INSTITUT FÜR NACHRICHTENTECHNIK	INTERNER BERICHT	REG. NR.: NE-NT-D-14-80 DATUM: 27.2.1980

BETREFF/VORHABEN/PROGRAMM: Vortrag in Oxfort, England in der Zeit vom 9.-11.1.80 anläßlich der Conferenz on Pattern Recognition 1980

Hierarchical Models and Analysis of Shape

ÜBERSICHT:

VERTEILER: INST.-LTNG.; LOGISTIK; ABT.-LTNG.:

DIESER BERICHT UMFASST:

24 BLATT, DAVON 24 BLATT TEXT, 3 BLATT ABBILDUNGEN , 2 Tab.

VERFASSER:	ABT.LTR.	ZUR VERÖFFENTLICHUNG FREIGEGEBEN:
Dr. Tom Henderson Thomas C. Henderson	i.V. [illegible] INST.LTR.	[illegible]

Hierarchical Models and Analysis of Shape

Tom Henderson*
Larry Davis
Computer Sciences Dept.
The Univ. of Texas
Austin, Texas 78712

Abstract

A major application of syntactic pattern recognition is the analysis of shape. In order for the syntactic approach to work, shapes to be analyzed must be segmented appropriately into pieces which correspond to the terminal symbols of some grammar, and these pieces must subsequently be analyzed by a parsing mechanism. Most syntactic methods assume that the pieces can be easily found. However, in many 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.

Our proposed solution to this problem is to apply contextual constraints at all levels of structural description of the shape in order to eliminate quickly a false alarm hypothesis. This requires two capabilities: a method for deriving the constraints from our grammatical model, and a method for applying these constraints to the hypothesized pieces.

Shape grammars are developed which allow for the description of syntactic and semantic constraints between symbols of the grammar. These grammars are called stratified context-free shape grammars, and they provide a strict hierarchical structure for vocabulary symbols. From these grammars, syntactic and semantic

* currently at DFVLR Oberpfaffenhofen, Inst. für Nachrichtentechnik

constraints for all the vocabulary symbols can be generated automatically.

The contextual constraints generated from a shape grammar can be used by a hierarchical constraint process in analyzing shapes. Such a process constitutes a bottom-up, constraint-based parsing method which attempts to overcome the combinatorial explosion in searching for a parse of the shape implied by the segmentation strategy.

Examples of the application of this hierarchical system to airplane recognition are described.

1. Introduction

A major application of syntactic pattern recognition is the analysis of shape. The syntactic paradigm has been applied in various domains, including electrocardiagram interpretation (Horowitz [1]), fingerprint classification (Moayer [2]) and earth resources image analysis (Brayer [3]). Another important technique in computer vision is constraint propagation which has been used for scene labeling (Waltz [4], Barrow [5]), low-level vision (Rosenfeld [6]) and shape matching (Davis [7]). We describe a way of combining syntactic and constraint propagation techniques; in particular, we describe a hierarchical constraint process which consists of a hierarchical shape model, a method for automatic constraint derivation from the model, and a method for using the derived constraints in analyzing shape.

Syntactic pattern recognition proceeds in three major steps: preprocessing, pattern representation and grammatical analysis.

Preprocessing improves the quality of an image containing the shape, e.g., filtering, enhancement, etc. Pattern representation includes segmenting the shape and associating the segments with the parts in the syntactic model. Finally, the primitive shape parts are organized according to the syntactic model. As a result of the syntax analysis, a parse tree is produced, and the parse tree can be used not only for recognition purposes, but also as a description of the shape.

A structural shape model describes 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 (Freeman [8]) and boundary segments determined by piecewise functional approximations (Pavlidis [9]). We use the latter as shape primitives (see Section 4.).

2. The Hierarchical Model

Syntactic models for shape analysis have been developed and investigated by many workers. With some simple modifications, constraint analysis techniques can be incorporated in a natural way into this type of model. An extension of the geometrical grammars of Vamos [10] and Gallo [11] is used to model shape.

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 start symbol.

Let V = (N ∪ T) be the set of vocabulary symbols. Associated with every symbol v ∈ V is a level number, ln(v), where ln(v) is in the range (0,1,...,n) and n is the level number of the start symbol. Every element of T has level number 0.

T consists of a set of symbols each of which corresponds to a relatively large piece of the shape modeled by the grammar, i.e., straight-edge approximations to the boundary of the shape.

