HIERARCHICAL CONSTRAINT PROCESSES FOR SHAPE ANALYSIS

Larry S. Davis

Thomas C. Henderson

November 1979

TR-115

THE UNIVERSITY OF TEXAS AT AUSTIN
DEPARTMENT OF COMPUTER SCIENCES
AUSTIN, TEXAS 78712

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.

The main thrust of this paper is to unite syntactic techniques and constraint propagation methods into one technique for analyzing two-dimensional shapes based on their boundaries. The terminal symbols of the stratified grammar will correspond to segments of the shapes being modeled. To analyze an actual shape, its boundary must be segmented into primitives which, ideally, correspond unambiguously and syntactically to particular terminal symbols of the grammar. In general, more primitives must be constructed than are actually required to parse the shape. Moreover, deciding which terminal symbols of the grammar correspond to a particular primitive is an ambiguous process - i.e., there may be several such choices for each primitive. In principle, every possible set of choices must be considered as a description of the shape. The number of hypotheses can be effectively reduced by constraint analysis; e.g., even though a terminal symbol may be a possible label for a particular primitive, there may not be any adjoining primitive whose hypotheses give the appropriate context for such a hypothesis. Such contextual constraints can be derived (or "compiled") automatically from a stratified grammar and used in the subsequent analysis.

The remainder of this paper discusses how this is achieved and is organized as follows: Section 2 reviews previous research relevant to the design of automatic shape analysis systems. In Section 3, we introduce a class of shape grammars, called stratified context-free shape grammars, which provide a strict hierarchical structure for vocabulary symbols. Both

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.

2. Related Research

Visual recognition of shape is an aspect of shape perception which has been widely investigated. The collection of psychological and psychophysical data is extensive and quite complicated and will not be treated here. However, it should be noted that in attempting to analyze or describe the structure of a shape, it may be convenient to extract parts of the shape in a manner similar to the way humans do. Extensive reviews of human shape perception include Zusne [4], Cornsweet [5], and Haber [6]. The reader is referred to them for a comprehensive introduction to the area. There is psychological evidence that the solidity of a figure is due essentially to the contour, and that transference of a learned task from a filled-in form to an outline figure and vice versa is excellent for most species [7]. Thus, shape analysis involves extracting information at various levels of detail. It is this informaton that will be organized instead of the raw image data itself. For example, Marr [8] has proposed using filtering operators to obtain a primal sketch. Recognition can then proceed in terms of this account of the image.

The solution to a shape analysis problem generally requires the design of shape modeling techniques and procedures for organizing unknown shapes according to such models. Once a shape modeling mechanism has been chosen, specific models for the classes of shapes to be analyzed can be constructed. This might involve simply determining values for specific shape features or, more generally, detailing the spatial organization

of a shape.

The process of choosing a modeling mechanism and then constructing shape models is complicated by a variety of factors that influence both the appearance of specific shapes in images and the segmentation of shapes of individual objects from images. These factors include:

(1) Geometric transformations including image plane transformations of rotation, scale and transformations of the object in its coordinate space which lead to perspective distortions. A shape may be arbitrarily oriented in the image plane, and, in general, this cannot be controlled.

(2) Partial information. Not only may a shape be imperfectly segmented, but major parts of the shape may be missing due to occlusion in the image, shadows, camouflage, etc.

(3) Agglomeration. Another segmentation related problem occurs when two or more shapes are segmented from an image as a single shape. A shape model should provide a description (or at least make such a description possible) for dissociating multiple shape objects.

(4) Noise. Noise is usually a global effect in an image and can influence the segmentation procedures. For example, edges that are actually smooth may appear jagged due to digitization noise.

Our discussion of shape models will be guided by the following definition:

A shape model consists of:

(1) a spatial decomposition of the shape,

(2) a description of the parts of the shape, and

(3) a description of the relations between the parts of the shape.

Shape models can be separated into two classes. When (1) is

the trivial decomposition into a single part, the model will be called a feature model. When (1) is non-trivial, the model will be called a structural model. A shortcoming of feature models is that they cannot ordinarily be applied in situations where only partial information is available about the shape. Feature models which have been extensively used in shape recognition include moment models [9] and Fourier models [10,11]. Structural, or syntactic models, on the other hand, can be used in situations where only partial information is available, and can also be used to decompose an agglomeration into several shapes.

Syntactic, or structural, pattern analysis consists of three major steps [12]:

(1) preprocessing - improving the quality of an image containing the shape, e.g., filtering, enhancement, etc. to facilitate segmentation,

(2) pattern representation - segmenting the shape and assigning the segments to the lowest level parts in the structural model, and

(3) syntax analysis - parsing the primitive shape parts according to the structural model.

Structural shape models describe the spatial decomposition of a shape, and consequently, must describe the primitive parts composing the shape. There are no established guidelines for choosing shape primitives; however, there are several desirable characteristics. Primitives should provide a compact description of the shape with little or no loss of information, and the extraction of shape primitives from a shape should be relatively simple using existing non-syntactic techniques.

Several classes of shape primitives have been proposed including chainlets [13] and more generally, boundary segments determined by piecewise functional approximations [14], both of which describe the boundary of a shape, and convex polygon decompositions [15-17] of the area of the shape.

The choice of primitives determines, to a large extent, the relations computed between the primitives. For piecewise linear approximations, relations such as adjacency, collinearity and symmetry may be computed. The exact relations computed are determined by the form of the structural model. There are, in practice, two kinds of structural model, one a special case of the other: single-level grammars, which are relational networks, and multi-level grammars. If relational networks are adopted as a representation, then shape analysis involves graph matching. Multi-level models require a parsing procedure.

This paper combines syntactic shape analysis with the constraint analysis tools used extensively in computer vision. In computer vision, constraint networks can be used in labeling a set of objects in such a way that a given set of constraints is satisfied [18-20].

Given a segmentation of a scene, the objects produced by the segmentation may not have obvious interpretations. Constraint analysis can be used to disambiguate the object interpretations. More precisely, a class of scenes can be modeled by specifying constraints (for example, on relative

position) between objects in such scenes. These constraints can be used to disambiguate objects in an unknown scene by ruling out interpretations which do not satisfy all (or some) of the required constraints with the assumed interpretations of other objects in the scene. Finding only the interpretations satisfying the constraints is the scene labeling problem. In [18], Rosenfeld et al introduce parallel, iterative algorithms, (both discrete and continuous) called relaxation procedures for enforcing local constraints.

Mackworth [21] discusses issues relating to search efficiency using constraint analysis. Gaschnig [22] has described a general constraint satisfaction method and has shown how it can be used to solve certain classes of state space search problems. For sufficiently constrained problems, the method can eliminate locally inconsistent assignments and avoid search redundancy. Variations of discrete relaxation have been proposed by Ullmann [23,24] for finding subgraph isomorphisms and for pattern association.

It is hoped that using constraint reduction analysis will increase the speed and efficiency of problem solving. Claims have been made that network consistency algorithms are better than more standard procedures, e.g., backtracking. However, Gaschnig [25] presents experimental case studies comparing several algorithms, including network consistency algorithms and backtrack algorithms, and concludes that for the particular problem studied (the N queens problem), the backtracking algorithm was the most efficient in terms of the performance

criteria chosen. Somewhat different results are reported by Haralick and Gordon [26].

Davis [3] has described a method for incorporating the use of constraint analysis with a hierarchical representation of objects. In particular, he suggested that a hierarchical relaxation process be designed which could apply the constraints implicit in a layered network of relational models. The process was demonstrated in the context of simple one-dimensional waveform analysis. The present paper is an extension of that work (see also [27]) to two-dimensional shape analysis.

3. Hierarchical Constraint Processes

This section discusses hierarchical constraint processes. They represent a union of structural techniques and constraint analysis techniques. In order to apply a hierarchical constraint process to the analysis of shape, a stratified shape grammar must be defined for the class of shapes under consideration (Section 3.1). Once a shape grammar has been specified, the syntactic and semantic constraints which are implicit in the grammar are compiled from the grammar (Section 3.2). Then, the hierarchical constraint process can be applied to the analysis of unknown shapes (Section 3.3). The hierarchical constraint process takes as input the compiled version of the grammar and the primitives obtained from the segmentation of an unknown shape and produces as output a layered network of hypotheses. The network may be empty, representing the assertion that the unknown shape was not from the class of shapes defined by the grammar, or may contain alternative parses of the shape.

In the following subsections, a grammar for a simple house shape will be used to illustrate the hierarchical constraint process. Section 4 describes an application to airplane recognition.

