then, define the set

$$s(k) = \{s' : s' \in V, \text{level}(s) = \text{level}(s'), \text{and there is some sentential form of level (s) symbols } x_1, x_2, \ldots, x_n, \text{ with } x_t = s \text{ and } x_t + k = s'\}.$$

Thus, s(k) is the set of symbols which can be found a distance k from s in a level(s) sentential form of the grammar. These sets can be easily computed, given the grammar.

We use the sets s(k) to construct local contextual constraints as follows: Let (s,p_1,p_2) denote a node in our network representing the hypothesis that the pattern segment from position p_1 to position p_2 is described by the symbol s. Then we can use the set s(1) to produce the local constraint: Retain the node (s,p_1,p_2) in the network only if there is another node (s',p_2+1,p_3) with $s' \in s(1)$. Similarly, we can create a

local constraint based on s(-1), or any other s(k).

The hierarchical relaxation system computes a bottom-up parse of the shape by applying the constraints to a network of low-level hypotheses about pieces of the shape. The processing of this network can be easily described by specifying three simple procedures and two sets which these procedures manipulate.

BUILD. Given level k of the network, BUILD uses the productions of the grammar to make level k+1 hypotheses. Any level k symbols which are used to generate a node at level k+1 are associated with that level k+1 node as supporting it, and it in turn is recorded as being supported by them. After all nodes are generated, nodes corresponding to boundary segments sharing an endpoint are linked only if the constraints allow the symbols hypothesized for each node to be adjacent at that endpoint. Building level 0 involves applying the segmentation strategy to the shape to generate the level 0 nodes.

RELAX. Since each node corresponds to a single hypothesis, and since nodes are only linked to compatible nodes, the within layer relaxation simply involves removing a node if it has no neighbor at some endpoint. Note that the process can be given some noise resistance by just demanding that some endpoints have neighbors, rather than all.

REDUCE. After BUILD generates level k+1, any level k node which does not support a level k+1 node will be removed. For any level k node which is removed, all level k-1 nodes which support only it are removed. If a level k node is removed, any level k+1 node it supports is removed.

These procedures operate on two sets of nodes, R_x and R_c, both of which are initially empty. When at level k with R_x and R_c empty, BUILD

produces the level k+1 hypotheses (or stops if k=n), and puts them into R_X while putting all level k nodes into R_C. RELAX then removes nodes from R_X, taking no action if the node has a neighbor at all endpoints, but otherwise deleting the node from the network and putting its same level neighbors in R_X and its across level neighbors in R_C. REDUCE removes nodes from R_C, taking no action if all the node's original supporting nodes still exist at level k-1 and the node still supports at least one level k+1 node (if level k+1 has been built); otherwise, REDUCE deletes the node from the network and puts its same level neighbors in R_X and its across level neighbors in R_C.

5.3 Examples

A PASCAL program (4600 lines, 50k) implementing HRP has been written and runs on the DEC-10 computer at the University of Texas. Input to HRP consists of a stratified shape grammar defining the class of shapes to be analyzed and a set of primitives, i.e., line segment descriptions including orientation, length and endpoints. HRP produces a (possibly empty) network of hypotheses relating primitives to the vocabulary symbols at each level of the grammar. Thus, any level n hypothesis corresponds to a shape in the grammar.

A grammar describing the top view of airplane shapes (down to the level of detail of engines) has been developed. The grammar consists of 33 productions and has seven levels of vocabulary symbols. Note that the grammar was not designed to describe a particular airplane (such as a 747), but rather to model a wide class of airplanes. We do not view parsing as a recognition procedure but rather as a process which imposes organization on the shape (by forming engines, wings, etc.). Recognition is subsequently performed by analyzing the organization.

We will describe the application of HRP to the top view of the airbus in Figure 7 (copied from You and Fu [33]). The split-and-merge algorithm was used to obtain piecewise linear approximations to the boundary of the airbus at several thresholds of goodness of fit. For this shape using three thresholds, a total of 35 primitives were found, of which only 27 were needed to define the shape of the airbus.