N consists of a set of symbols each of which has a level number from 1 to n associated with it. The start symbol has level n, and in any rule v := a (the rewrite part of a production), if ln(v) = k, then every symbol in the string a has level number (k-1). Furthermore, ∀ v ∈ V

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

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

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

[semantic part] is a set of predicates which describes certain aspects of the symbol.

P consists of productions of the form (v:=a,A,C,Ga,Gs), where 1) v := a is the rewrite part that indicates the replacement of the symbol v by the group of symbols a, where v ∈ N and the string a = v1v2...vk (vi ∈ V and ln(vi) = (ln(v)-1), i=1,k). 2) A is a set of applicability conditions on the syntactic ar-

rangement of the vi.

3) C is a set of predicates describing the semantic consistency of the vi, that is, geometric and other properties of the vi.

4) Ga consists of rules for generating the attachment part of v.

5) Gs consists of rules for generating the semantic part of v.

Thus, the higher the level number of a vocabulary symbol, the more of the boundary of the shape is accounted for. For example, in the airplane grammar described in [12], the level 5 <head section> may represent as many as 26 level 0 symbols, whereas the level 1 <engine> symbol is comprised of only 3 level 0 symbols. Moreover, a grammar is written to describe a general class of shapes, say airplanes, and not some particular airplane.

As an example of the productions of the grammar, consider how the <engine> symbol is formed (see Figure 1):

<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',e2") and Join(e1"' or e2"' ,e1")]

C : [Parallel(a',a"') and Length(a')=Length(a"') and
Perpendicular(a',a") and Parallel(a",Vector(Midpt(a'),
Midpt(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 : [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, Ga and Gs 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), where Join(x,y) means that x and y are the same point. 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>. Ga and Gs describe the derivation of the attachment points and semantic features for <engine>; the unjoined endpoints of the <engine side> symbols can be given either attachment point name due to the symmetry of the symbol, where Unjoined(x,y) indicates the endpoint which did not satisfy the Join predicate in the applicability part of the production, and Set(x,y) means that the point x of the hypothesis being created is given the physical attributes of the existing endpoint y. The main axis, a, 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 constraint process will utilize to analyze shape.

3. Constraint Generation

We now discuss the procedures for deriving the local constraints from the shape grammar. Two types of constraints, syn-

tactic and semantic, are described. The semantic attributes of a vocabulary symbol are computed from the attributes of the symbols which produce it (see Knuth [13] for a discussion of defining semantics for context-free languages using both synthesized and inherited attributes; we use only synthesized attributes). Consider a vocabulary symbol as representing a piece of the boundary of a shape. If a vocabulary symbol is part of a complete shape, then it is adjacent to pieces of the shape which can combine with it to produce a higher level vocabulary symbol. Therefore, if the set of all possible neighbors of a vocabulary symbol is known, and at one of its attachment points no hypothesis for any of these symbols exists, then that vocabulary symbol hypothesis can be eliminated. This type of constraint is called a syntactic constraint. Without these constraints several levels of vocabulary symbols might be built before it can be determined that some hypothesis lacks the appropriate context. The use of constraints, however, makes it possible to detect much earlier the lack of appropriate context.

The other type of constraint involves some (usually geometric) relation between the semantic features of two vocabulary symbols, e.g., the main axis of a <plane> is parallel to the main axis of an <engine>. These kinds of constraints are called semantic constraints. This makes it possible for high level information to be specified, e.g., the orientation of the plane, and this information can be used to delete hypotheses which are not consistent with the given information.

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

1) (v,av) Ancestor (w,aw) iff $\exists$ p $\in$ P $\ni$ the rewrite rule of p is v := ...w... and $\exists$ aw $\in$ at(w) $\ni$ aw is identified with av in Ga of p. Then we say that v is an ancestor of w through attachment point av of v and aw of w, where av and aw represent the same physical location. For example, in Figure 1 the attachment points for the symbol <engine> are associated with the unjoined attachment points of the <engine side> symbols, thus making <engine> an ancestor of <engine side> through any choice of endpoints.

2) (w,aw) Descendent (v,av) iff (v,av) Ancestor (w,aw).

3) (v,av) Neighbor (w,aw) iff

a) $\exists$ p $\in$ P $\ni$ the rewrite rule of p is x := ...v...w..., and aw is specified as being joined to av in the applicability condition of p, or

b) $\exists$ x $\in$ V with ax $\in$ at(x), and $\exists$ y $\in$ V with ay $\in$ at(y) $\ni$ (x,ax) Ancestor (v,av) and (y,ay) Neighbor (x,ax) and (w,aw) Descendent (y,ay).