3.1 The Grammatical Shape Model

With some simple modifications, syntactic shape models can be integrated with constraint analysis techniques. The shape grammars which we will consider are an extension of the geometrical grammars suggested by Vamos [28] and Gallo [29].

We define a stratified context-free grammar, G, as a 4-tuple (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 U T) be the set of vocabulary symbols.

Associated with every symbol $v \in V$ is a level number, ln(v). For each terminal symbol v, ln(v) = 0. The set of terminal symbols corresponds to the smallest segments of the shapes modeled by the grammar, e.g., short straight-edges of the shape boundary.

Each non-terminal symbol has a level number from 1 to n associated with it. A start symbol has level number n, and for any rule $v = v_1 v_2 \ldots v_r$, if $ln(v) = k$, $1 \leq k \leq n$, then $ln(v_i) = k-1$, $i = 1,\ldots,r$. Unlike conventional string grammars, vocabulary symbols have a non-trivial structure. A vocabulary symbol v is composed of a <name part>, {attachment part} and a [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, which are required to specify how the symbol can be combined with other symbols to form higher level symbols, and

c) [semantic part] is a set of predicates which describe certain properties of the symbol, such as its axis, length, etc.

Each production in the grammar is of the form ($v:=v_1v_2...v_r, A, C, G_a, G_s$), where

a) $v:=v_1v_2...v_r$ is the rewrite part which indicates that the symbol v is constructed from the group of symbols $v_1v_2...v_r$.

b) A is an applicability condition on the syntactic arrangement of the v_i. A specifies how the various endpoints of the v_i must be attached in order to produce a v.

c) C contains semantic constraints on the v_i, and consists of various predicates describing geometric and other properties and relations of the v_i which must be satisfied in order to produce a v.

d) G_a contains rules for generating the attachment part for v, and

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

As an example of a production, consider how engines are formed (see Figure 1) in a grammar for an airplane.

```
< engine > { e1,e2 } [ a,span ] :=
          < engine side > { e1',e2' } [ a' ] +
          < engine front > { e1'',e2'' } [ a'' ] +
          < engine side > { e1''',e2''' } [ a''' ]

A : [ Join(e1' or e2',e1'') and Join(e1''' or e2''',e2'')
   or Join(e1''' or e2''',e1'') and Join(e1' or e2',e2'') ]

C : [ Parallel(a',a''') and Length(a')=Length(a''')
```

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

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

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 (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

