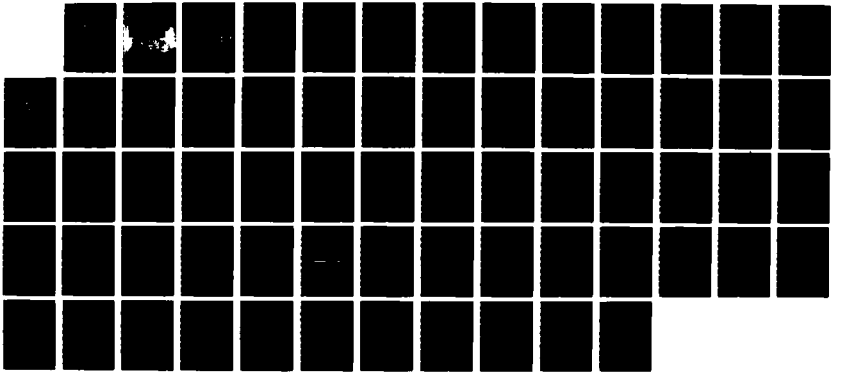
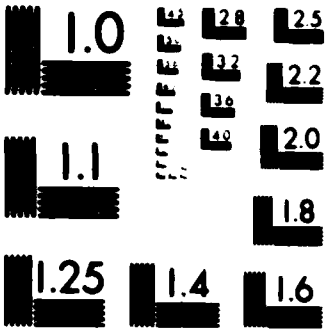
AD-A181 378 PROCESSINN WAVEFORMS AS TREES FOR PATTERN RECOGNITION 1/1
(U) RICE UNIV HOUSTON TX DEPT OF ELECTRICAL AND
COMPUTER ENGINEERING S W SHAW ET AL MAY 86 EE-8605
UNCLASSIFIED N00014-85-K-0152 F/G 12/9 NL





(12)

DTIC FILE COPY

RICE UNIVERSITY

GEORGE R. BROWN SCHOOL OF ENGINEERING

AD-A181 378

PROCESSING WAVEFORMS AS TREES FOR PATTERN RECOGNITION

(Supported by the NASA grant NGT 44-006-806 and ONR contract N 00014-85-K-0152)

by

Scott W. Shaw & Rui J.P. deFigueiredo

Technical Report EE8605
Department of Electrical and Computer Engineering
Rice University, Houston, Texas 77251-1892
May, 1986

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

HOUSTON, TEXAS

(12)

PROCESSING WAVEFORMS AS TREES FOR PATTERN RECOGNITION

(Supported by the NASA grant NGT 44-006-806 and
ONR contract N 00014-85-K-0152)

by

Scott W. Shaw & Rui J.P. deFigueiredo

Technical Report EE8605
Department of Electrical and Computer Engineering
Rice University, Houston, Texas 77251-1892
May, 1986

A

Processing Waveforms as Trees for Pattern Recognition

Abstract

Waveforms may be represented symbolically such that their underlying, global structural composition is emphasized. One such symbolic representation is the relational tree. The relational tree is a computer data structure that describes the relative size and placement of peaks and valleys in a waveform. Researchers have developed various distance measures which serve as tree metrics. A tree metric defines a tree space. We are able to cluster groups of trees by their proximity in tree space.

Linear discriminants are used to reduce vector space dimensionality and to improve cluster performance. A tree transformation operating on a regular tree language accomplishes this same goal in a tree space. Under certain restrictions, relational trees form a regular tree language.

Combining these concepts yields a waveform recognition system. This system recognizes waveforms even when they have undergone a monotonic transformation of the time axis. The system performs well with high signal to noise ratios, but further refinements are necessary for a working waveform interpretation system

Table of Contents

Introduction

When classifying signals of one or two dimensions, it is sometimes only necessary to take global structural information into account. By structural information, we mean the relative height and placement of peaks and valleys in the waveform. A new automatic waveform interpretation system has been developed which exploits waveform structure through syntactic representation and cluster analysis. In addition to drawing on previously developed concepts, we show how a symbolic space can be transformed to improve the perfomance of a clustering system. Simulations prove that such a recognition technique might be used to classify waveforms with few errors. This paper describes the components of the waveform recognition system and how they interact.