That is, vocabulary symbols are either directly specified as neighbors in a production, or they are neighbors indirectly by being at the end of higher level symbols which are neighbors.

Using matrix representations for these relations, the descendents and neighbors of a symbol at a particular attachment point can be computed (see Gries [14] for an introduction to binary relations, their representation using matrices and their manipulation). Let s be the number of vocabulary symbols in G, and let the Boolean matrix A_{mn} be the square matrix of order s whose (i,j)th entry is 1 iff symbol v_i is in relation A to symbol v_j through endpoint m of v_i and endpoint n of v_j (consider the endpoints of vocabulary symbols to be ordered). A relation (which is dependent on endpoints) is then fully specified by a

total of k^2 matrices, where k is the number of endpoints per symbol. However, if the grammar is written so that endpoints are interchangable, then one matrix will define the relation, i.e., all k^2 matrices are the same. The Ancestor relation, A_{mn}, is then specified by putting a 1 in the (i,j)th position of A_{mn} if the condition given in the definition is satisfied. The Descendent relation, D_{mn}, is just the transpose of A_{mn}. Given A_{mn}, D_{mn} and ln(S), i.e., the level number of the start symbol, the neighbor relation, N_{mn}, is computed by iterating the following computation ln(S)-1 times:

$$N_{mn}^{(l+1)} := N_{mn}^{(l)} + \sum_{p=1}^{k} \{D_{mp} * [\sum_{q=1}^{k} (N_{pq}^{(l)} * A_{qn})]\},$$

where + is Boolean "or" and * is Boolean "and", and $N_{pq}^{(0)}$ is just the explicit neighbors given in the productions. If a hypothesis fails to have a neighbor from this set, then that hypothesis can be deleted; this relation constitutes the syntactic constraints.

Semantic constraints can be generated in much the same way as syntactic constraints: by defining binary relations and computing their transitive closure. For example, the axes of two symbols are parallel if a production states this explicitly or by transitivity through some third symbol. There are two methods for dealing with semantic constraints relating to angles.

1) Associate with each endpoint of a vocabulary symbol an end angle. Then, in either the semantic consistency or the applicability condition, indicate an allowed range of angle for two symbols to have if joined at particular endpoints. This approach is analagous to the syntactic neighbor case; now a relation is defined between every two symbols joined at particular endpoints, and the closure will give results for implicit symbol neighbors

(that is, even if two symbols do not appear together in the right hand side of some production, but are joined at a higher level). There are several problems with doing things this way.

In practice, the range of the angle is too large to give much extra constraint, and the explicit semantic constraints in the productions are adequate. A more serious problem concerns making use of the semantic constraints in a more global way. Since a range of angle is given at every endpoint, then if the allowed range is computed for symbols several neighbors away, the allowed range is useless as a constraint.

2) Alternatively, design special purpose relations which when computed will yield semantic constraints between vocabulary symbols. Such constraints can be more general in nature and can be applied with much more certainty. For example, the parallel relation can be used to delete an <engine side> hypothesis that is not parallel to some other <engine side> hypothesis. Such relations also allow for semantic features to be fixed (set to some constant), e.g., the orientation of the main axis of a <plane> symbol could be set to 45°, and this certain information can be propagated through the network of hypotheses. This can be done, for example, by having global information available describing known orientations of the vocabulary symbols. In this way, it is possible to determine whether certain hypotheses can be deleted.

Nothing precludes the use of both methods; however, due to the reasons given above, only the special purpose relations were used. In particular, the parallel relation was computed between all vocabulary symbols. Note that not every symbol is necessarily parallel to another symbol, and as pointed out above, some hypothesized symbols may require other hypotheses of the same

symbol to exist, e.g., <engine side>. The parallel relation was computed using a binary-valued matrix, whose rows and columns correspond to the axes of the vocabulary symbols.

In general, a transitive relation is computed as:

$$P := (P^{(0)} + \bar{I}) * P^{(0)}!$$

where I is the Boolean identity matrix and $P^{(0)}!$ is the transitive closure of $P^{(0)}$, the explicit parallel relation. Computed this way a symbol is only parallel to itself if there must exist another distinct hypothesis for the same symbol. Relations which are not transitive, e.g., perpendicular, require special procedures for their computation.

