HIERARCHICAL CONSTRAINT PROCESSES FOR SHAPE ANALYSIS

Larry S. Davis

Thomas C. Henderson

November 1979 TR-115

This research was supported in part by funds derived from the National Science Foundation under Grant ENG-7904037.

ABSTRACT

A major application of syntactic pattern recognition is the analysis of two dimensional shape. This paper describes a new syntactic shape analysis technique which combines the constraint propagation techniques which have been so successful in computer vision with the syntactic representation techniques which have been successfully applied to a wide variety of shape analysis problems. Shapes are modeled by stratified shape grammars. These grammars are designed so that local constraints can be compiled from the grammar describing the appearance of pieces of shape at various levels of description. Applications to the analysis of airplane shapes are presented.

1. INTRODUCTION

A major application of syntactic pattern recognition is the analysis of two-dimensional shape. In order to adopt the syntactic approach, the shapes to be analyzed must be segmented into pieces which correspond to the terminal symbols of some grammar, and these pieces must subsequently be analyzed by a parsing mechanism. Many syntactic methods assume that the pieces can be found easily (top-down methods provide a wide class of exceptions, e.g., see Stockman [1]). However, in most real problems, the design of a segmentation procedure that can find (almost) all of the pieces will require the acceptance of a high false alarm rate - i.e., many of the hypothesized pieces may not, in fact, be part of a "grammatical" description of the shape.

This paper discusses a general parsing procedure which has been designed specifically to overcome this problem. Shapes are modeled by hierarchical, or stratified grammars. These grammars are designed in such a way that local contextual constraints on the appearance of a shape can be automatically compiled from the grammar at all levels of description of the shape. These constraints can then be iteratively applied to an initial set of hypotheses by a relaxation procedure (see Davis [2] or Davis and Rosenfeld [3]). In what follows, we will describe algorithms designed to compile these constraints and to employ the constraints to analyze shapes.

syntactic and semantic contextual constraints for all the vocabulary symbols can be generated automatically from such grammars. The contextual constraints provided by the shape grammars can be exploited by a hierarchical constraint process. Such a process constitutes a bottom-up constraint-based parsing method and attempts to overcome the combinatorial explosion in parsing the shape implied by the segmentation. Examples of the application of this hierarchical system to airplane recognition are described in Section 4. Performance criteria for a hierarchical constraint process are defined, and several shapes are analyzed and the hierarchical constraint process evaluated according to the criteria. Section 5 discusses how hierarchical constraint processes can be used with uncertain hypotheses. Finally, Section 6 contains a brief summary.

```
        and Perpendicular(a',a'')
        and Parallel(a'',Vector(Midpt(a'),Midpt(a''')))  ]
```

```
   Ga : [ Set(e1, Unjoined(ei',e2')) and
          Set(e2, Unjoined(e1''',e2''')) or
          Set(e1, Unjoined(e1''',e2''')) and
          Set(e2, Unjoined(e1',e2')) ]
```

```
   Gs : [ 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 (see Figure 1). The predicate Join(x,y) is true if x and y correspond to the same point in the shape. 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 derivation of 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 function Unjoined(x,y) computes the endpoint which did not satisfy the Join condition in the applicability condition of the production. The function Set(x,y) assigns the physical attributes of the existing endpoint y to the endpoint x of the symbol being constructed. The main axis is the average of those of the "engine side" symbols, and the span is exactly that of "engine front".

As a simple example of a complete grammar, consider a house grammar G = (T,N,S,P), where

T = { roof, wall, floor },

N = { top, bottom, house },

S = { house }, and

P = {

(production 1) :

```
<top> {e1,e2} [a] :=
      <roof> {e1',e2'} [a'] +
      <roof> {e1",e2"} [a"]
 a : [ Join(e1',e1") or Join(e1',e2") or
      Join(e2',e1") or Join(e2',e2") ]
 c : [ Perpendicular(a',a") and
       Equal( Length(a'),Length(a") ) ]
Ga :  [ Set(e1,Unjoined(e1',e2')) and
       Set(e2,Unjoined(e1",e2")) or
       Set(e1,Unjoined(e1",e2")) and
       Set(e2,Unjoined(e1',e2')) ]
Gs :  [Set(a,Vector(e1,e2)]
```

(production 2) :

```
<bottom> {e1,e2} [a] :=
         <wall> {e1',e2'} [a'] +
         <floor> {e1",e2"} [a"] +
         <wall> {e1"',e2"'} [a"']
 a : [ Joined(e1',e1") and Joined(e2",e1"') or
       Joined(e1',e1") and Joined(e2",e2"') or
       Joined(e1',e2") and Joined(e1",e1"') or
       Joined(e1',e2") and Joined(e1",e2"') or
       Joined(e2',e1") and Joined(e2",e1"') or
       Joined(e2',e1") and Joined(e2",e2"') or
       Joined(e2',e2") and Joined(e1",e1"') or
       Joined(e2',e2") and Joined(e1",e2"') ]
 c : [ Parallel(a',a"') and Perpendicular(a',a")"and
       Equal( Length(a'),Length(a"),Length(a"') ) ]
Ga :  [Set(e1,Unjoined(e1',e2')) and
       Set(e2,Unjoined(e1"',e2"')) or
       Set(e1,Unjoined(e1"',e2"')) and
       Set(e2,Unjoined(e1',e2')) ]
Gs :  [Set(a,a")]
```

3.2 Compilation of Constraints

Two types of constraints, syntactic and semantic, can be compiled from a stratified shape grammar. Syntactic constraints describe the possible neighbors a symbol may have at a specific attachment point. If v is a symbol in a grammar, G, then let Nei(v,a) denote the set of ordered pairs of symbols and attachment points which can be attached to v at attachment point a in some sentential form of G. That is, (v',a') $\in$ Nei(v,a) if and only if v can be attached to v' using point a of v and a' of v'. Now, suppose during the analysis of an actual shape, a shape segment s is hypothesized to be an instance of v. Then some actual point of s, say p, is associated with a by the hypothesis. Of course, other segments have been hypothesized as corresponding to other vocabulary symbols. A necessary condition for the hypothesis relating v to s to be part of a grammatical description of the shape is that some other hypothesis relates symbol v' to a segment s' and a point p' in s' to attachment point a' of v' such that

1) (v',a') $\in$ Nei(v,a), and

2) p' is actually attached to p in the shape.