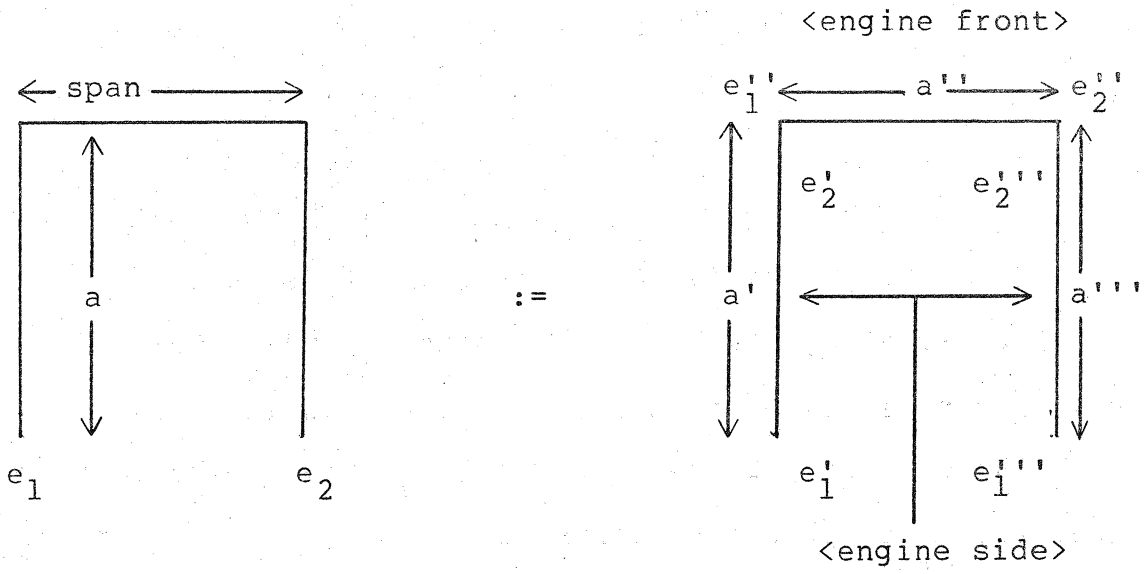


Figure 1 : Example of a Production

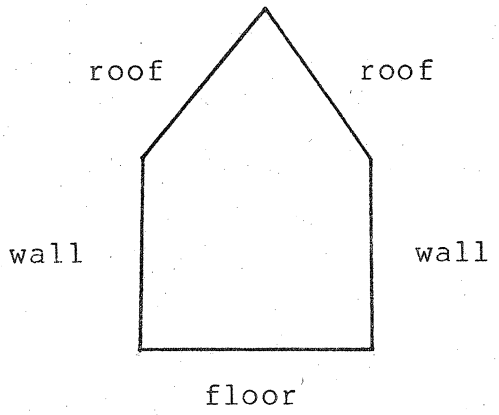


Figure 2: Typical House Shape

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")]
```

(production 3) :

<house> { } [a] :=
<top> {e1',e2'} [a'] +
<bottom> {e1",e2"} [a"]
a : [Joined(e1',e1") and Joined(e2',e2") or
Joined(e1',e2") and Joined(e2',e1")]
c: [Equal(a',a") and Parallel(a',a")]
G_a : [ ]
G_s : [Set(a,a')]}.

Figure 2 shows a typical example of the shapes described by G.

Stratified grammars contain a large set of implicit contextual constraints on the organization of a shape. It is these constraints which the hierarchical constraint process will utilize to analyze unknown shapes.

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.

Semantic constraints correspond to relations between the semantic features of vocabulary symbols. For example, it can be determined from the grammar for airplanes that the main axis of a <plane> is parallel to the main axis of an <engine>. Semantic constraints facilitate the incorporation of high level information into the analysis of a shape - e.g., if the orientation of the fuselage of a plane is known (possibly due to prior image analysis which has discovered runways), then this information can be automatically compiled into constraints on the orientation of the wings through the productions of the grammar.

3.2.1 Compiling Syntactic Constraints

Let G = (T,N,P,S), and let v, w and x $\in$ V. Let at(v) denote the set of attachment points of v, and let av $\in$ at(v). We define the three binary relations:

1) v ancestor:av,aw w iff there exists a production p such that the rewrite rule of p is v := ...w... and there exists an aw $\in$ at(w) such that aw is identified with av in G_a of p. That is, the attachment point aw of the right-hand side symbol, w, becomes the attachment point av of the left-hand side symbol v. For example, in Figure 1 the attachment points for the symbol "engine" are associated with the unjoined attachment points of the "engine side" symbols.

2) w descendent:aw,av v iff v ancestor:av,aw w.

3) v neighbor:av,aw w iff

a) there exists a production p such that the rewrite rule of p is x := ...v...w... and aw is specified as being joined to av in the applicability conditions, A, of p, or

b) there exists x $\in$ V with ax $\in$ at(x), and y $\in$ V with

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:

$$A = A_{11} = A_{12} = A_{21} = A_{22} = \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

where the rows and columns 1 through 6 correspond to the vocabulary symbols <roof>, <wall>, <floor>, <top>, <bottom> and <house>, respectively. Note that the <house> symbol could be left out as it has no attachment points. The descendent matrix D_{ij} is just the transpose of A_{ij}:

$$D = D_{11} = D_{12} = D_{21} = D_{22} = \begin{matrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} .$$

Finally, the explicit neighbor relation is given as:

$$N = N_{11} = N_{12} = N_{21} = N_{22} = \begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} ,$$

which is determined by the grammar. The full neighbor relation is computed as:

$$N = N + DNA = \begin{matrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{matrix}$$

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.

Again consider the grammar for the simple house shape. Let the matrix P denote the parallel relation between the axes of the vocabulary symbols (ordered as in Section 3.2.1). Since only one axis exists per vocabulary symbol, only one matrix is necessary to define the parallel relation. The matrix given directly by the grammar is:

$$P' = \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{matrix} .$$

The transitive closure of P' is:

$$P = \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{matrix} ,$$