Once the primitives are generated, each one must be associated with an initial set of level zero hypotheses. Analysis of the histograms of segment lengths and orientations allows many hypotheses to be rejected,

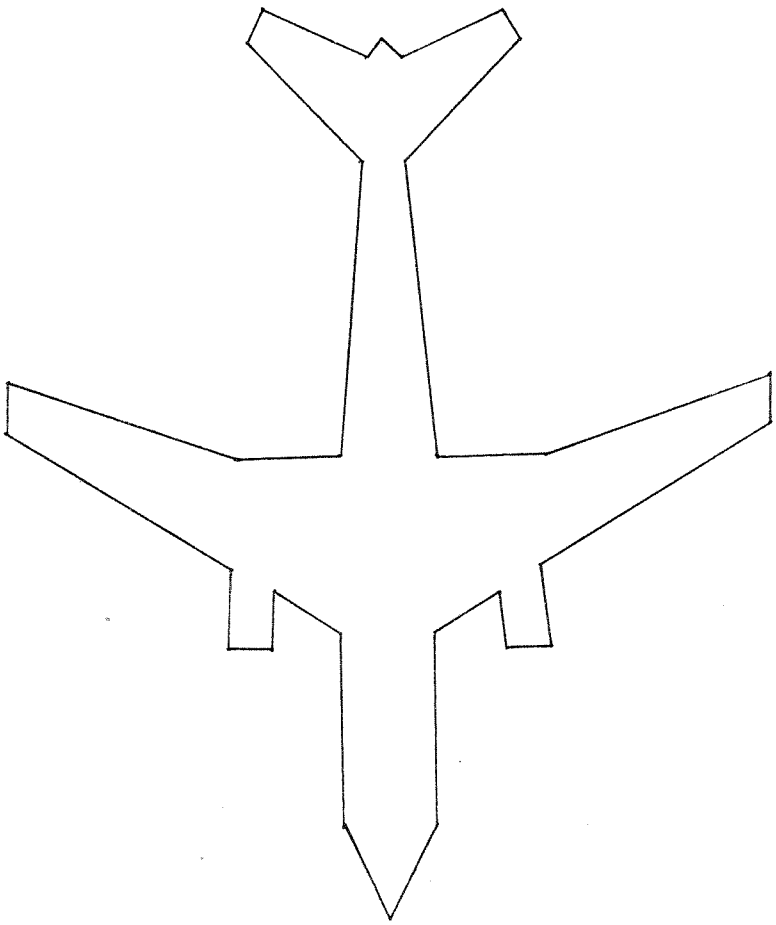


Figure 7. Airbus

e.g., the very longest segments, if substantially longer than the rest, are not very likely to be wing tips. When the initial set of labels was restricted to only the correct hypotheses, a total of 87 nodes were generated during a complete parse, and HRP required one minute and eleven seconds of CPU time to analyze the shape. Table 1 describes the results of runs with 1, 2 and 3 initial hypotheses per primitive. At each level RELAX and REDUCE eliminated unsupported hypotheses, and the last column of Table 1 shows the complete stable network. Run times for these sets of initial hypotheses were 1 minute 11 seconds, 2 minutes 22 seconds, and 4 minutes 51 seconds, respectively.

Level generated	0	1	2	3	4	5	6	Total	Time
A. 1.0 hyp/prim									
New hypotheses	35	27	23	15	8	2	1	111	1:11
After Relax and Reduce									
0	31	31	31	27	27	27	27		
1		27	27	23	23	23	23		
2			23	19	19	19	19		
3				9	9	9	9		
4					6	6	6		
5						2	2		
6							1		
B. 2.3 hyp/prim									
New hypotheses	81	55	48	33	8	2	1	228	2:22
After Relax and Reduce									
0	59	59	44	27	27	27	27		
1		55	40	23	23	23	23		
2			36	19	19	19	19		
3				9	9	9	9		
4					6	6	6		
5						2	2		
6							1		
C. 3.2 hyp/prim									
New hypotheses	114	74	58	32	9	2	1	290	4:51
After Relax and Reduce									
0	80	75	50	29	27	27	27		
1		71	46	25	23	23	23		
2			42	21	19	19	19		
3				11	9	9	9		
4					6	6	6		
5						2	2		
6							1		

Table 1 - Hypothesis generation and deletion (by level)

6. Summary

This chapter has discussed planar shape matching based on relational network representations and syntactic representations. Both shape matching procedures relied heavily on constraint propagation techniques, often referred to as relaxation procedures. To conclude this chapter, we will discuss to what extent these two matching procedures deal with the five factors (described in Section 1) which contribute to making the shape recognition problem difficult.