The sets Nei(v,a) represent the syntactic constraints, and they can be utilized to discard extraneous hypotheses. If these constraints are not applied to the analysis of a shape, then several levels of vocabulary symbols might be built before it is discovered that some hypothesis lacks appropriate context. The use of such constraints, however, makes it possible to detect the lack of appropriate context much earlier.

ay $\in$ at(y) such that x ancestor:ax,av v, and y neighbor:ay,ax x, and w descendent:aw,ay y. Note that computing the neighbor relation for level k symbols assumes knowing the neighbor relation for all levels greater than k.

Using matrix representations for these relations, the descendents and neighbors of a symbol at a particular attachment point can be computed (see Gries [30] for an introduction to binary relations, their representation using matrices and their manipulation). The notation w R:aw,av v indicates that w is in relation R to v through endpoint aw of w and av of v. Given k attachment points per vocabulary symbol, the neighbor:i,j relation (which is equivalent to the sets Nei(v,a) discussed above) is computed by iterating the following computation n-1 times:

$$\text{neighbor:i,j} := \text{neighbor:i,j} + \sum \{\text{descendent:i,m} * [\sum (\text{neighbor:m,l} * \text{ancestor:l,j})]\}.$$

As an example, consider the house grammar given in Section 3.1. An ancestor matrix M_{ij} is a square matrix whose order is the number of vocabulary symbols in the grammar, and for which i specifies the attachment point of the vocabulary symbol of the row, and j specifies the attachment point of the vocabulary symbol of the column. Since there are two attachment points for each symbol of the grammar, there should be four matrices specifying the ancestor relation; however, as the attachment conditions and the attachment part generator are symmetric, all four ancestor matrices are equal:

```
0 0 1 0 0 0
0 0 0 0 0 0
```

The full neighbor relation includes the relation between <roof> and <wall> which was not directly represented in the grammar. Each row gives the set of vocabulary symbols which can possibly neighbor the symbol associated with that row.

3.2.2 Compiling Semantic Constraints

Semantic constraints can be generated in exactly the same way as syntactic constraints, i.e., by defining binary relations and compiling their transitive closure. This approach is analagous to the syntactic neighbor case; now a relation is defined between every two symbols whose semantic features are related and the closure contains relations not explicitly mentioned in the grammar.

As an example, consider the parallel relation. The parallel relation can occur between the axes of two vocabulary symbols in a variety of ways:

1) they can be explicitly defined as parallel in the semantic consistency part of a production, or

2) the semantic generation part of a production may set an axis of the new vocabulary symbol equal to an axis of one of the vocabulary symbols being used to produce the new symbol, or

3) they may may be indirectly parallel if there exists a third vocabulary symbol to which they are both parallel.

These relations are computed using a binary-valued matrix, whose rows and columns correspond to the axes of the vocabulary symbols.

a <floor> symbol. Since <floor> is a lower level symbol, all <floor> symbols will already have been built by the time <top> symbols are being built, and this can be used to delete <top> hypotheses which are not parallel to any <floor> hypotheses.

In order to add other semantic constraints, a matrix to represent the constraints is needed. The matrix can be computed from the grammar once the relation has been defined in terms of the predicates which appear in the productions. Parallel is a transitive relation, and other transitive relations can be computed in much the same way. Relations which are not transitive, e.g., perpendicular, require special procedures for their computation.

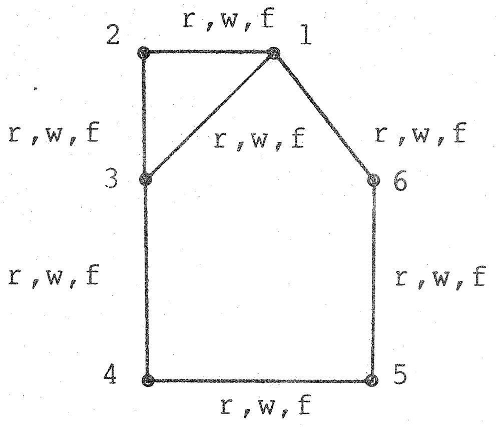
Some applications may prohibit the pre-compilation of all constraints, e.g., due to the size of the grammar. In such cases a possible alternative is to compute relations only when necessary. Of course, once the relations between the features of two vocabulary symbols have been computed, they can be stored for future reference and need not be recomputed every time.

corresponding to adjacent boundary segments are linked only if the constraints allow the symbols hypothesized for that pair to be adjacent. Building level 0 involves applying the segmentation strategy to the shape to generate the level 0 nodes.