except that elements on the diagonal of P' remain unaltered, where $P = (P' \vee \neg I) \wedge P'!$, where I is the Boolean identity matrix and P'! is the transitive closure of P'. This must be done since by transitivity a vocabulary symbol would be parallel to itself through any other element to which it is parallel. For example, any <wall> hypothesis must have a distinct <wall> hypothesis parallel to it. However, this constraint was already explicit in the productions; a less obvious constraint is that any <top> symbol must be parallel to

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.

3.3 The Hierarchical Constraint Process

As discussed previously, a major problem associated with applying syntactic techniques to shape analysis is segmenting a shape into pieces which correspond to the terminal symbols of the grammar required to parse the shape. In general, in order to obtain all the required segments, the segmentation procedures must have a high false alarm rate. This results in a large search space for parses of the shape. To overcome this problem, a hierarchical constraint process (HCP) 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.

The hierarchical constraint process computes a bottom-up parse of the shape by applying the constraints to a network of low-level hypotheses about the pieces of the shape and constructing a layered network of hypotheses containing alternative parses of the shape. The processing of this network can be easily described by specifying three simple procedures and two sets which these procedures manipulate. The three procedures are:

1) BUILD - given level k of the network, BUILD uses the productions of the grammar to construct nodes corresponding to 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 supported by them. After all nodes are generated, node pairs

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) CONSTRAIN - 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.

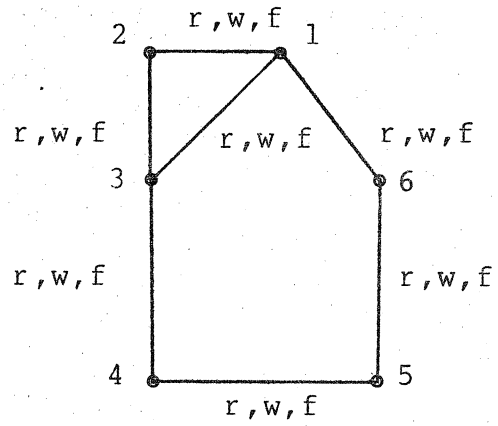
3) COMPACT - given a node n at level k, if level k+1 has been built and n does not support a level k+1 node, then n is deleted from the network. If any of the nodes which produced n have been deleted, then n is deleted, too.

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.

HCP does not eliminate any hypothesis which can be part of a complete parse. This can be seen as follows: BUILD simply generates the next level symbols, and if used without CONSTRAIN and COMPACT, will produce all possible hypotheses at every level. CONSTRAIN deletes a hypothesis h only if one end point of h has no neighboring hypothesis which can be joined to it, or if it fails to satisfy a semantic constraint. Thus, either h cannot be used to build a higher level symbol, or any symbol which h can be used to build will lack appropriate context at the next higher level. As for COMPACT, there are two cases to consider. First, if a level k hypothesis is not used to produce any level k+1 hypothesis, then due to the stratification, that level k hypothesis will never produce any higher level hypothesis; thus, it cannot be part of a parse. Finally, if a level k hypothesis h loses the support of one or more of the hypotheses which produced it, then it cannot be part of a complete parse because if it were, then its supporting nodes would be, too.

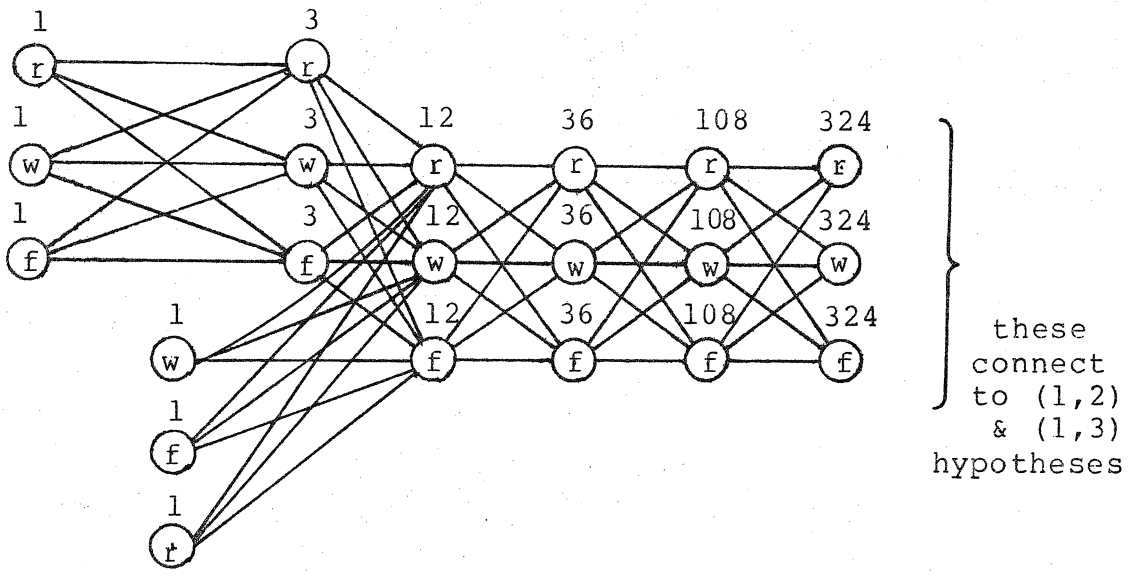
To illustrate the application of HCP, consider once more the house grammar. Suppose the level 0 hypotheses for the 7 primitives are as given in Figure 3a, where r, w, f, t, b, and h stand for <roof>, <wall>, <floor>, <top>, <bottom> and <house>, respectively, and every possible hypothesis has been made for each primitive. The syntactic and semantic constraints are as given in Sections 3.2 and 3.3.

Figure 3b shows the network of hypotheses as an adjacency graph. The nodes correspond to level 0 hypotheses, and given



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