In our symbolic recognition system, waveforms are represented syntactically and then grouped into clusters which reflect some underlying structural similarity. Erich and Foith [1], and Lu [2], have described hierarchical computer data structures to represent structural information for waveforms. Of these, the relational tree is particularly simple. We shall use the relational tree, in concert with traditional cluster analysis and formal tree automata theory, to construct the waveform recognition system. A tree clustering objective function will be introduced to assess clustering performance, and a tree transformation is introduced to improve this performance. The tree clustering system, illustrated by simulations, compares favorably with numerical techniques when the circumstances are appropriate.

The first two chapters of this report describe the building blocks of our waveform recognition system, i.e. the relational tree representation and traditional cluster analysis. Following that, Chapter 3 shows how previously developed tree automata theory can be applied to the problem of tree cluster optimization. We then introduce a tree clustering objective function to quantify cluster improvement. Lastly, in Chapters 4 and 5, the components are assembled into a total system and tested on noisy data.

We shall begin by thoroughly describing various tree topologies for representing signal structure. There are three; the *relational tree*, the *skeletal tree*, and the *complete tree.* Due to its simplicity, the relational tree is the most appropriate for this work. In order to provide a comprehensive overview, we shall also discuss the skeletal and complete trees.

Chapter 1

Review of Tree Representations

1.1 The Relational Tree Representation

Ehrich and Foith first proposed representing a signal by a relational tree (RT) [1]. The relational tree provides a two-dimensional description of a one-dimensional signal. It draws on the intuitive notion of a waveform as a sequence of peaks and valleys. Without attributes attatched to its nodes, the RT contains only information about the relative sizes and placement of peaks and valleys in the signal. Attributes may be added to the nodes of the tree to supply information such as the value of time (independent variable) and amplitude. The RT is insensitive to scaling on either axis. This is beneficial in identifying signals that undergo this type of scaling.

In order to discuss the concept of relational trees, we will adopt the following definitions:

Definition 1:

A waveform segment is a one-dimensional positive function of finite length containing a finite number of maximum and minimum points. In addition, its endpoints must be lower than any other point in the segment.

Erich and Foith [1] define a peak and a valley as relations;

Definition 2:

(x_i^P, y_i^P) is a peak P_i iff $y_i^P \geq y$, $\forall (x,y) | \; x_i^P - \delta \leq x \leq x_i^P + \delta$

(x_i^V, y_i^V) is a valley v_i iff $y_i^V \leq y$, $\forall (x,y) | x_i^V - \delta \leq x \leq x_i^V + \delta$

Definition 3:

A waveform segment's left (right) peak is that portion of the segment bounded on the left (right) by the left (right) boundary of the segment, and on the right (left) by the lowest valley in the segment.

Definition 4:

A *tree* is a rooted, directed, acyclic graph with no more than one edge entering each node, and zero or more edges exiting each node. Nodes which have zero edges exiting are called *terminal* nodes. All other nodes are known as *non-terminal*. Each tree has one node with no edges entering it known as the root. A tree is *binary* if no more than two edges exit any node.

Definition 5:

A *relational tree* representing a single peak segment consists of an isolated terminal node $t_S = p$. A segment S containing two or more peaks has an associated RT consisting of a root node σ with two descendants t_1 and t_2, written

$$t_S = \sigma(t_1, t_2)$$

where t_1 and t_2 are RT's describing the left and right peaks of S. Symbolically,

$$t_S ::= p_i | \sigma(t_1, t_2),\ i=0,1,...$$

Each non-terminal node in an RT represents a valley in the waveform. Each terminal node represents a peak. The valleys are nested according to relative depth. The root node of the RT is chosen to represent the deepest

valley in the waveform. This divides the waveform into two segments; one to the right of this valley, and one to the left. The descendants of this node will be RT's describing each segment. Each root node is labeled by its dominant peak, i.e. the highest peak in either segment. The non-terminal descendants of any node represent the deepest valleys in the right and left segments. They are in turn labeled by their dominant peaks (see figure 1 a & b). If a segment contains only a peak and no valleys, it is represented by a terminal node and then labeled by that peak.