1) orientation--both matching procedures solve the orientation problem by basing their computations on orientation invariant features -- e.g., angles, adjacency, etc.

2) size--the graph matching procedure deals with variations in size by explicitly computing a similarity transformation for the best match which includes scale information. HRP does not compute a scale factor in the same way that the graph matcher does, because its models are generic (i.e., meant to represent a variety of shapes), while the graph matcher models are highly specific (e.g., the coastline of Cuba). Neither representation, however, can currently account for the changes in appearance of an object as size changes, although the syntactic models that HRP uses can primitively deal with this by having many "lowest" levels (i.e., levels where the symbols correspond to entities that can be simply and directly extracted from the shape). However, this in no way accounts for the continuous changes in appearance that accompany continuous size changes.

3) distortion--both matching procedures deal with distortion

implicitly by allowing local discrepancies in matching, but demanding a high level of global consistency between model and unknown. So, for example, the graph matcher is very lenient when matching pairs of angles between the model and the unknown, but subsequently applies a more stringent global consistency check during node and edge deletion. A weakness of both procedures is their lack of any explicit model for distortion. This is quite difficult to obtain, but see Ullmann [34] for a specific model of distortion for character recognition, and Grenander [35] for a more general discussion of pattern distortion.

4) partial information--as discussed in Section 2, the graph matcher was specifically designed to solve the segment matching problem. HRP can also naturally be adapted to recognize shapes based on partial information by simply applying HRP using a subgrammar of the original grammar. For example, if there is reason to believe that only partial information is available (e.g., the shape may not be closed), HRP can apply the subgrammar rooted at "wing" to find any wings in the available data. Of course, once a wing is found, it can be used to predict the location of other pieces, so that the maximal amount of the shape data available can be analyzed with respect to the model. We are currently constructing a shape analysis system which uses HRP to solve the partial information problem in just this way.

5) agglomeration--although neither system has been applied to a shape matching problem involving agglomeration, it seems clear that HRP could more naturally deal with agglomeration than the graph matcher, whose node and edge deletion criteria are, in practice, determined by the size (i.e., number of angles) in the unknown. HRP could approach the

agglomeration problem in much the same way that it would solve the partial information problem -- i.e., by finding maximal subsets of the available data that can be consistently accounted for by the model.

References

1. D. Marr and K. Nishihara, "Representation and recognition of the spatial organization of three-dimensional shapes," *Phil. Trans. Roy. Soc. B, 275.*

2. T. Pavlidis, "A review of algorithms for shape analysis," Princeton University, C. S. Tech. Rep. 216, Sept. 1976.

3. R. Duda and P. Hart, *Pattern Classification and Scene Analysis,* Wiley, New York, 1973.

4. T. Pavlidis, "A survey of algorithms for shape analysis," Proc. 4th Int. Joint Conf. on Pattern Recognition, Kyoto, Japan, 1978.

5. T. Pavlidis, *Structural Pattern Recognition,* Springer-Verlag, Berlin, 1977.

6. H. Freeman, "On the encoding of arbitrary geometric configurations," *IRE Trans. Electron. Comput.,* vol. EC-10, pp. 260-268, 1961.

7. L. Shapiro, "A structural model for shape,"

8. L. Davis and A. Rosenfeld, "Applications of relaxation labeling: Spring-loaded template matching," *Proc. 3rd Int. Joint Conf. Pattern Recognition,* pp. 591-597, 1976.

9. H. Barrow and J. Tenenbaum, "MSYS: A system for reasoning about scenes," Stanford Research Institute AI Tech. Note 121, 1976.

10. L. Davis, "Shape recognition using relaxation techniques," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. PAMI-1, pp. 60-72, 1979.