4. Grammatical Analysis of Shape

As discussed in the introduction, a major problem associated with syntactic pattern recognition 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 constraint process (HCP) uses hierarchical models of objects and uses 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., parallel, relative length, etc.) constraints can be automatically generated to guide the analysis of a 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 [15] fits straight edges to the boundary using the cornerity measure proposed by Freeman and Davis [16] to choose break points. For each point on the original boundary, an error measure defined as the minimum distance from that boundary point to the line segment which approximates a boundary segment containing that boundary point is computed. Then, an error measure for the line segment is defined to be the sum of the errors of each underlying boundary point. Primitives are generated at various error thresholds by applying stricter thresholds to segmentations already generated. By computing several segmentations, it is hoped that all the necessary primitives will be found. The search will be made feasible by the constraints implied in the grammar and imposed by the constraint techniques.

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) and on the semantics associated with the symbols of the grammar. If the grammar is a string grammar, then the relationship of concatenation between segments must be computed. Notice that for a set of primitives corresponding to linear segments obtained by computing piecewise linear approximations to the border of the shape, 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 slightly overlap. Matters are made more complicated by the fact that for short segments, a

small amount of overlap with a larger segment might lead not only to the assertion that the two segments are adjacent, but also that the larger one contains the smaller. For relations more complicated than adjacent, such as "left of","right of", and "inside", the correct definitions become more elusive (see Freeman [17] for a survey of models for computing spatial relations and a discussion of the difficulties associated with making such computations).

The association of terminal symbols with primitives will (in the limit) be to hypothesize every terminal for each primitive. However, methods for reducing the number of hypotheses include using a more global analysis to derive indications of appropriate scale, orientation, etc. from the simple global properties, e.g., histogram selected features of the primitives themselves and use the model to infer properties of particular terminal symbols.

Each match of a terminal symbol to a segment results in a node being entered into level 0 of the network. That node (or hypothesis) corresponds to the hypothesis that the segment of the boundary should be labeled with that specific terminal symbol. Nodes are connected by an edge if the corresponding boundary segments are physically adjacent. Note that a hypothesis may be incorrect in one of two ways: 1) it may associate the wrong terminal symbol with an actual piece of the correct low level segmentation of the shape, or 2) the segment itself may not be part of a correct segmentation of the shape. In this case it does not matter which terminal symbol is associated with that segment.

The first problem can be overcome by assigning every possible terminal symbol to every primitive. Another idea is to histogram some feature of the primitives, e.g., the length, and use this to

restrict the number of hypotheses. For example, the wing tip is one of the shortest pieces in the decomposition of an airplane shape, and therefore the very longest primitives need not have the wing tip hypothesis. The second problem can be circumvented by designing flexible grammars, or by softening the hierarchical constraint process to allow a certain number of missing pieces.

The hierarchical constraint 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 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, nodes corresponding to boundary segments sharing an endpoint are linked, but 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.

CONSTRAIN - since each node corresponds to a single hypothesis, and since nodes are only linked to compatible nodes, the within level application of constraints simply involves removing a node if it has no neighbor at some endpoint.

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, Rx and Rc, both of which are initially empty. When at level k with Rx and Rc empty, BUILD produces the level k+1 hypotheses (or stops if k=n) and puts them into Rx while putting all level k nodes into Rc. CONSTRAIN then removes nodes from Rx, 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 Rx and its across level neighbors in Rc.COMPACT removes nodes from Rc, 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, COMPACT deletes the node from the network and puts its same level-neighbors in Rx and its across level neighbors in Rc.

HCP does not eliminate any hypothesis which contributes to 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 is applied to a set of nodes taken one at a time, and if a hypothesis h is deleted, it is precisely for the reason that at one endpoint of h, no neighboring hypotheses can be joined to h to produce a higher level symbol. 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 since level k+1 is only built once, that level k hypothesis will never produce any higher level hypothesis. Thus, eliminating it does not affect any complete parse. Finally, if a level k hypothesis h loses the support of one or more of the hypotheses which produced it, then clearly

if h were part of some complete parse, then the supporting nodes would be, too.

Of course, constraints can be generated from grammars that are not stratified, but the application of the constraints will not prevent the repeated production of symbols which fail to satisfy the constraints. This is due to the fact that a hypothesis cannot be discarded since it could be used at any time. However, stratification insures that level k symbols will be generated by BUILD only once.