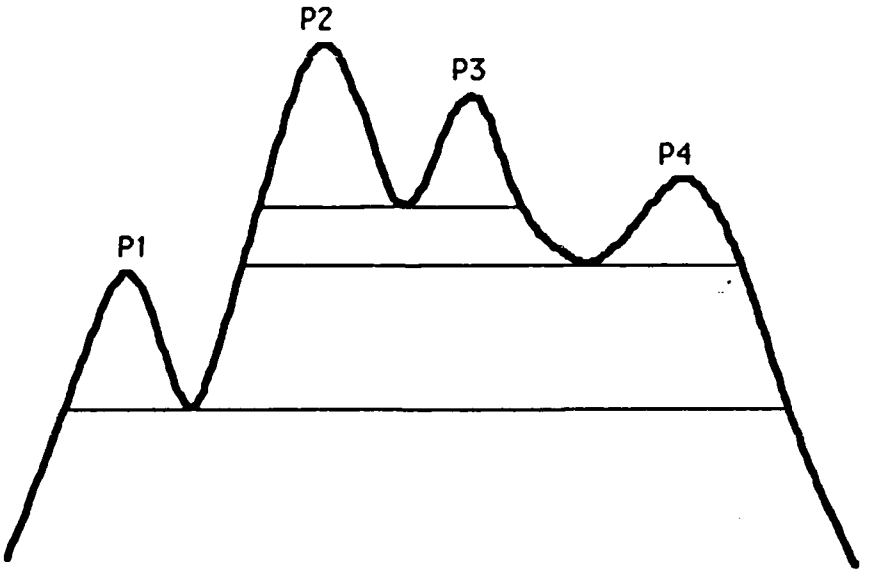


Figure 1.1 a
Waveform Segments

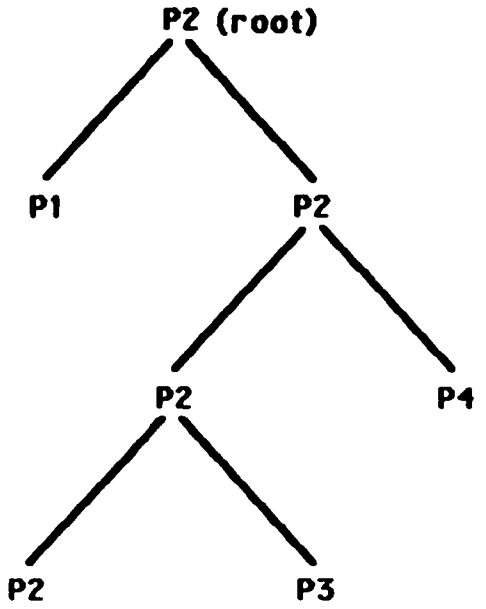


Figure 1.1 b
The Relational Tree

Erich and Foith list 8 properties of these trees [1]:

1) The frontier of the tree is a left to right description of the waveform,
2) If a peak label occurs at N nodes, then that segment has N-1 subpeaks,
3) Each parent node corresponds to a valley that separates two descendant peaks,
4) Each parent node has the same label as the descendant having the largest vertical height,
5) If a sequence of k valleys all have the same height, the corresponding node has $k+1$ descendants. If the left or right

dominant peak of a valley is not unique, the label of its node consists of all peaks with maximum height in that segment,

6) Although several nodes in the tree have the same name, their occurrences are different. The attribute list attatched to the node contains specific information,

7) RT's partition the set of one-dimensional functions into equivalence classes,

8) As one moves from frontier to root through nodes of the same label, relative peak heights will be strictly decreasing.

It is property 7 that is of most interest for this report. The partitions may be viewed as clusters in a feature space. The nature of the relational tree structure allows us to classify functions based on that structure.

Special cases occur when valleys or consecutive peaks within a segment have equal height. The convention is to represent these as n-tuples as described in property 5. If the resulting trees are to be recognizeable by finite tree automata, the number of descendants from a node must be limited to a finite number of choices. Erich and Foith describe a modified relational tree implementation which imposes a binary topology on the resultant trees. A similar implementation will be adopted for this work. Peak or valley dominance in these cases will be decided on the basis of position.

Although the relational tree is the structure adopted for this work, two more tree structures for representing waveforms should be of interest to the reader.

1.2 Alternate Tree Structures

Researchers have defined alternate tree structures for describing waveforms [2]. These more complex tree structures borrow from the relational tree concept. The results are trees whose topology represents time and amplitude information in addition to the relative placement of peaks and valleys.