11. World Data Bank I. NTIS PB-223 178/PDK.

12. K. S. Fu, *Syntactic Methods in Pattern Recognition.* New York: Academic Press, 1974.

13. T. Pavlidis and S. Horowitz, "Segmentation of plane curves," *IEEE Trans. Computers,* vol. C-23, pp. 860-870, 1974.

14. U. Ramer, "An iterative procedure for polygonal approximation of plane curves," *Computer Graphics and Image Processing,* 1, pp. 244-256, 1972.

15. A. Rosenfeld and E. Johnston, "Angle detection digital curves," *IEEE Trans. Computers,* vol. C-22, pp. 874-878, 1973.

16. L. Davis, "Understanding shape, I: Angles and sides," *IEEE Trans. Computers,* vol. C-26, pp. 236-242, 1977.

17. T. Pavlidis and F. Ali, "Computer recognition of handwritten numerals using polygonal approximation," *IEEE Trans. Systems, Man and Cybernetics,* vol. SMC-5, pp. 610-614, 1975.

18. J. Freeman, "The modelling of spatial relations," *Computer Graphics and Image Processing,* 4, pp. 156-171, 1975.

19. R. Ledley, L. Rotolo, T. Golab, J. Jacobsen, M. Ginsberg and J. Wilson, *Optical and Electro-Optical Information Processing,* (ed. J. Tippet et al), M.I.T. Press, Cambridge, MA, 1965.

20. R. Narasimhan, "On the description, generation and recognition of classes of pictures," in *Automatic Interpretation and Classification of Images,* (ed. A. Grasselli), Academic Press, New York, 1969.

21. B. Moayer and K. S. Fu, "Fingerprint Classification," in *Syntactic Pattern Recognition Applications*, (ed. K. S. Fu), Springer-Verlag, Berlin, pp. 179-214, 1977.

22. A. Shaw, "A formal picture description scheme as a basis for picture processing systems," *Information and Control, 14*, pp. 9-52, 1969.

23. J. Feder, "Plex languages," *Information Sciences, 3*, pp. 225-241, 1971.

24. J. Pfaltz and A. Rosenfeld, "Web grammars," *1st Int. Joint Conf. on Artificial Intelligence*, Washington D.C., pp. 609-619, 1969.

25. A. Aho and J. Ullmann, *The Theory of Parsing, Translation and Compiling, Vol. I*, Prentice Hall, NJ, 1972.

26. G. Stockman, "A problem-reducing approach to the linguistic analysis of waveforms," University of Maryland, TR-538, May 1977.

27. A. Mackworth, "Consistency in networks of relations," *Artificial Intelligence*, 8, pp. 99-118, 1977.

28. R. Haralick and G. Elliott, "Increasing tree search efficiency for constraint satisfaction problems," to be presented at *IJCAI*, Tokyo, 1979.

29. T. Vámos and Z. Vassy, "Industrial pattern recognition experiment--a syntax aided approach," *IJCPR-73*, Washington, pp. 445-452.

30. V. Gallo, "A program for grammatical pattern recognition based on the linguistic method of the description and analysis of geometrical structures," *IJCAI-75*, Tbilisi, Georgia, USSR, pp. 628-634, 1975.

31. H. Freeman and L. Davis, "A corner-finding algorithm for chain-coded curves," *IEEE Trans. Computers*, pp. 297-303, March, 1977.

32. L. Davis and T. Henderson, "Shape recognition using hierarchical constraint analysis," to appear in Pattern Recognition and Image Processing Conference, Chicago, IL, 1979.

33. K. You and K. S. Fu, "Syntactic shape recognition," in Image Understanding and Information Extraction, Summary Report of Research for the period Nov. 1, 1976 - Jan. 31, 1977, T. S. Huang and K. S. Fu, Co-Principal Investigators, March 1977, pp. 72-83.

34. J. R. Ullmann, "Subset methods for recognizing distorted patterns," *IEEE Trans. Systems, Man and Cybernetics*, vol. SMC-7, pp. 180-191, 1977.

35. V. Grenander, *Pattern Synthesis*, Springer-Verlag, Berlin, 1976.