(1,2) (1,3) (2,3) (3,4) (4,5) (5,6) (6,1) Primitives



b) Network of Hypotheses

Figure 3: House Hypotheses

two nodes, n_1 and n_2, n_1 is connected by a directed arc to n_2 if the primitive corresponding to n_1 in the shape "precedes" and is adjacent to the primitive corresponding to n_2 (precedes is well defined if we adopt a specific sense for following the shape boundary). The number next to each node is the number of distinct paths (from left to right) to that node from a left-most node. This number is computed for nodes from left to right by assigning the left-most nodes a value of 1, and continuing to the right assigning to each node the sum of those nodes immediately to the left and adjacent to it. Each right-most node connects to each left-most node for a closed shape. The goal of HCP is to find all the cycles in the network that correspond to shapes in the language defined by the grammar.

HCP begins by applying CONSTRAIN to the network of hypotheses. All the syntactic constraints are satisfied since all primitives have been labeled with all level 0 vocabulary symbols. However, the <wall> hypotheses of primitives (1,3) and (6,1) are deleted since neither of them has a parallel <wall> hypothesis, thus failing to satisfy a semantic constraint. The result is shown in Figure 4. The syntactic constraints now cause several hypotheses to be deleted: the <floor> hypotheses are deleted from (1,2), (1,3), and (5,6) since one attachment point fails to have a neighboring <wall> hypothesis. Figure 5 shows the resulting network.

At this point, all syntactic and semantic constraints are satisfied, and all possible level 1 vocabulary symbols can be

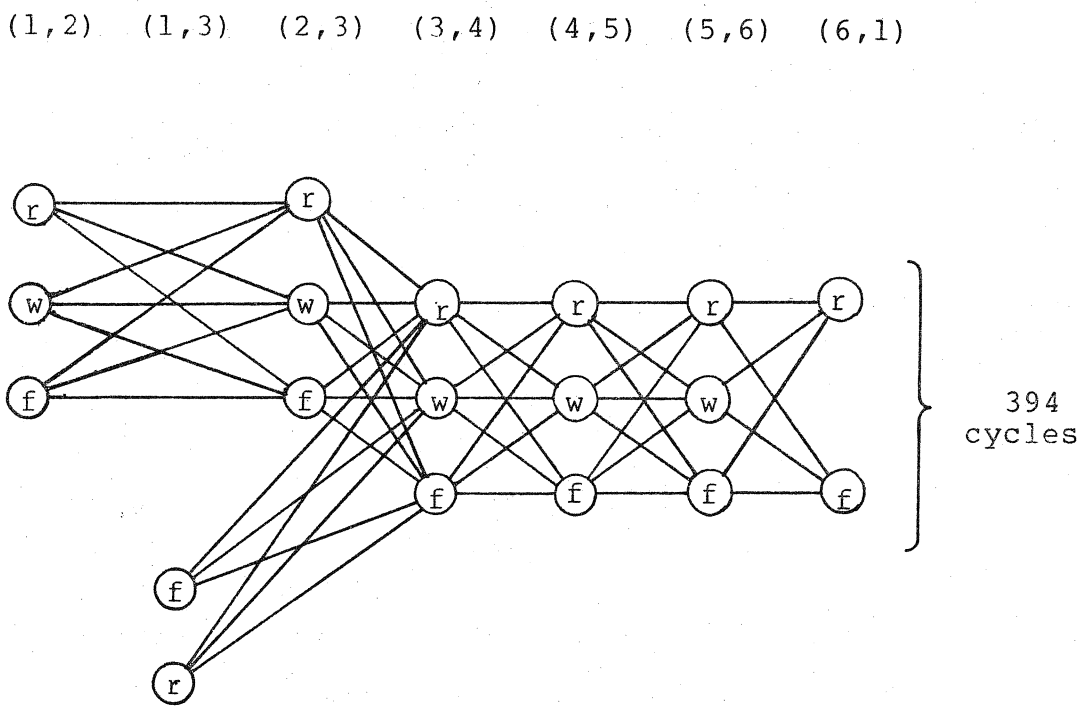


Figure 4: Hypotheses after Semantic Constraints

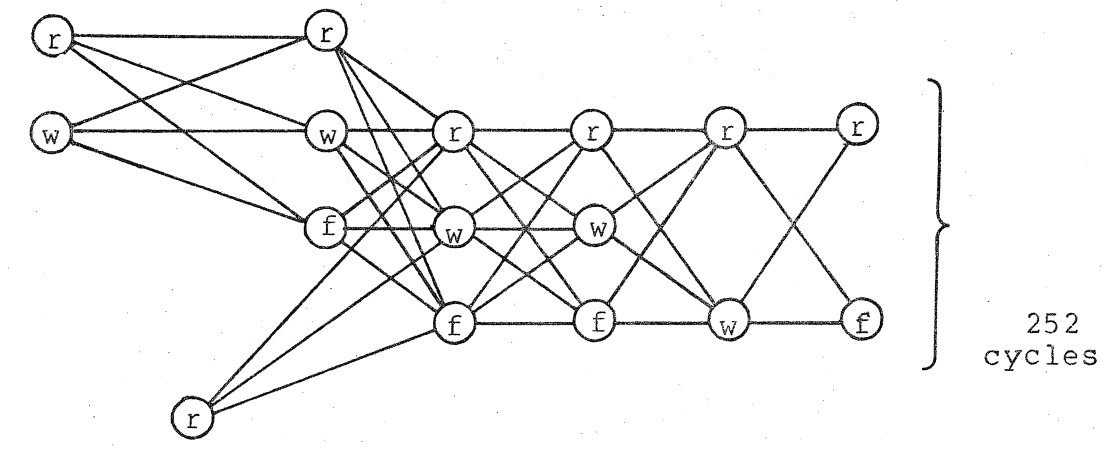


Figure 5: Hypotheses after Syntactic Constraints

built. Each hypothesis is checked against all applicable productions in building the next level. For example, the primitives in Figure 5 can be combined in the following way: the (1,2) <roof> hypothesis can be joined with the (2,3) <roof> hypothesis to form a <top> hypothesis; however, the axis of the resulting <top> hypothesis has no <floor> hypothesis parallel to it, and this <top> hypothesis is therefore discarded. The (1,2) <roof> hypothesis and the (1,6) <roof> hypothesis, although connected properly, fail the semantic consistency part of the production as they are not perpendicular. The other level 0 hypotheses are matched to productions in a similar manner, and the resulting level 1 hypotheses are shown in Figure 6. COMPACT is now applied to the level 0 hypotheses, and the result is shown in Figure 7. At this point, all incorrect hypotheses have been eliminated, and the level 2 start symbol <house> is directly built. Even this simple example shows the advantage of using syntactic and semantic constraints.

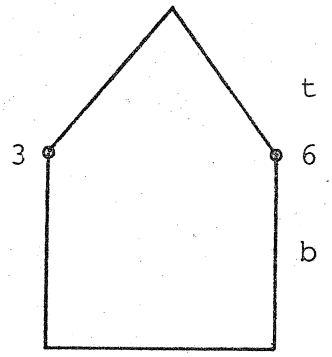


Figure 6: Level 1 of House

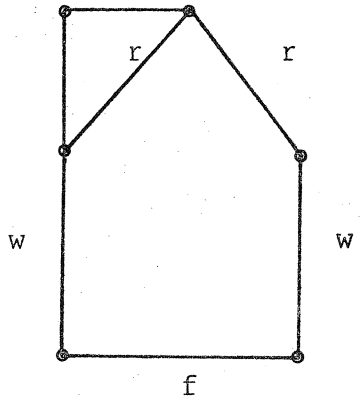


Figure 7: Final Level Ø Hypotheses

4. Experiments

A set of PASCAL programs implementing HCP have been developed at the University of Texas. Input to HCP consists of a stratified shape grammar defining the class of shapes to be analyzed and a set of primitives computed from a shape to be analyzed. The primitives are a set of line segments whose descriptions provide information including orientation, length and endpoints. HCP 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 complete 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 37 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 will describe the application of HRP to the top view of airplanes. The shapes used in this study were obtained from the literature (shape 1 [31]) and from model airplanes (shapes 2 through 4). Figures 8-11 display all of the shapes.