The input to HCP consists of the compiled grammar, that is, the productions of the grammar and the derived constraints, and a set of primitives. Semi-PASCAL versions of HCP, CONSTRAIN and COMPACT are:

```
Procedure HCP;
begin
constrain-set := compact-set := empty set;
level := -1;
while level < n  do
  begin
  level := level + 1;
  Build(level);
  while (constrain-set not empty) or
        (compact-set not empty)  do
    begin
    Constrain(constrain-set);
    Compact(compact-set);
    end;
  end;
end;
```

```
Procedure Constrain(constrain-set);
while constrain-set not empty  do
  begin
  node := Next-in-set(constrain-set);
  if not Satisfies(node)  then
    begin
    Put-on-constrain( Neighbors(node) );
    Put-on-compact( Across-level-neighbors(node) );
    Delete-from-network(node);
    end;
  end;

Procedure Compact(compact-set);
while compact-set not empty  do
  begin
  node := Next-in-set(compact-set);
  if not Supportable(node)  then
    begin
    Put-on-constrain( Neighbors(node) );
    Put-on-compact( Across-level-neighbors(node) );
    Delete-from-network(node);
    end;
  end;
```

5. Experiments

A grammar describing the top view of airplane shapes (down to the level of detail of engines) has been developed (see [12] for the complete grammar). The grammar consists of 37 productions and

has 7 levels of vocabulary symbols. We do not view parsing as a recognition procedure, but rather as a process which imposes organization on the shape (by forming wings, engines, etc.). Recognition is subsequently performed by analyzing the organization.

We will describe the application of HCP to the top view of airplanes. The twelve shapes used in this study were obtained from the literature (see [18]) and from model airplanes and boats. Figure 2 gives a typical shape.

The split-and-merge algorithm 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. Figure 3 gives the primitives obtained from the shape in Figure 2.

Once the primitives have been found, the initial hypotheses for each primitive must be made. Results reported here are with all possible hypotheses. HCP was run with full constraints and with no constraints. Running HCP with no constraints builds every vocabulary symbol which can possibly be built, regardless of whether or not it can be part of a complete parse of the boundary. A measure of efficiency can be defined in terms of the number of nodes produced at each level versus the number of nodes absolutely necessary to produce the shape. Given a shape and a level, i, there is some fixed number of hypotheses, Na(i), which is required to produce the shape. Let $N_0(i)$ be the number of nodes produced when no constraints were used, and let $N_1(i)$ be the number of nodes produced when the constraints were used. Then the efficiency of each process can be given as:

$$e_0(i) = Na(i)/N_0(i) \text{ and } e_1(i) = Na(i)/N_1(i).$$

In the case of the boat shapes, Na(i) = 0, and we propose the following relative measure (where it exists):

$$e_r(i) = N_1(i)/N_0(i).$$

These measures reflect the efficiency of the processes in terms of storage space used, where a value of e(i) = 1 means that only as many nodes were produced at level i as were needed.

Table 1 gives a comparison of node efficiency of HCP for each shape at each level. The first row gives the node efficiency when the complete network is built, i.e., 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.

The plot given in Table 2 shows the average node efficiency at each level of HCP with and without constraints. This plot reveals that HCP is much more efficient computationally as far as storage requirements are concerned.

6. Conclusions and Future Research

The hierarchical constraint process has been successfully used to recognize silhouettes of airplanes. A system of programs has been developed which automatically generates syntactic and

semantic constraints for a given stratified shape grammar and applies them and the grammar to analyze a set of low level hypotheses about a shape. For an account of an earlier set of experiments applying HCP to shape recognition, see Henderson [19]. The goals accomplished and reported in this study include:

1) Programs written in FORTRAN for obtaining a segmentation of a shape. These programs take as input a chain code and produce as output a set of piecewise linear approximations which includes segments from various thresholds of goodness of fit.

2) Hierarchical shape models with a semantic component for specifying geometric relations between the pieces. The design and debugging of shape grammars is one major difficulty with using HCP. There are no strict criteria for a best or even a good grammar, e.g., no indication of the trade-off between the number of symbols in the right hand side of the rewrite rules vs. the number of levels in the grammar. Automated grammatical inference may be helpful in these respects, but the desirability of decomposing a shape into natural pieces may require an interactive approach.

3) Procedures for deriving syntactic and semantic constraints that are implicit in the grammar. The constraints are generated prior to the analysis of the data and are compiled only once for a given grammar. The semantic constraints include the parallel relation, and other transitive geometric constraints can be added to HCP in a modular fashion. The experiments run indicate that the use of these constraints provides a great increase in the efficiency of the analysis.

Several extensions can be made to improve HCP. The syntactic constraints as currently implemented require every hypothesis to have some supporting hypothesis adjacent at each endpoint. This

might be "softened" in BUILD by requiring that only some part of the right hand side to be present, e.g., all but one symbol there. This would provide a means of completing a parse even if a correct hypothesis had been omitted.