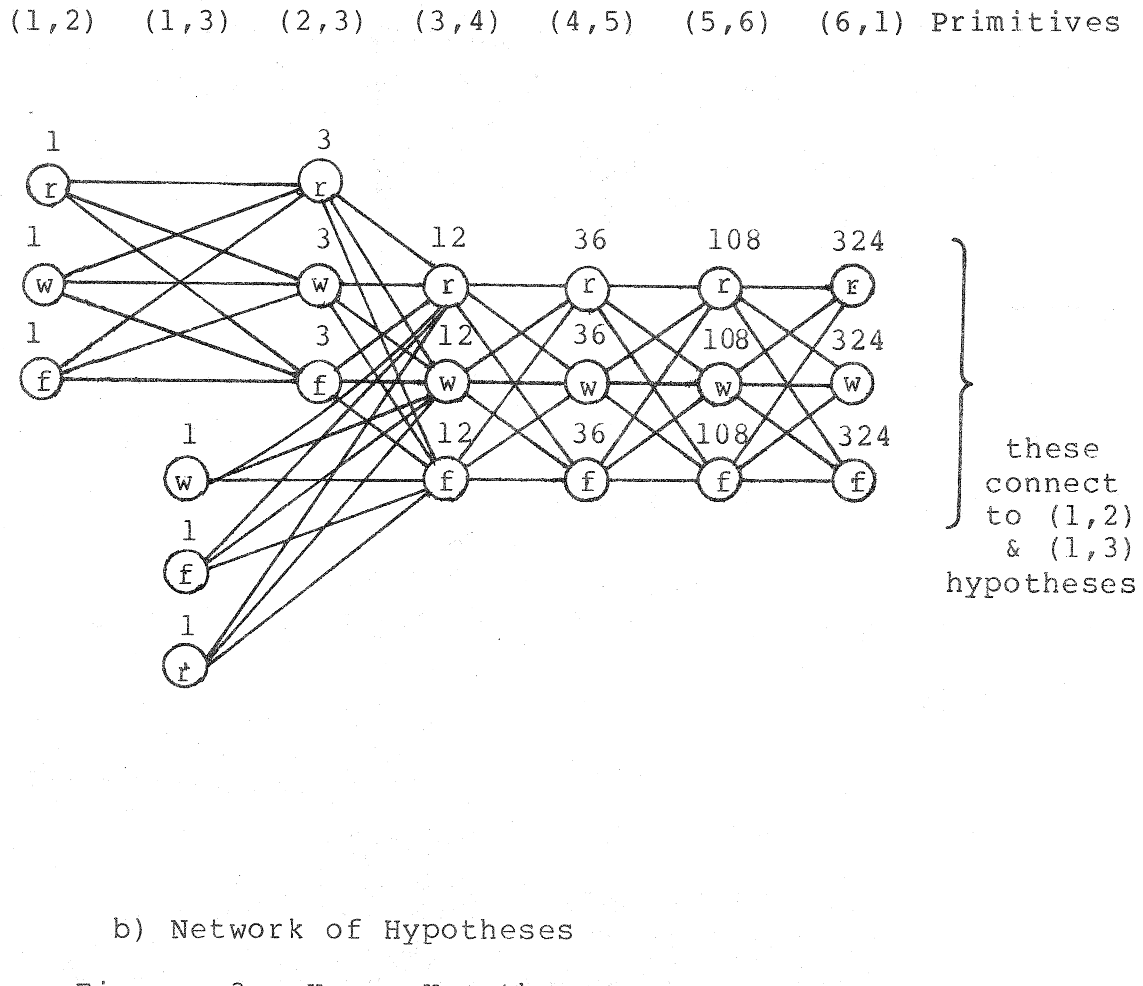
2) <u>CONSTRAIN</u> - since each node corresponds to a single hypothesis, and since nodes are only linked to compatible nodes, the within layer application of syntactic constraints simply involves removing a node if it has no neighbor at some endpoint. Likewise, a node is removed if for any semantic feature, no other node exists satisfying the semantic constraints.

3) <u>COMPACT</u> - given a node n at level k, if level k+1 has been built and n does not support a level k+1 node, then n is deleted from the network. If any of the nodes which produced n have been deleted, then n is deleted, too.

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. CONSTRAIN examines nodes from R_x. Let n be a node from R_x. If n satisfies all syntactic and semantic constraints, then CONSTRAIN simply deletes n from R_x. Otherwise, CONSTRAIN deletes node n from the network and puts its same level neighbors in R_x (since n might have been their only neighbor at some attachment point) and its across level neighbors in R_c. COMPACT removes nodes from R_c, taking no action if all of 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, COMPACT deletes the node from the network and puts its same level neighbors in R_x and its across level neighbors in R_c.