The split-and-merge algorithm [14]was used to obtain piecewise linear approximations to the shape. The algorithm was applied at several thresholds of goodness of fit. For these shapes two thresholds were used, i.e., both a close fit and a loose fit were obtained. The close fit picks out small

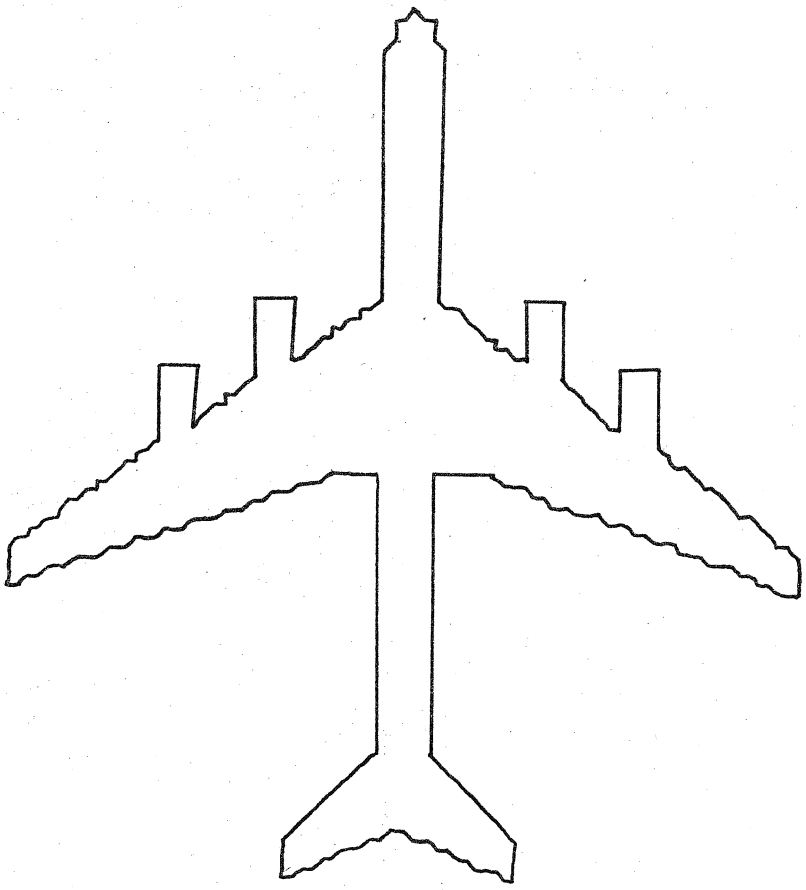


Figure 8â: Shape 1

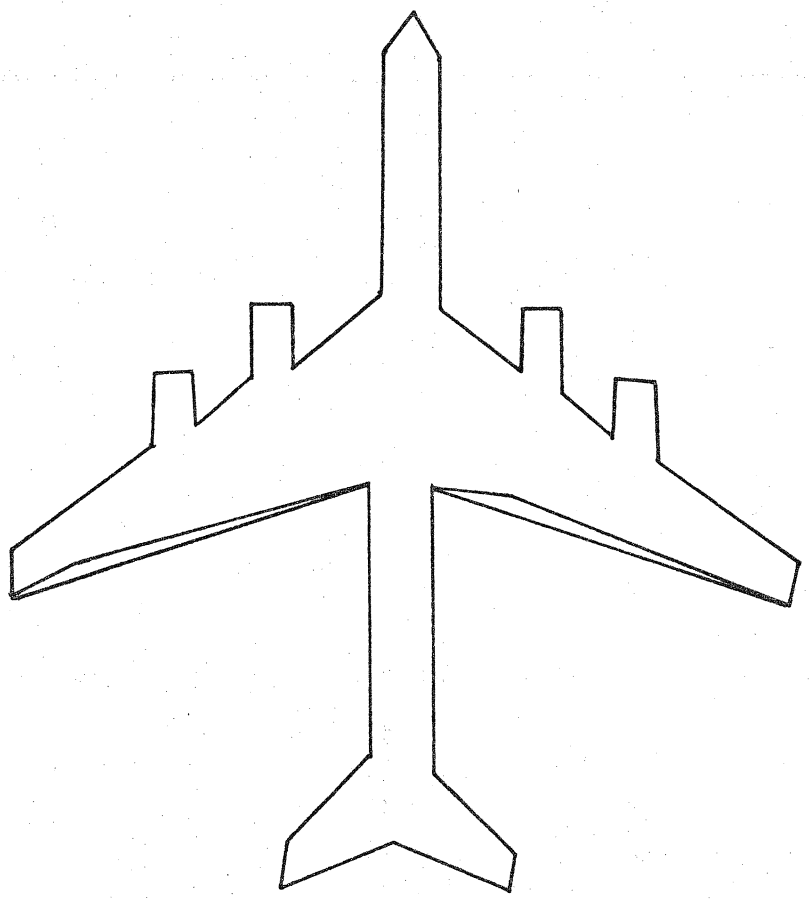


Figure 8b: Segmentation of Shape 1

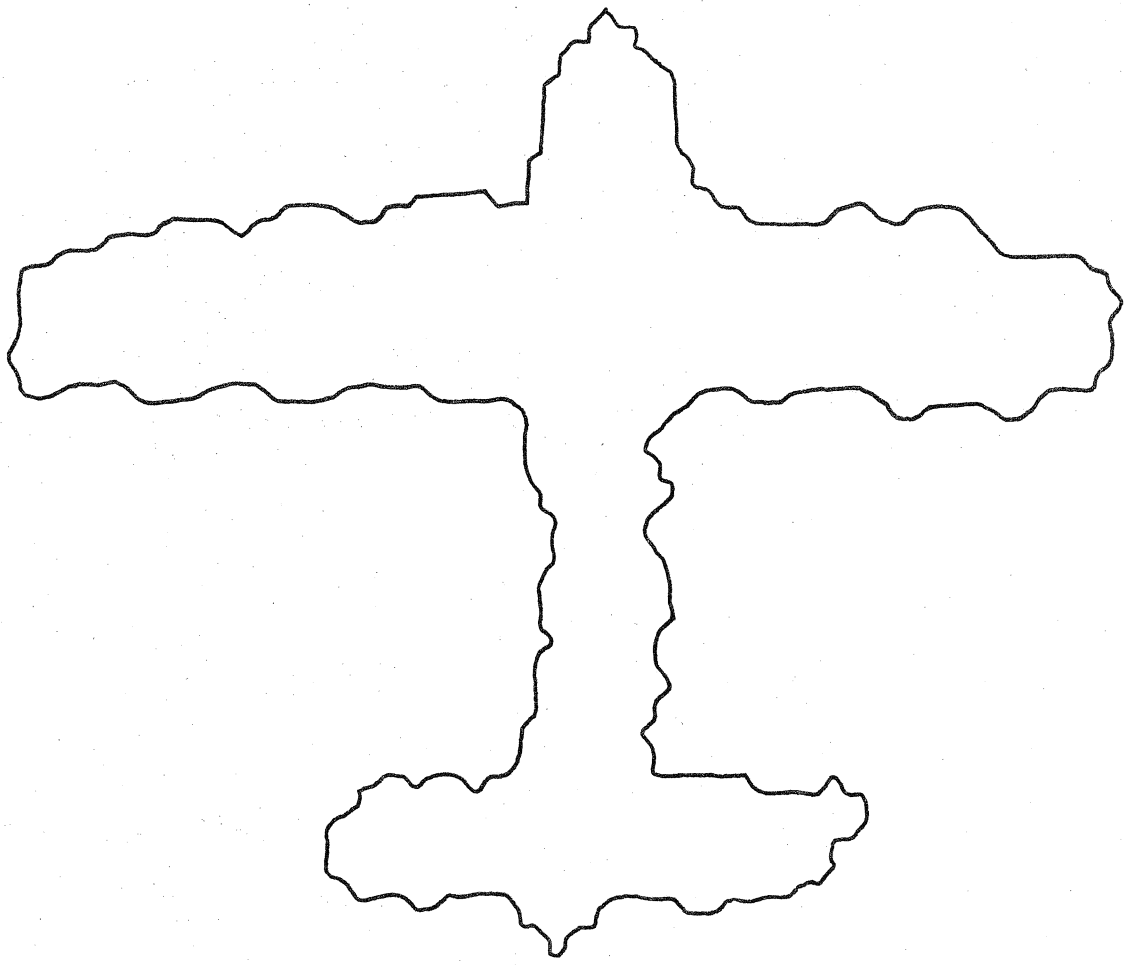


Figure 9a : Shape 2

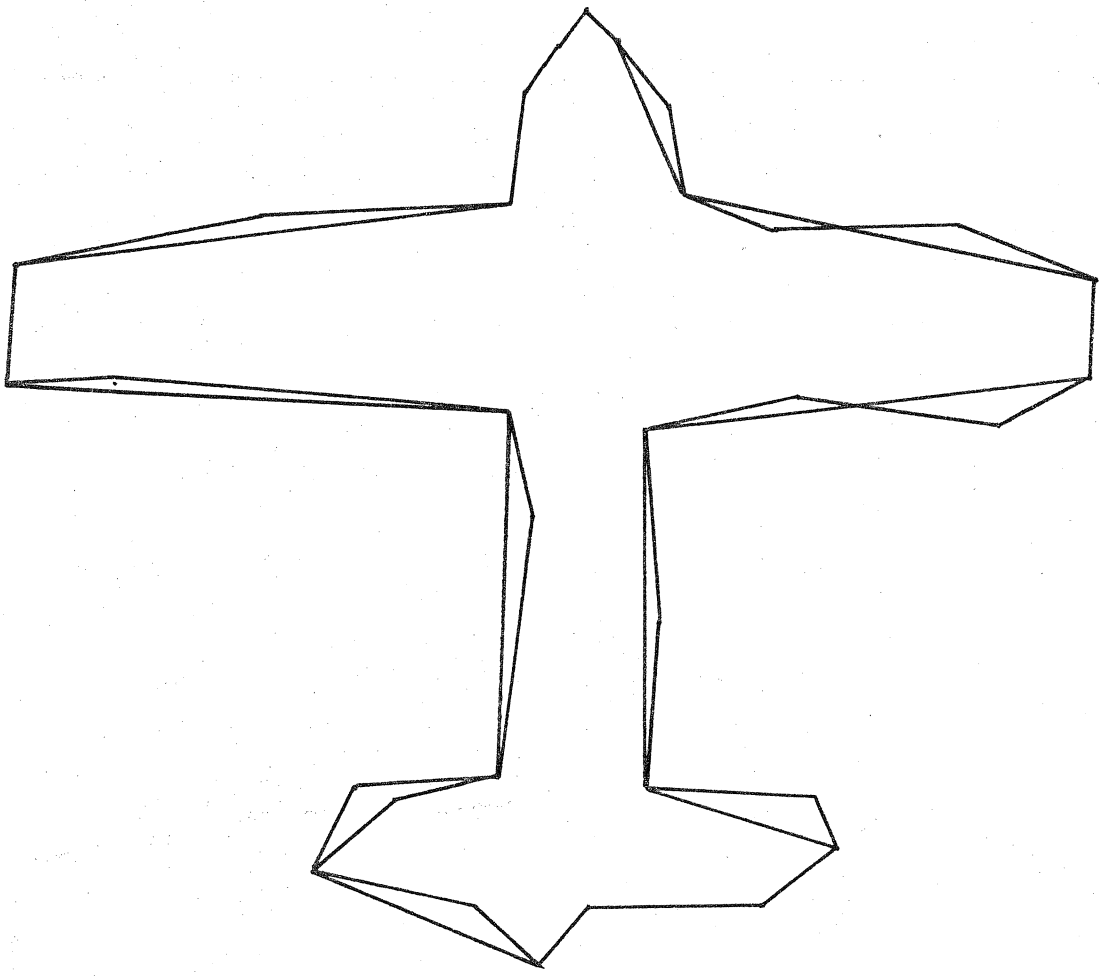


Figure 9b : Segmentation of Shape 2

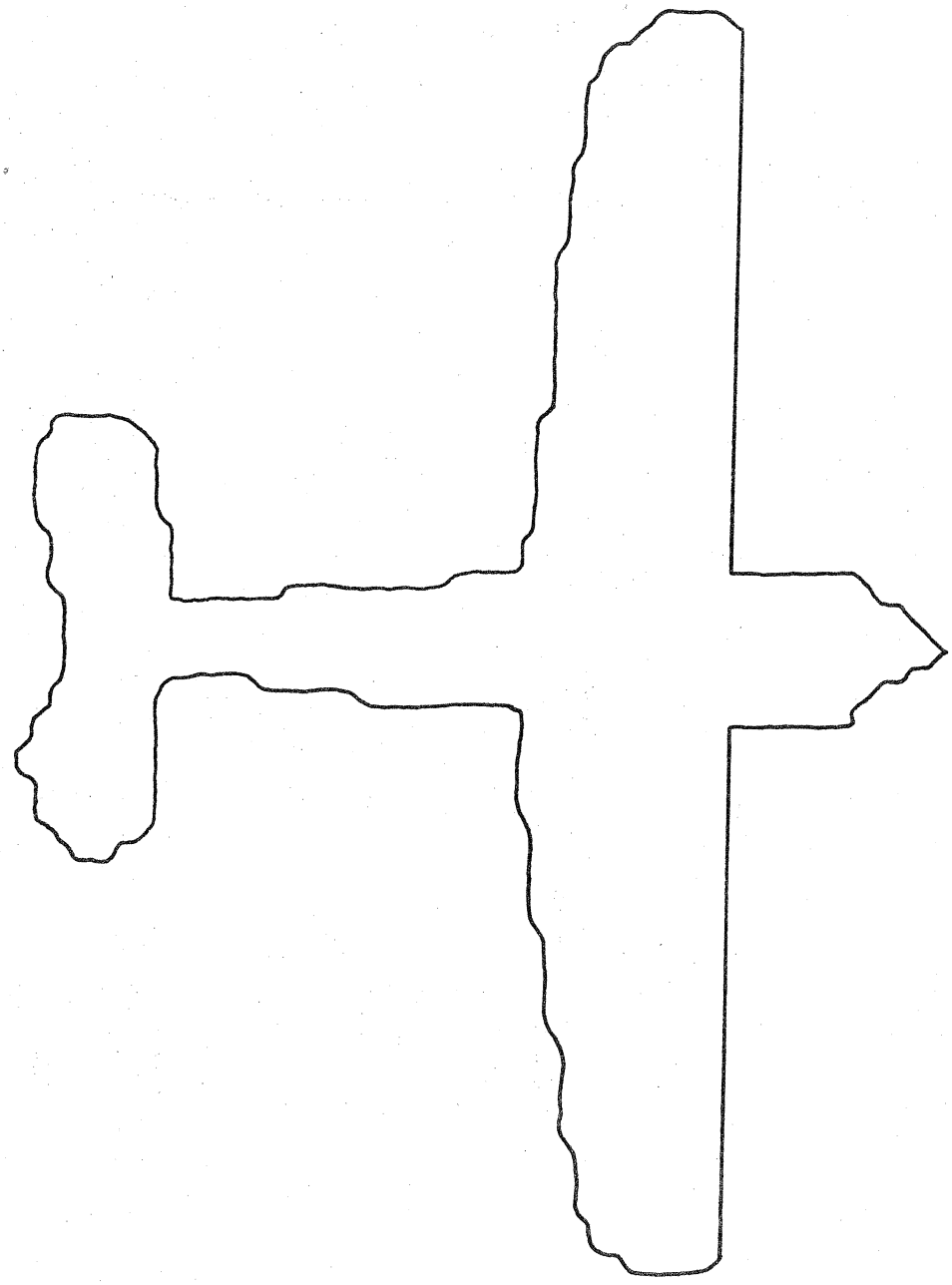


Figure 10a Shape 3

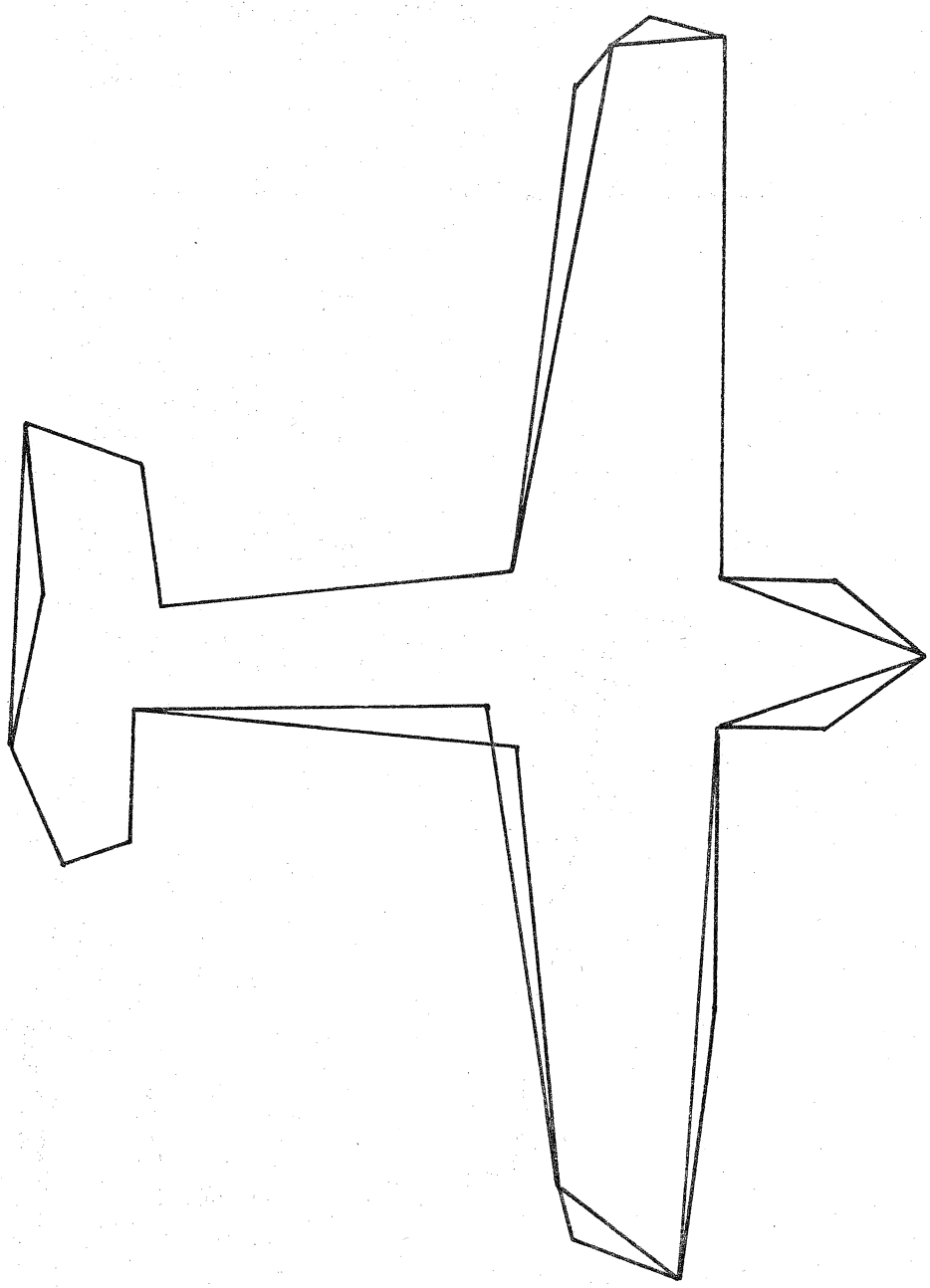


Figure 10b Segmentation of Shape 3

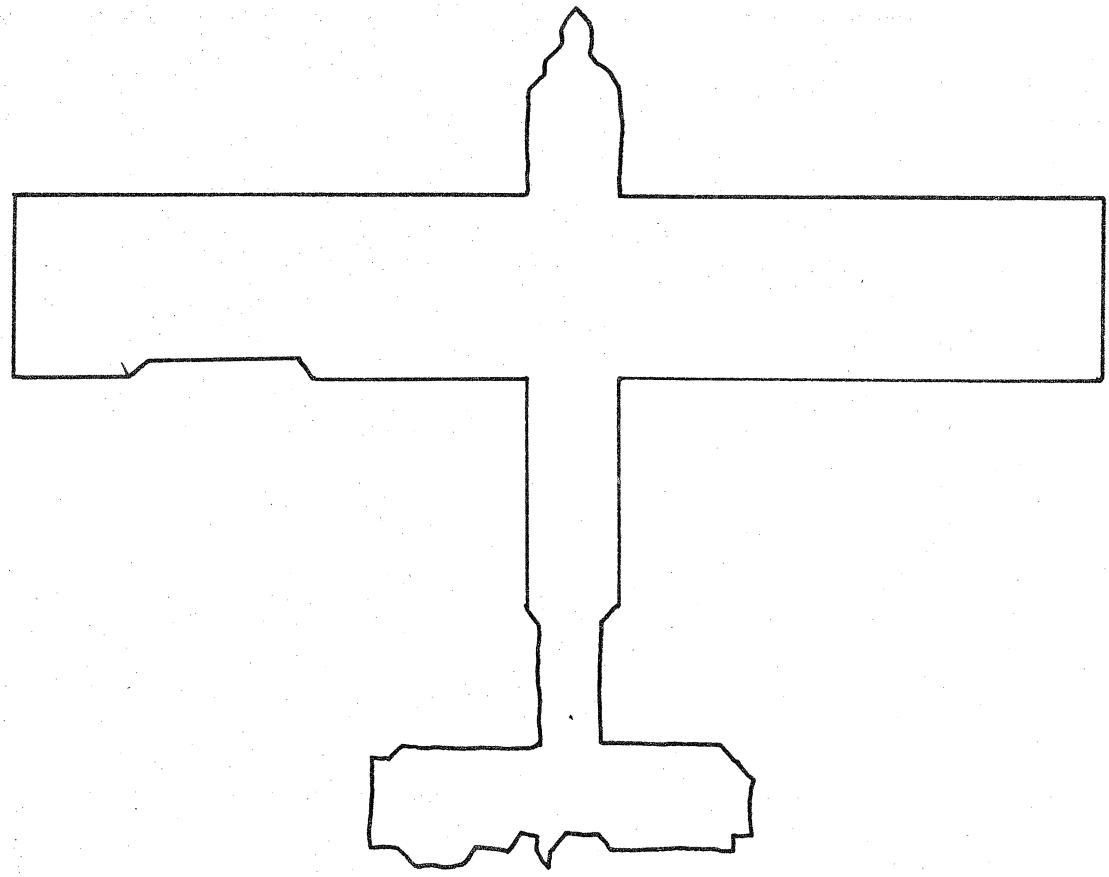


Figure 11a: Shape 4

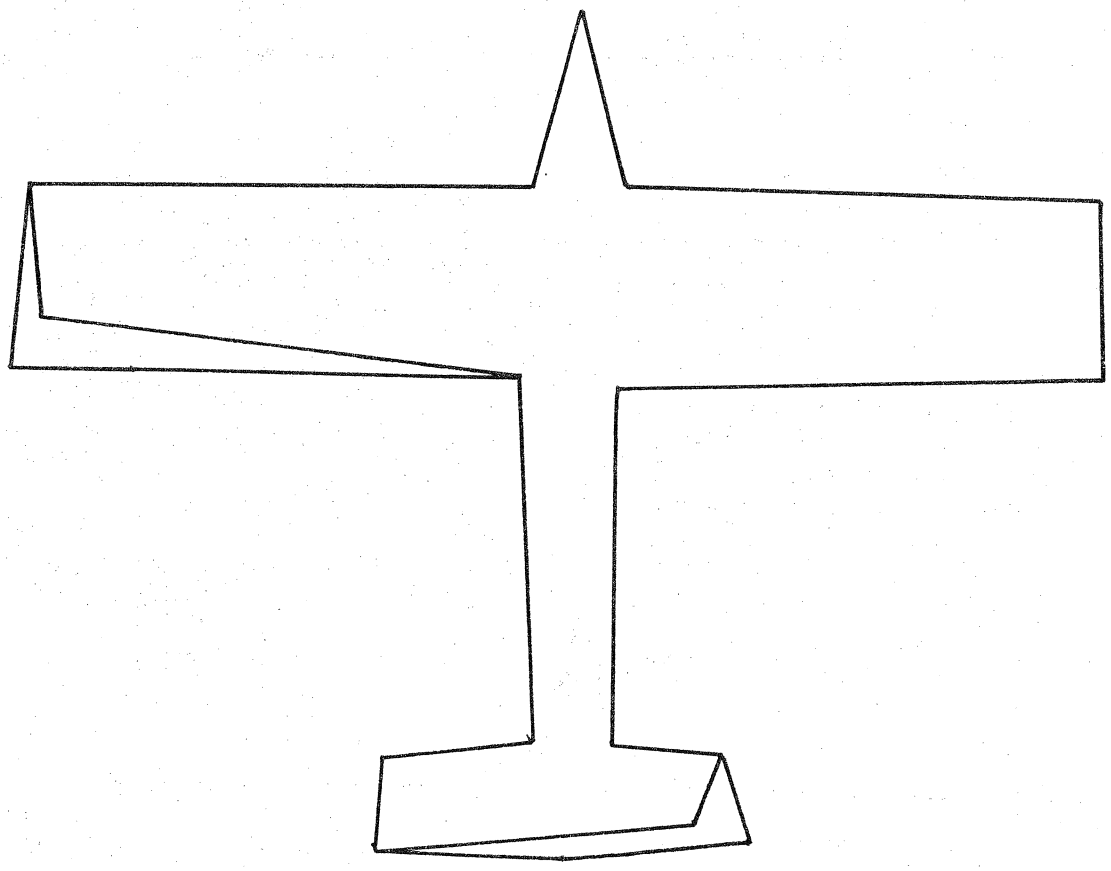


Figure 11b Segmentation of Shape 4

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:

$e_0(i) = N_a(i)/N_0(i)$ and $e_1(i) = N_a(i)/N_1(i)$.

These measures reflect the efficiency of the processes in terms of storage space used, where a value of 1 means that only as many nodes were used at level i as were needed. Tables 1-2 give the results for the experiments.

Tables 1-2 gives a comparison of the node efficiency of HCP for each shape at each level. The first row gives the node efficiency when no constraints are applied to eliminate hypotheses. The second row gives the node efficiency of HCP with all constraints applied. For several shapes, the node efficiency remains fairly constant over the first three levels. This is due to the fact that the first two levels are involved in the description of airplane engines, and if the shape has no engines, then each symbol usually gives rise to a single higher level counterpart. It should be observed that HCP is consistently more node efficient at all levels and converges much more rapidly to the correct solution. As a matter of fact, HCP always found the correct solution by level 5.

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	

Table 2 - Node Efficiency with 3 Hypotheses Per Primitive

	Node Level							
Shape	0	1	2	3	4	5	6	
1	.30	.26	.21	.21	.30	.50	1	(No constraints)
	.50	.44	.39	.50	.75	1	1	(All constraints)
2	.17	.25	.50	.45	.32	.50	1	
	.18	.63	.63	.90	1	1	1	
3	.22	.33	.64	.85	.75	1	1	
	.33	.33	.78	.92	1	1	1	
4	.25	.25	.25	.22	.24	1	1	
	.30	.30	.30	.29	1	1	1	
Average	.24	.27	.40	.43	.40	.75	1	
	.33	.42	.52	.65	.95	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 U 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.

If only the most likely start symbol hypothesis is desired, then HCP can be embedded in a search algorithm in such a way as to find the best (i.e., most likely) start symbol hypothesis first. The search algorithm employed here is a modified version of the state-space search algorithm (called M*) described by Barrow and Tennenbaum [32]. The state space search is organized as a tree. Let T(r) be a finite tree with root node r. For every $n \in T(r)$, let T(n) designate the sub-tree with root node n. If t is a terminal node of T(r), then v(t) denotes the value of t. A best terminal node t is a terminal node such that $v(t) > v(n)$ for every $n \in$ terminal nodes. If n is a non-terminal node, let v(n) be the value of the best terminal node in T(n). Finding the best parse can now be formulated as finding the best terminal node in a search tree, T(r), and can be accomplished by the A+ algorithm [32]. This is just one form of the ordered search algorithm.

Algorithm A+.

Let f be an evaluation function for estimating the value of nodes in T(r). That is, f(n) is an estimate of v(n) and v(n) is bounded by f(n). Search algorithm A+ is defined as follows:

(1) Put node r in a set called OPEN.

(2) Select $n \in$ OPEN such that $f(n) > f(m)$, for any m distinct from n in OPEN. Break ties arbitrarily, but in favor of terminal nodes.

(3) If n is a terminal node, then terminate; else continue.