Another possibility is to integrate hypothesis formation into the constraint system. This will have a major impact on the efficiency and performance of the system. Instead of assuming that only level 0 symbols have semantic descriptions which can be directly compared with the descriptions of the primitives, we will assume 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 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 [20,21,6] 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 (one study has been done by Gaschnig [22]). 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 asses the real significance and practical importance of such systems.

Hierarchical organization of shape and constraint analysis have been shown to be useful concepts in shape analysis. A

method has been provided for dealing with noise and ambiguity in the data. The analysis of a shape is based on the local constraints which can be generated from a high level model. These constraints are applied at all levels of the model and lead to a more efficient analysis.

Bibliography

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

2. Moayer, B. and K.S. Fu, "Syntactic Pattern Recognition of Fingerprints," Purdue U., TR-EE 74-36, Dec. 1974.

3. Brayer, J.M., P.H. Swain and K.S. Fu, "Modeling of Earth Resources Satellite Data," in Syntactic Pattern Recognition, Applications (ed. K.S. Fu), Springer-Verlag,1977, pp. 215-242.

4. Waltz, D., "Understanding Line Drawings of Scenes with Shadows," in The Psychology of Computer Vision, (ed. P.H. Winston), McGraw-Hill, 1975, pp. 19-91.

5. Barrow, H.G., A.P. Ambler and R.M. Burstall, "Some Techniques for Recognizing Structures in Pictures," in Frontiers of Pattern Recognition, (ed. S. Watanabe), Academic Press, 1972, pp. 1-29.

6. 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.

7. Davis, L., "Shape Matching Using Relaxation Operations," U. of Maryland, TR-480, Sept. 1976.

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

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

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

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

12. Henderson, T., "Hierarchical Constraint Processes for Shape Analysis," Ph.D. Thesis, U. of Texas, Dec.,1979.

13. Knuth, D., "Semantics of Context-Free Languages," Math. Systems Theory, Vol. 2, No. 2, 1968, pp. 127-145.

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

15. Pavlidis, T. and S. Horowitz, "Segmentation of Plane Curves," IEEE Trans. on Computers, Vol. C-23, 1974, pp. 860-870.

16. Freeman, H. and L. Davis, "A Corner-Finding Algorithm for Chain Coded Curves," IEEE Trans. on Computers, Vol. C-26, March 1977, pp. 297-303.

17. Freeman, J., "The Modeling of Spatial Relations," Computer Graphics and Image Processing, 4, 1975, pp. 156-171.

18. 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 to Jan. 31, 1977, March, 1977, pp. 72-83.