a) Primitives (1,2), (1,3), etc. and their hypotheses



b) Network of Hypotheses

Figure 3: House Hypotheses

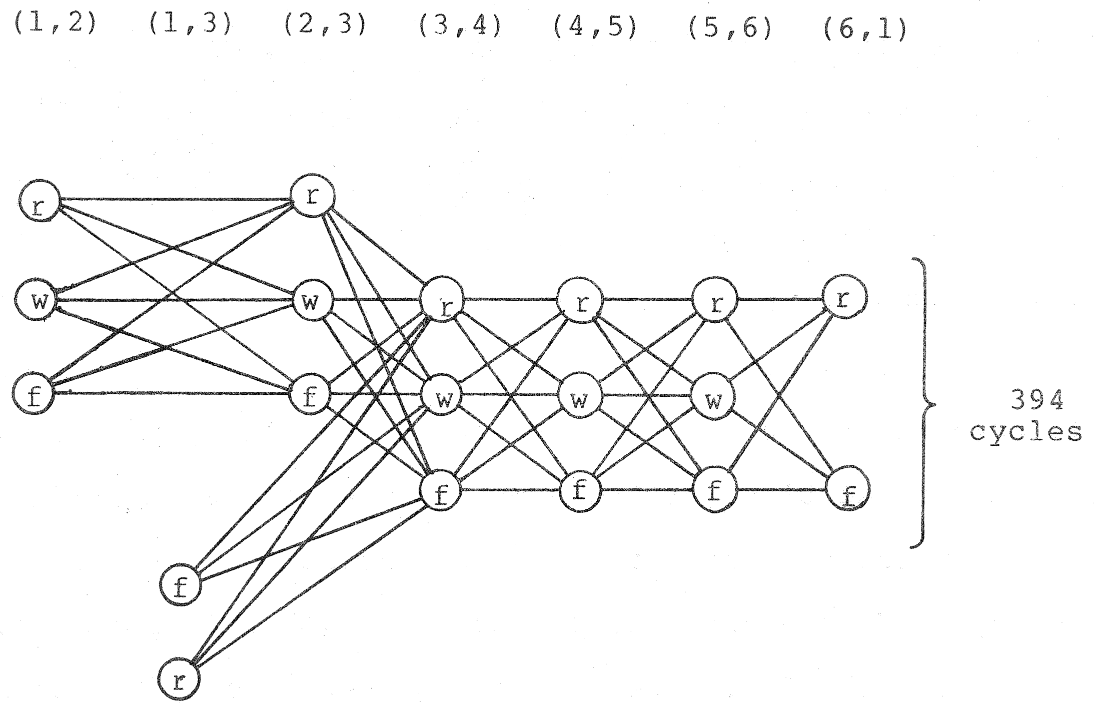


Figure 4: Hypotheses after Semantic Constraints

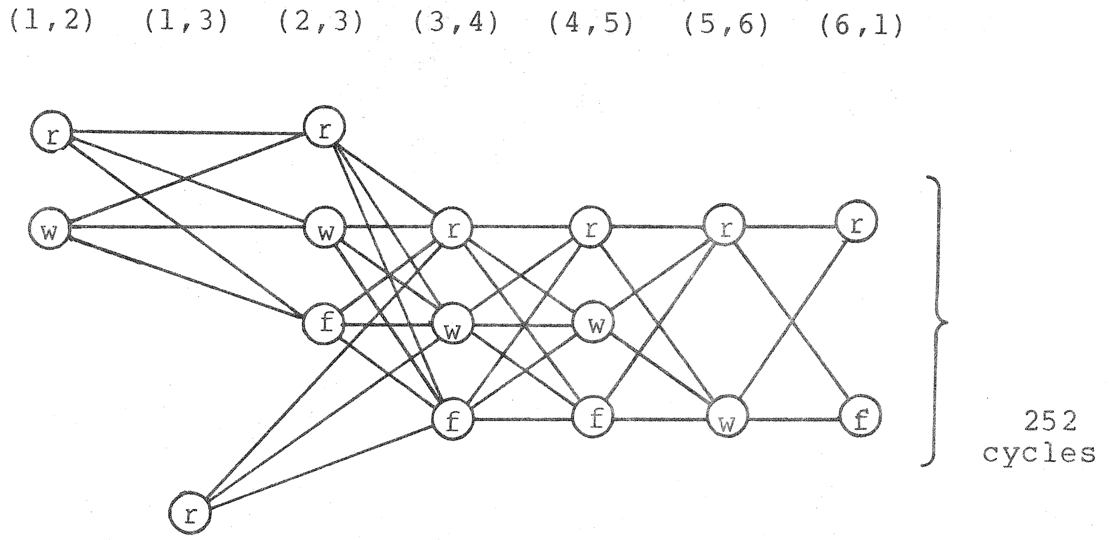


Figure 5: Hypotheses after Syntactic Constraints

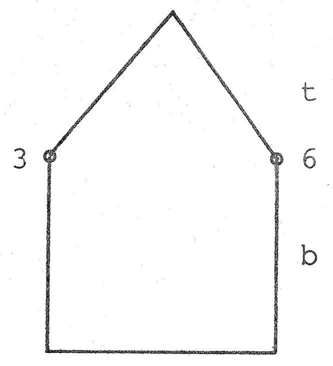


Figure 6: Level 1 of House

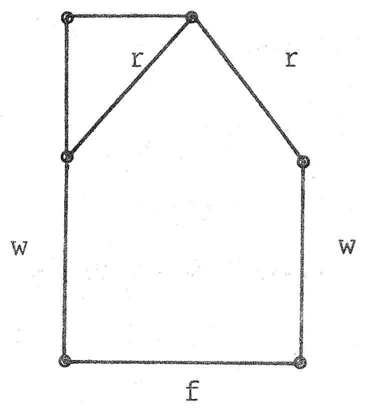


Figure 7: Final Level Ø Hypotheses

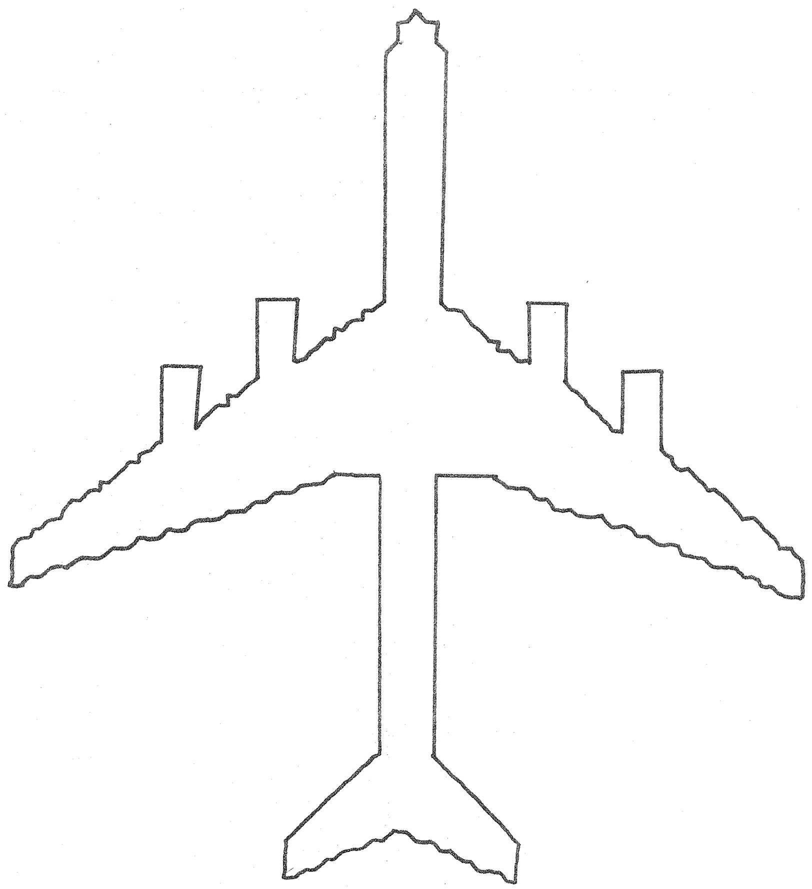


Figure 8a: Shape 1

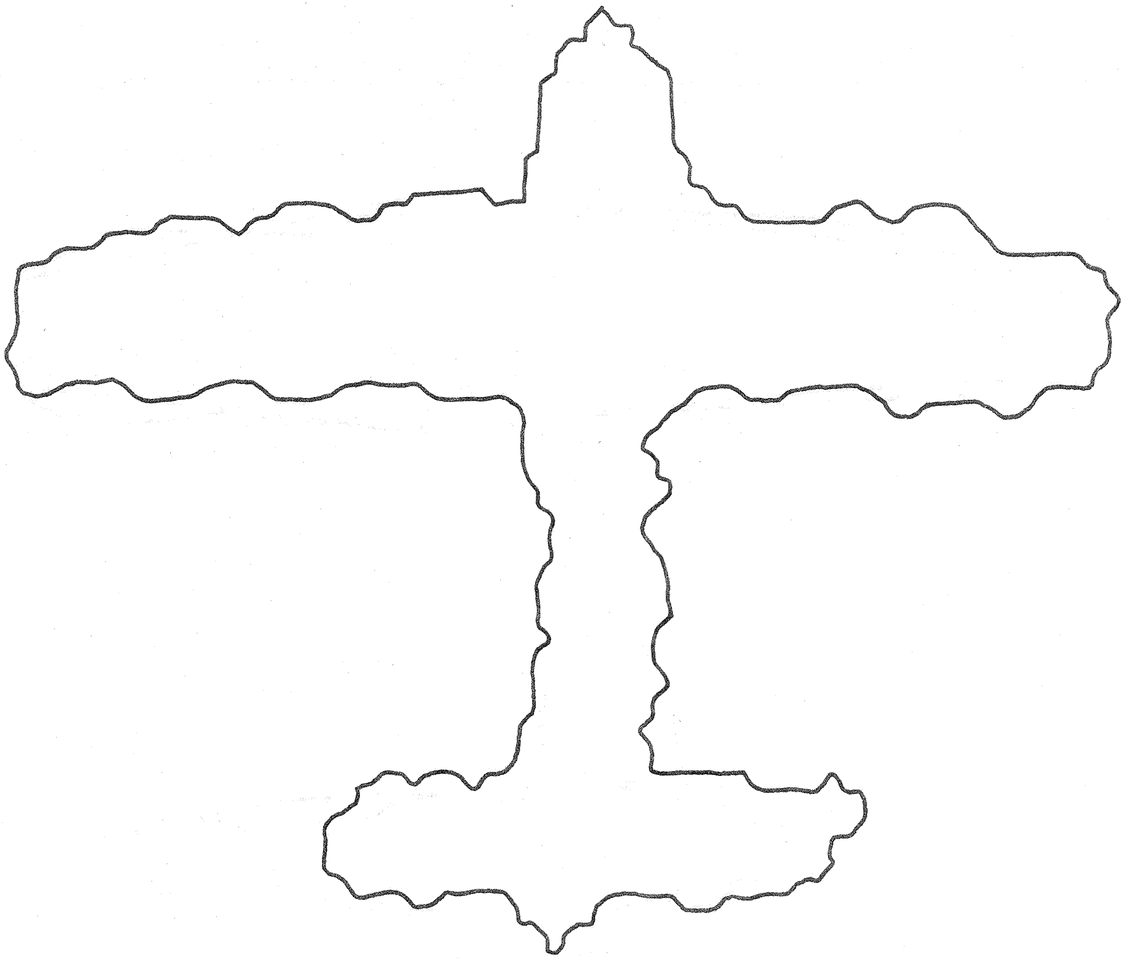


Figure 9a : Shape 2

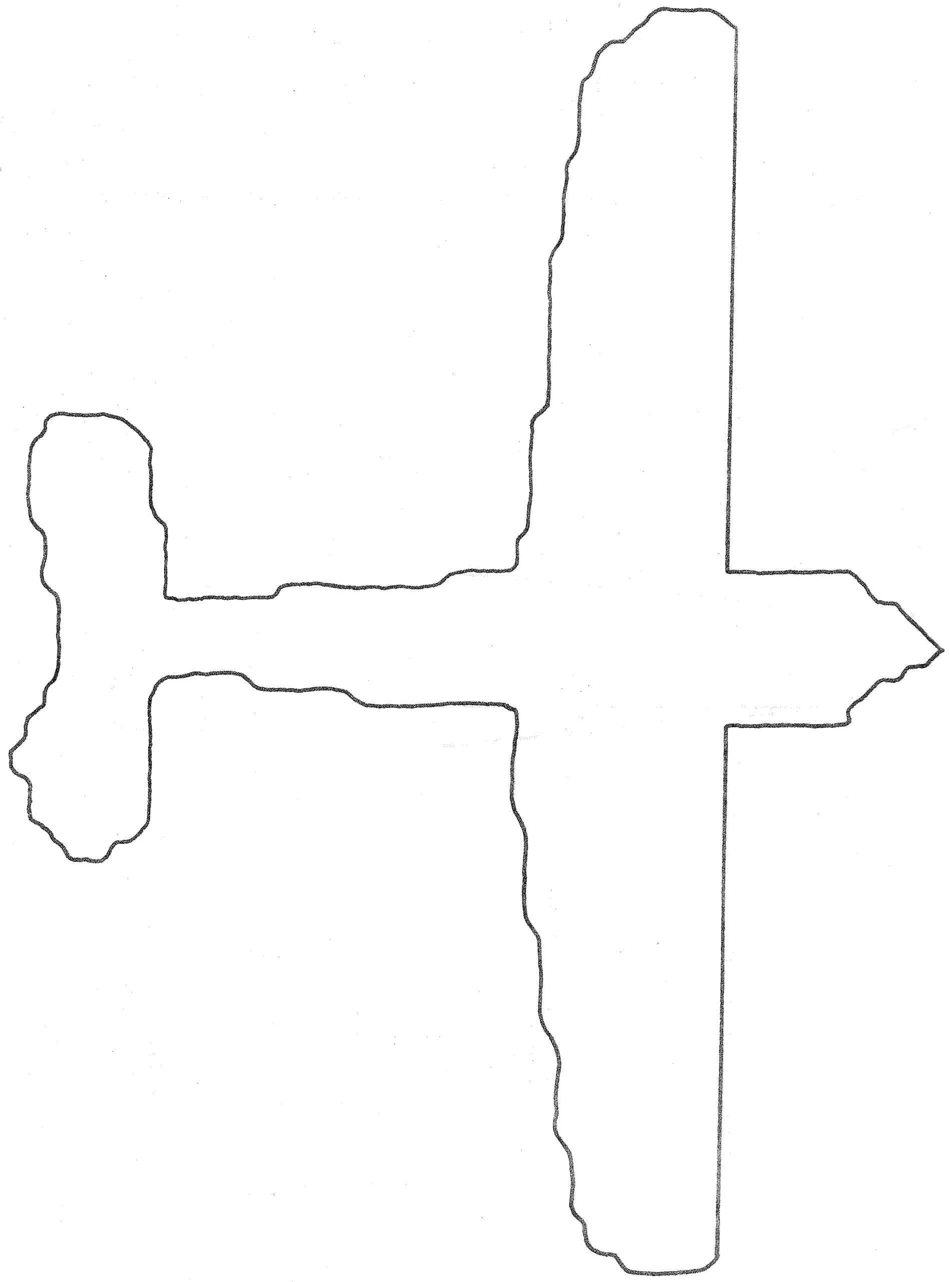


Figure 10a Shape 3

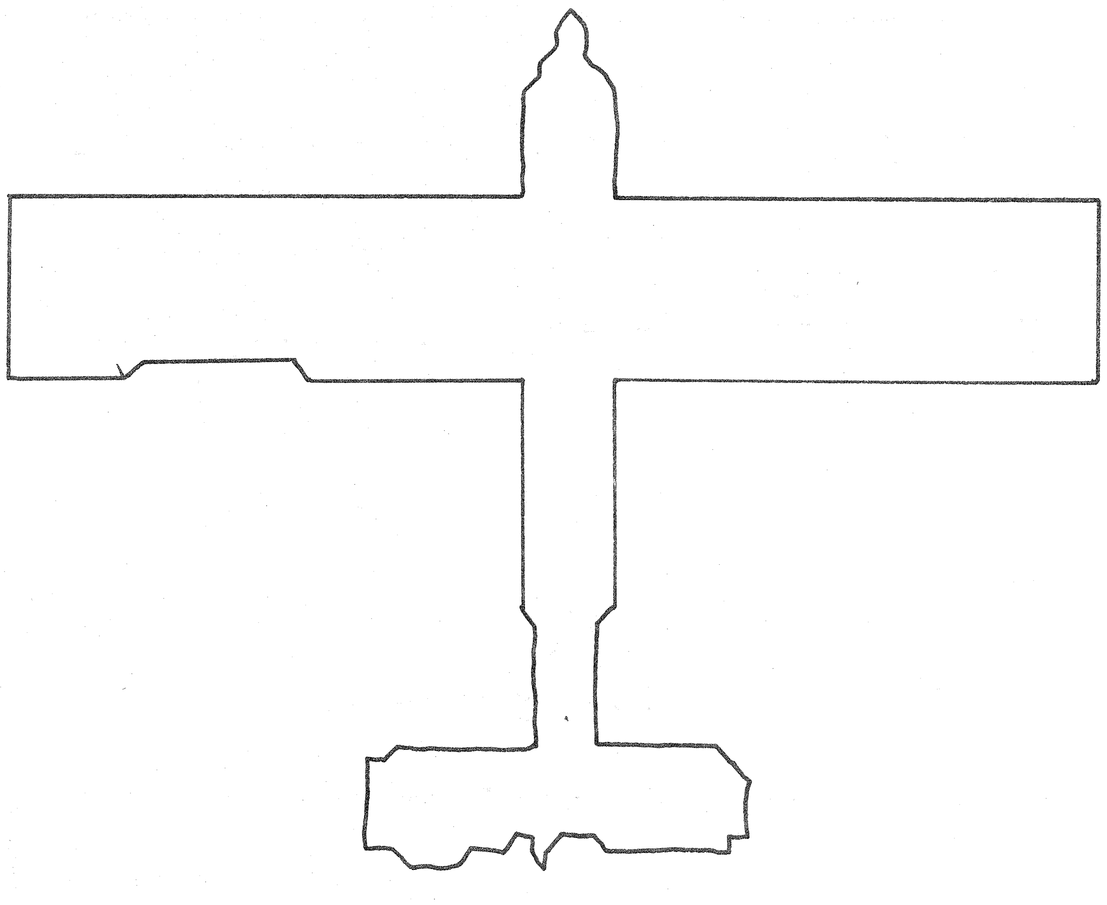


Figure 11a: Shape 4

pieces of the shape, while the loose fit picks out longer pieces.

Once the primitives have been found, the initial hypotheses for each primitive must be made. HCP was run with two different numbers of hypotheses per primitive. When only one hypothesis was associated with each primitive, the correct one was associated with each primitive that formed part of a grammatical