(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

likelihoods of its neighboring hypotheses. For a hypothesis to contribute to a complete parse, it must be joined to a neighboring hypothesis at each endpoint. Thus, given a hypothesis h, an upper bound on the likelihood of any hypothesis produced from h is the minimum of L(h) and the maximum likelihood of any neighboring hypothesis. Then, in addition to applying CONSTRAIN and COMPACT to sets of hypotheses, we define a constraint operator (called CONSTRAIN*) to be applied to the likelihood of a hypothesis. The constraint operator assigns likelihoods as follows:

$$L^{t+1}(h) := \min\{ L^{t}(h), \max\{ L^{t}(h_i) : h_i \in Nei(h) \} \},$$

where $L^{t}(h)$ is the likelihood of hypothesis h after the tth iteration of CONSTRAIN*, and Nei(h) is the set of hypotheses which neighbor h. $L^{0}(h)$ is the initial likelihood of hypothesis h computed when the level containing h is built. This operator is applied iteratively until no changes in likelihood occur. The M* algorithm of Barrow and Tennenbaum [32] can be modified to perform search in conjunction with HCP. We call this new algorithm HCP*:

Algorithm HCP*

(0) Generate the initial set of hypotheses and compute the corresponding likelihood of each hypothesis. Apply CONSTRAIN* and CONSTRAIN to the network. Save the result on OPEN.

(1) Select the current globally best node, s, from OPEN, and remove s from OPEN. In case of a tie, choose any terminal node ; if none, choose the node with the highest level hypotheses. If s is a terminal node, then halt.

(2) If s has no ambiguous pieces of boundary, then:

(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,

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

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

.
.
.

$S_m \rightarrow h_{m1}h_{m2}\ldots h_{rm}$,

and let $L(S_i) = \min\{ L(h_j) : 1 < j < ri \}$. Suppose that S_b is the best start symbol hypothesis, i.e., $L(S_b) > L(S_i)$ for $i \neq b$ and $1 \leq i < m$. We first show that if h_{k_i} is a level k hypothesis used in the production of any S, then even if the constraint operator is applied, $L(h_{k_i}) > L(S)$.

Lemma: Let h_{k_i} be a level k hypothesis used in the production of a start symbol Sc. Then $L(h_{k_i})$ is never lowered below L(Sc) by CONSTRAIN*.

Proof: For k = 0, it is true, since initially $L(h_{c_i}) \geq L(S_c)$, for $1 \leq i \leq rc$, by definition. Suppose $L^t(H_{c_i}) > L(S_c)$ for all i, $1 \leq i \leq rc$. Then $L^{t+1}(h_{c_i}) = \min\{ L^t(h_{c_i}), \max\{ L^t(h_j) : h_j \in Nei(h_{c_j}) \}\}$; but $L^t(h_{c_i}) \geq L(S_c)$ by assumption, and since $h_{c_{i+1}} \in Nei(h_{c_i})$ and $L^t(h_{c_{i+1}}) \geq L(S_c)$ by assumption, then $L^{t+1}(h_{c_i}) \geq L(S_c)$.

Now suppose that for $k < j$, $L(h_{k_i}) \geq L(S_c)$, for all hypotheses in level 0, ..., j-1 used to produce S_c. Then, initially, all level j hypotheses, h_{j_i}, have $L(h_{j_i}) \geq L(S_c)$ since they are produced from level j-1 hypotheses whose likelihoods are greater than or equal to $L(S_c)$. But then, by an induction similar to above, $L^t(h_{j_i}) \geq L(S_c)$, for all t.

Thus, CONSTRAIN* never lowers the likelihood of any hypothesis used to construct Sc below L(Sc).

Proposition 1: If $L(S_b) > L(S_c)$, $c \neq b$, then S_b is the first start symbol hypothesis produced by HCP*.

Proof: Suppose not. Let v be the first node removed

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 $L(S_b) > L(S_c)$.

6. Conclusions

This paper has discussed the design of hierarchical constraint processes, and discussed their application to shape recognition. The particular class of shape considered was the silhouettes of airplanes viewed from above. We have not considered the more general problem of recognizing a three-dimensional object based on an arbitrary two-dimensional projection.