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

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

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

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

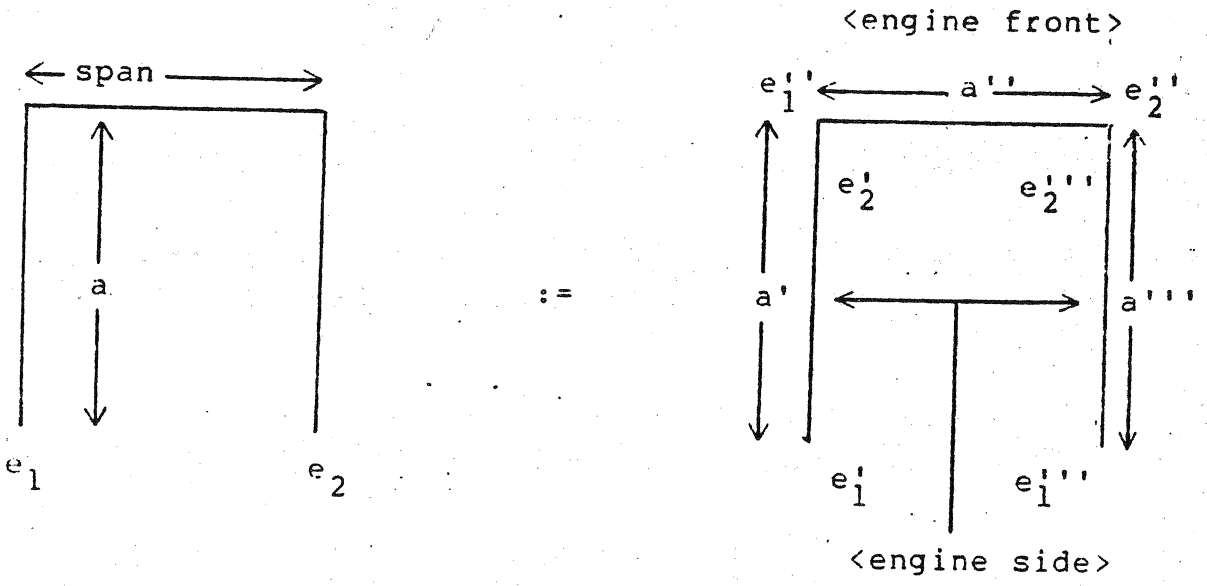

Figure 1 : Example of a Production

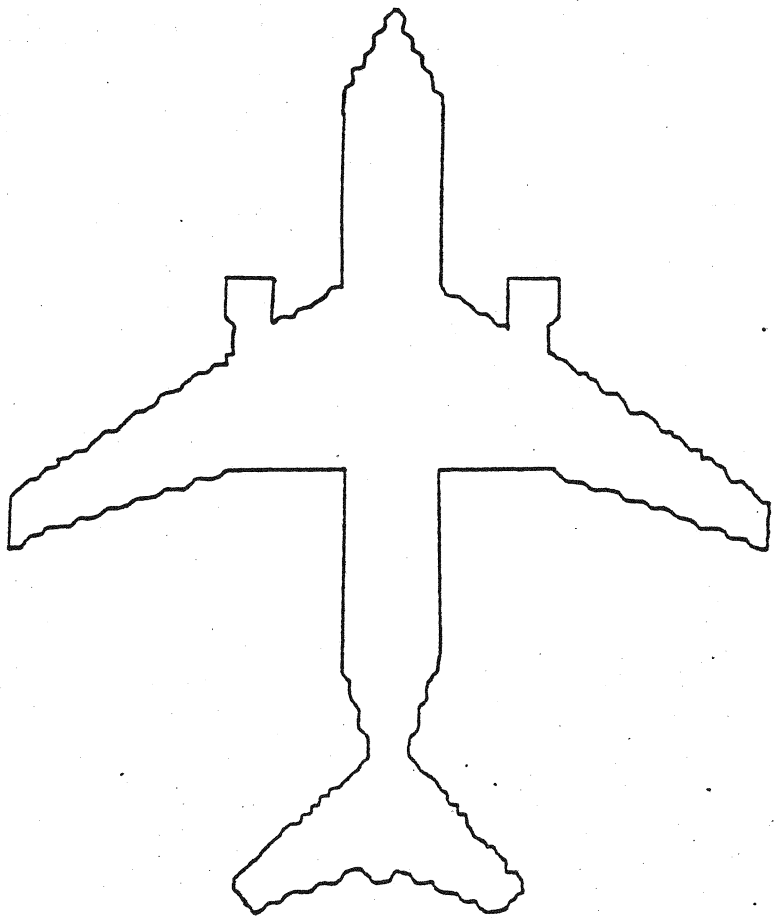

Figure 2 : A Typical Airplane Shape

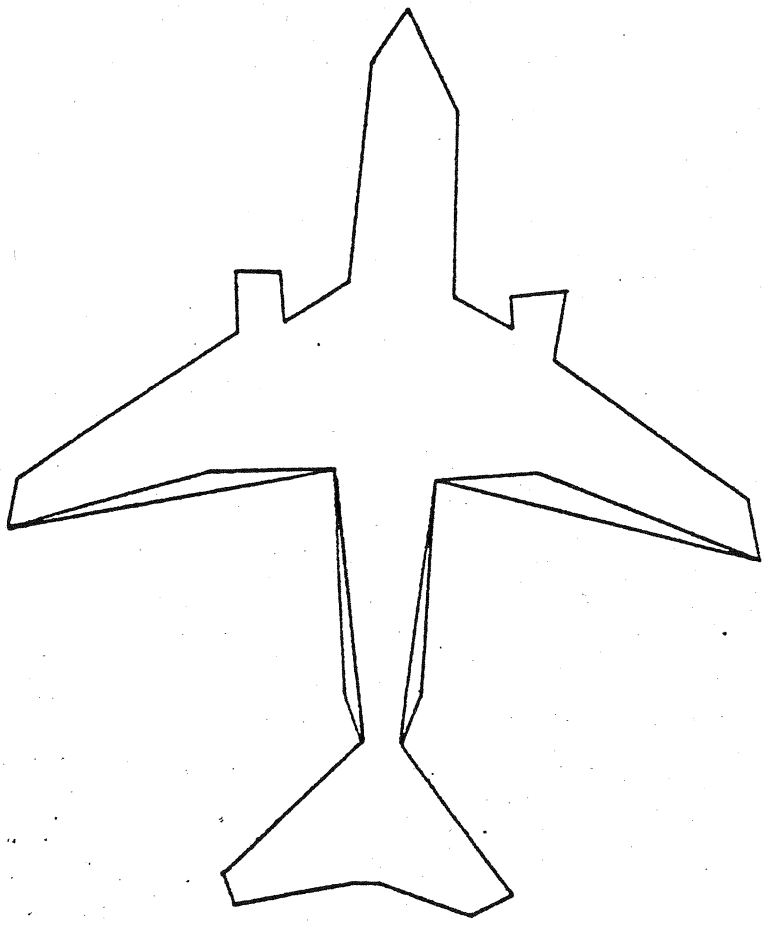

Figure 3: Segmentation of Shape in Figure 2