description of the shape, and a "reasonable" hypothesis was chosen for each other primitive (e.g., if the primitive were short, then it might be labeled as an engine side). In the other experiment described, three hypotheses were associated with each primitive. In general, every terminal symbol should be associated with each primitive, unless some prior information on size or orientation is available which can eliminate some of those guesses.

For each of these sets of initial hypotheses, HCP was run with full constraints and with no constraints. Running HCP with no constraints means that procedure CONSTRAIN is not applied. A measure of efficiency was defined in terms of the number of hypotheses produced at each level versus the number of hypotheses actually necessary to parse the shape. Given a shape and a level, i, there is some fixed number of hypotheses, $N_a(i)$, which are required at that level to construct all parses of the shape. Let $N_0(i)$ be the number of nodes produced at level i when no constraints were used, and let $N_1(i)$ be the number of nodes produced at level i when the constraints were used. Then the efficiency of each process can be given as:

Table 1 - Node Efficiency with 1 Hypothesis per Primitive

	Node Level							
Shape	0	1	2	3	4	5	6	
1	.91	.94	.92	.93	.56	.83	1	(No constraints)
	.95	.94	.92	1	1	1	1	(All constraints)
2	.50	.50	.50	.45	.32	.50	1	
	.63	.63	.63	.90	1	1	1	
3	.59	.59	.59	.6	.5	.5	1	
	.70	.70	.70	.75	.75	1	1	
4	.76	.76	.76	.80	.60	1	1	
	.89	.89	.89	1	1	1	1	
Average	.69	.69	.69	.69	.49	.71	1	
	.79	.79	.79	.91	.94	1	1	

5. Using HCP with Uncertain Hypotheses

In the preceding discussions, all hypotheses of vocabulary symbols for shape segments were considered to be equally likely. In many situations though, some hypotheses should be regarded with more confidence than others. In what follows, we present a generalization of the discrete HCP described above to an HCP which associates likelihoods with hypotheses and applies continuous relaxation-like operators to update the likelihoods. We also discuss embedding HCP into a state-space search procedure for finding the most likely parse of a shape.

Let $G = (P,N,T,S)$ be a stratified context free grammar, and let $V = N \cup T$. A hypothesis consists of a vocabulary symbol and a likelihood. For hypothesis h, let $L(h)$ be the likelihood of h. If h is a level k+1 hypothesis formed from the level k hypotheses $h_1,\ldots,h_n$, then the likelihood of h is obtained as follows:

$$L(h) = \min\{ L(h_i) \},\ i = 1,\ldots,n.$$

Hypotheses relating terminal symbols to primitives are constructed if certain features of the primitive satisfy numerical constraints specified in the definition of the terminal symbol. For example, the length of a primitive might be measured, and if the length is found to be less than some value, then it may be possible for that primitive to play the role of an <engine side> in the current shape being processed. The degree to which these numerical constraints are satisfied determine the likelihood of the associated hypothesis.

(4) Expand n. Put the successors of n on OPEN. Remove n from OPEN.

(5) Go to 2.

It is shown in [32] that A+ is admissible and optimal.