There are a variety of issues concerning the design of a generally useful shape analysis system which this paper has not addressed. For example, the constructon of the airplane grammar was a painful, time consuming process. Here, it would have been useful to have an interactive tool for constructing such models. Completely automatic grammatical inference mechanisms do exist, but they tend to produce unweildy and unnatural grammars.

Another important question which deserves further consideration is the way in which hypothesis formation is integrated into the constraint application system. This will have a major impact on the efficiency and performance of the system. For example, instead of assuming that only level 0 symbols of the grammar have semantic descriptions which can be directly compared with the descriptions of the primitives, it may be that there are several levels of the grammar at which this is possible. HCP would now begin by detecting primitives at some suitably high level in the grammar, and applying

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.

REFERENCES

1. Stockman, G. "A Problem-Reduction Approach to the Linguistic Analysis of Waveforms," U. of Maryland, TR-538, May 1977.

2. Davis, L., "Shape Matching Using Relaxation Techniques," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol.1, 1979., pp. 60-72.

3. Davis, L. and A. Rosenfeld, "Hierarchical Relaxation for Waveform Parsing," in Computer Vision Systems, (ed. Hanson and Riseman), Academic Press, 1978, pp. 101-109.

4. Zusne, L., Visual Perception of Form, Academic Press, N.Y., 1970.

5. Cornsweet, T.N., Visual Perception, Academic Press, N.Y., 1970.

6. Haber, R.N. and M. Hershenson, The Psychology of Visual Perception, Holt, Rinehart and Winston, N.Y., 1973.

7. Sutherland, N.S., Shape Discrimination by Animals, Experimantal Psychology Society Monograph No. 1.

8. Marr, D., "Early Processing of Visual Information," Phil. Trans. Royal Society, B, 276, 1976, pp. 483-524.

9. Hu, M., "Visual Pattern Recognition by Moment Invariants," IRE Trans. on Inf. Theory, Vol. IT-8, Febuary 1962, pp. 179-187.

10. Zahn, C.T. and R.Z. Roskies, "Fourier Descriptors for Plane Closed Curves," IEEE Trans. on Computers, Vol. C-21, 1972, pp. 269-281.

11. Grunland, G.H., "Fourier Preprocessing for Hand Print Character Recognition," IEEE Trans. on Computers, Vol. C-21, 1972, pp. 195-201.

12. Fu, K.S., "Introduction to Syntactic Pattern Recognition,"

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.

24. Ullmann, J.R., "Subset Methods for Recognizing Distorted Patterns," IEEE Trans. on Systems, Man, and Cybernetics, Vol. SMC-7, 1977, pp. 180-191.

25. Gaschnig, J., "Experimental Case Studies of Backtrack vs. Waltz-type vs. New Algorithms for Satisficing Assignment Problems," Proc. of the 2nd National Conf. of the Canadian Soc. for Computational Studies of Intelligence, Toronto, Canada, July 19-21, 1978, pp. 268-277.

26. Haralick, R. amd G. Elliot, "Increasing Tree Search Efficiency for Constraint Satisfaction Problems," 5'th Int. Joint Conf. on Artificial Intelligence, Tokyo, Japan, August 1979.

27. Henderson, T. and L. Davis, "Shape Recognition Using Hierarchical Constraint Analysis," Proc. Conf. on Pattern Recognition and Image Processing, Chicago, Il., August 6-8, 1979.

28. Vamos, T. and Z. Vassy, "Industrial Pattern Recognition Experiment - A Syntax Aided Approach," IJCPR-73, Washington, pp. 445-452.

29. Gallo, V., "A Program for Grammatical Pattern Recognition Based on the Linguistic Method of the Description and Analysis of Geometrical Structures," IJCAI-75, Tbilisi, Georgia, USSR, 1975, pp. 628-634.

30. Gries, D.,Compiler Construction for Digital Computers, John Wiley, N.Y., 1971.

31. You, K. 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.

32. Barrow, H. and M. Tennenbaum, "MSYS: A System for Reasoning About Scenes," SRI AI Tech. Report No. 121, SRI, Menlo Park, Ca., 1976.