Table 1 - Node Efficiency

	Node Level							
Shape	0	1	2	3	4	5	6	
4	.11	.16	.26	.26	.74	1	1	(No constraints)
	.16	.16	.26	.34	1	1	1	(All constraints)
8	.10	.12	.17	.15	.24	.57	1	
	.14	.14	.17	.24	.88	1	1	
10	.11	.11	.15	.13	.16	1	1	
	.13	.13	.18	.19	1	1	1	
11	.71	.83	.91	.81	-	-	-	(Relative Eff.)
Avg.	.11	.13	.19	.18	.38	.86	1	
	.14	.14	.20	.26	.96	1	1	

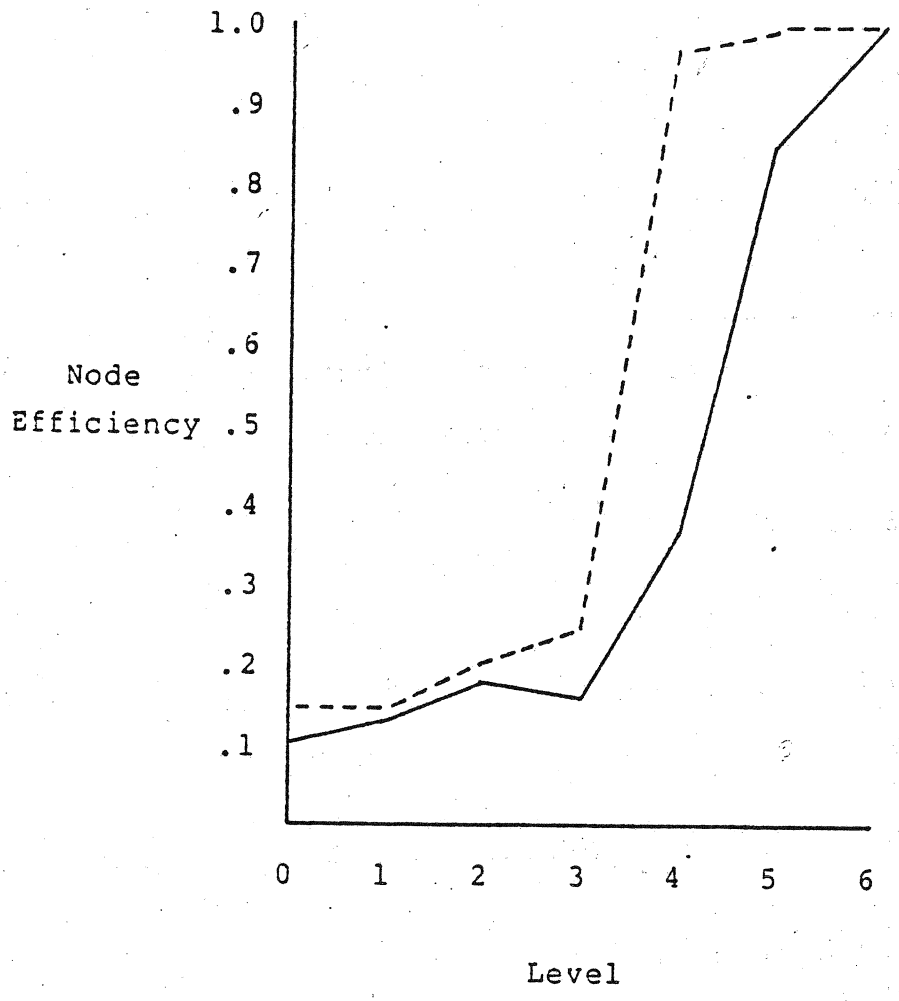

Table 2 - Average Node Efficiency