We next describe how HCP can be embedded in the A+ algorithm. The nodes (or states) of the tree represent multi-layer networks of hypotheses. Nodes having start symbol hypotheses are terminal nodes. Each non-terminal node has either

1) one successor corresponding to the result of applying BUILD to the highest level hypotheses of that node, or

2) two successors: one representing the assertion of the most likely hypothesis (called the instantiation hypothesis) for a previously ambiguous piece of the boundary, and the other representing the denial of that assertion.

A level k hypothesis is asserted for a piece of the boundary if no other level k hypothesis which concerns that piece of the boundary is allowed to remain in the network. Likewise, a hypothesis is denied by being deleted from the network. The evaluation function used to order OPEN, f, is the maximum of the likelihoods of the highest level hypotheses of a state.

The constraints between pieces of a shape are no longer used simply to delete a hypothesis, but rather to change its likelihood. The likelihood of a hypothesis is dependent not only on the hypotheses which produced it, but also on the

(a) build the next level of the network for s, put the resulting node on OPEN and go to (3), otherwise

(b) disambiguate a piece of the boundary by instantiating the best hypothesis of s, i.e., generate a branch corresponding to asserting and denying of the instantiation hypothesis, setting up a new node for each.

(3) Apply CONSTRAIN, COMPACT and CONSTRAIN* to the new nodes.

(4) Evaluate the global score of each node by computing f, the maximum likelihood of the highest level hypotheses in the network for the node. (If all possible primitive hypotheses are deleted, set the score to 0).

(5) Update the likelihoods of the hypotheses associated with new nodes, and put the new nodes on OPEN.

(6) Go to (1).

HCP is the above algorithm with (2b) removed, and with likelihoods $\{0,1\}$.

We will now show that the application of such an operator during the search is admissible, i.e., the start symbol produced using HCP* is the same as the start symbol which is produced by using HCP and then choosing the most likely start symbol.

Let $H = \{h_{00}, h_{01}, \ldots, h_{0n}\}$ be the level 0 hypotheses for some shape, and let $S = \{S_1, S_2, \ldots, S_m\}$ be the start symbol hypotheses which can be constructed from H according to G. That is,

$$S_1 \to h_{11}h_{12}\ldots h_{1r1}$$

$$S_2 \to h_{21}h_{22}\ldots h_{2r2}$$

.
.
.

from OPEN containing a start symbol hypothesis $S_c \neq S_b$. Then there must be some node, m, on OPEN containing all of the hypotheses required to construct S_b up to level $k < n$ (node m could not yet contain S_b, since otherwise m would have been picked from OPEN). But from the Lemma, the likelihood of those hypotheses must be greater than or equal to $L(S_b)$, and thus, $f(m) \geq L(S_b)$. But $f(v) = L(S_c)$ because S_c is the only level n hypothesis in the network of node v. Thus, $L(S_c) = f(v) > f(m) = L(S_b)$, contradicting the assumption that $L(S_b) > L(S_c)$.

CONSTRAIN, COMPACT, and BUILD to the resulting layered network. Once HCP has stabilized on this network (all higher levels constructed and all constraints satisfied), the surviving lowest level hypotheses can serve to guide the search for still lower level, and probably even less reliably detected, pieces of the shape.

Many claims have been made [19-21] about the relative efficiency of constraint processes when compared with conventional search strategies, but very little effort has been devoted to substantiating or invalidating these claims (exceptions include Gaschnig[25] and Haralick and Gordon[26]). As another research goal, the computational complexity of HCP needs to be investigated by both analytical and empirical (e.g., simulation) studies on abstractions of the pattern analysis problem. Only through such studies can we hope to assess the real significance and practical importance of such systems.

in Syntactic Pattern Recognition Applications (ed. K.S. Fu), Springer-Verlag, Berlin, 1977, pp. 1-31.

13. Freeman, H., "On the Encoding of Arbitrary Configurations," IRE Trans. on Electronic Computers, Vol. EC-10, 1961, pp. 260-268.

14. Pavlidis, T., "Linguistic Analysis of Waveforms," Software Engineering, (ed. J. Tou), Academic Press, 1971, pp. 203-225.

15. Pavlidis, T., "Analysis of Set Relations," Pattern Recognition, Vol. 1, November 1968, pp. 165-178.

16. Pavlidis, T., "Representation of Figures by Labelled Graphs," Pattern Recognition, Vol. 4, 1972, pp. 5-17.

17. Pavlidis, T., "Structural Pattern Recognition: Primitive and Juxtaposition Relations," in Frontiers of Pattern Recognition, (ed. S. Watanabe), Academic Press, 1972, pp. 421-451.

18. Rosenfeld, A., R.A. Hummel and S. Zucker, "Scene Labeling by Relaxation Operations," IEEE Trans. on Systems, Man, and Cybernetics, Vol. SMC-6, 1976, pp. 420-433.

19. Haralick, R. and L. Shapiro, "The Consistent Labeling Problem," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-1, 2, April 1979, pp. 173-183.

20. Haralick, R, L. Davis, A. Rosenfeld and D. Milgram, "Reduction Operators for Constraint Satisfaction," Information Sciences, Vol. 14, 1978, pp. 199-219.

21. Mackworth, A.K., "Consistency in Networks of Relations," Artificial Intelligence, 8, 1977, pp. 99-118.

22. Gaschnig, J., "A Constraint Satisfaction Method for Inference Making," Proc. of the 12th Annual Allerton Conf. on Circuit and System Theory, October 2-4, U. of Illinois, 1974.

23. Ullmann, J.R., "An Algorithm for Subgraph Isomorphism," J. ACM, 23, 1976, pp. 31-42.