Cheng and Lu [2] expanded the relational tree structure to take into account amplitude and time information. These alternative tree representations reduce the amount of semantic information which must be stored at nodes. In the skeletal tree , the waveform amplitude is quantized. An interval is delineated when the waveform crosses a quantization level boundary. Each node represents a pair of these crossings (see figure 2).

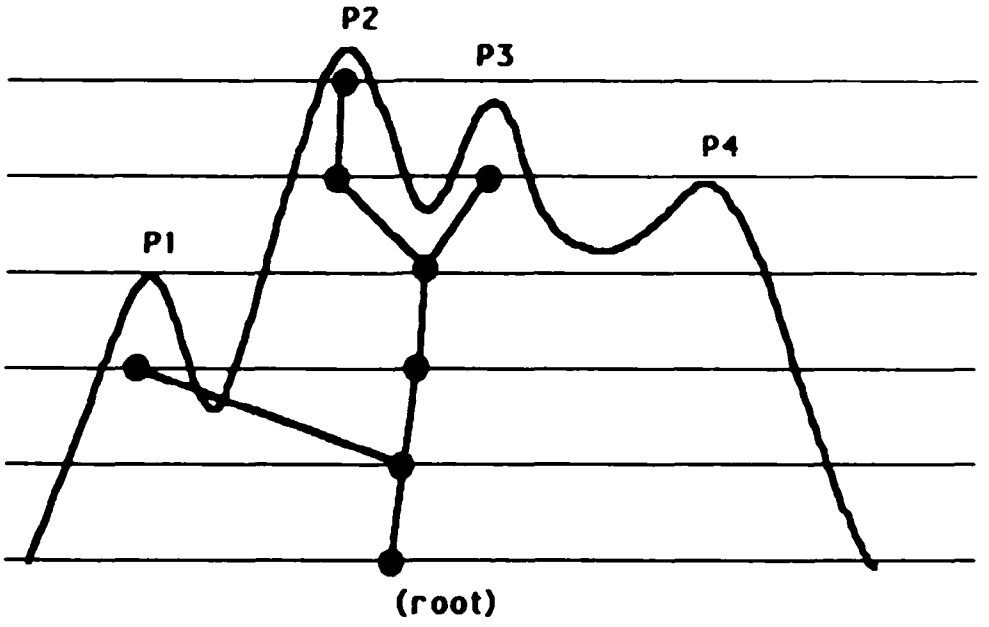


Figure 1.2
The Skeletal Tree

Cheng and Lu also describe the "complete tree" representation. For a complete tree, the structure reflects both waveform amplitude and interval widths. A complete tree is constructed by superimposing a two-dimensional grid over the waveform. Interval boundaries are decided by time line crossings (see figure 1.3).

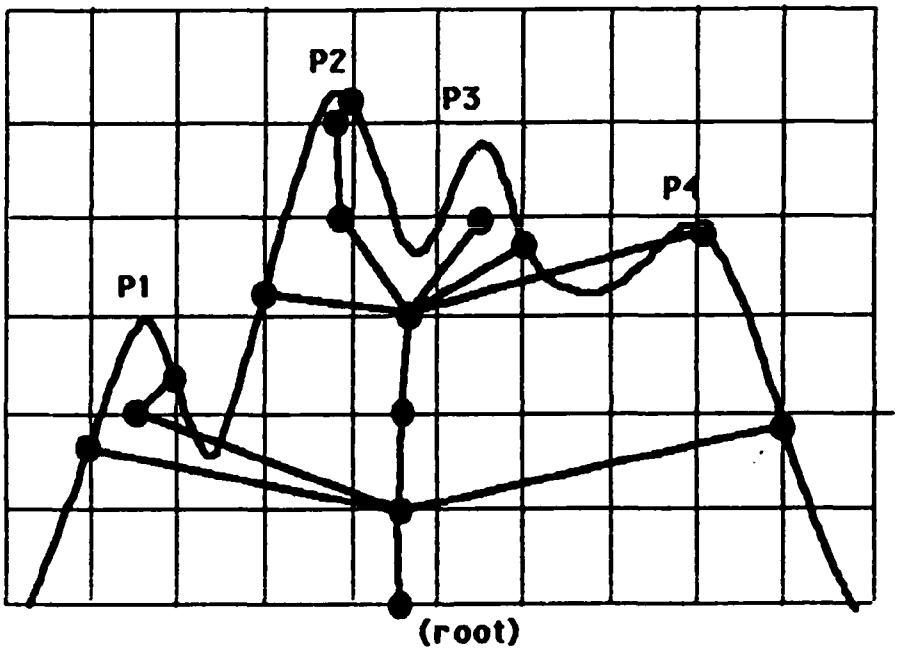


Figure 1.3
The Complete Tree

Skeletal and complete trees possess the following properties [2]:

1) The depth of a tree is equal to the number of quantization levels,
2) The leaves of the tree are peaks in the waveform. Peaks smaller than a quantization interval are not visible,
3) The depth of a leaf is equal to the amplitude of its corresponding peak,

4) The nearest common predecessor of two nodes corresponds to a valley on the waveform,

5) A node represents an interval on a quantization level,

In addition, complete trees have these properties:

1) The subtree containing all the non-terminal nodes, and only non-terminal nodes, is the skeletal tree for that waveform,

2) The original waveform can be reconstructed by tracing the leaf nodes from left to right.

Cheng and Lu used the complete tree to correlate waveforms. This was done by matching nodes between the trees representing two signals.

In order to perform operations on relational trees, we need a way to compare them. A number of techniques for calculating distances between trees have appeared in the literature. In the next section we will examine various author's approaches and assess their usefulness.

1.3 Review of Tree Distance Measures

The tree distance measures discussed here [2,4,5,6] all make use of he minimum number of some elementary transformation necessary to make one tree into another. The earlier tree distance measures [3], [4] rely on syntactic error correction schemes, whereas recently developed distances [5], [6] are independent of grammatical considerations.

Lu and Fu [3] propose a structure-preserved-error-correcting-tree-automata (SPECTA). This type of distance measure parses a tree and

compares it to a given grammar. Differences between two trees are viewed as syntactic errors. The parser finds the error correcting transformation involving the smallest number of node substitutions. The distance between two trees is the smallest number of node substitutions required to transform one tree into another while maintaining predecessor descendant relationships. If such a transformation exists, it is symmetric and satisfies the triangle inequality.

In another publication , Lu and Fu [4] developed a generalized error-correcting tree automata (GECTA). This distance measure is similar to SPECTA, except that node insertion and deletion errors are allowed in addition to node substitutions.

SPECTA and GECTA both require that the trees be described by some type of tree grammar. Neither distance measure is guaranteed to exist.

Another tree-to-tree distance algorithm is descibed by Lu [5]. This distance measure requires no knowledge of a tree grammar and employs insertion, deletion, and substitution errors. The distance $d_{ids}(\alpha,\beta)$, where α and β are trees, is defined as the minimum cost necessary to derive α from β such that:

1) The predecessor-descendant relation does not change,

2) Nodes in β do not split or merge,

3) The sequence of postfix ordering does not change after transformation.

Lu shows [6] that $d_{ids}(\alpha,\beta)$ can sometimes lead to anomalous values. For this reason, a new tree matching algorithm, $d_{sm}(\alpha,\beta)$, was proposed to

make use of node splitting and merging. The basic node operations in $d_{sm}(\alpha,\beta)$ are defined as father-son splitting, brother splitting, father-son merging, and brother merging. The distance $d_{sm}(\alpha,\beta)$ is the minimum number of these four operations necessary to derive α from β. This distance always exists and obeys the following properties [6]:

1) $d(\alpha,\alpha) = 0$,

2) $d(\alpha,\beta) = d(\beta,\alpha)$,

3) $d(\alpha,\beta) \leq d(\alpha,\gamma) + d(\gamma,\beta)$.

The next chapter provides a review of clustering as it has been used to recognize patterns which can be described by a feature vector. We shall focus on the specific techniques which are of use in clustering trees.

Chapter 2

Review of Feature Space Clustering

2.1 Traditional Cluster Seeking Techniques

There are many techniques available for partitioning a set of samples into clusters [8,9]. The non-parametric algorithms rely on a distance measure between samples. Instead of the Euclidian distance between vectors, we can subsitute an appropriate tree distance. These techniques include the k-means algorithm and hierarchical clustering.

The k-means algorithm [9] clusters samples by minimizing the sum of squared distances from cluster members to the cluster center. There is no proof of convergence for this algorithm. Its success depends on the value chosen for k, and the initial cluster configuration. If this algorithm is to be used for trees, the definition of a tree cluster center is required. Finding a cluster center is an awkward and time consuming problem.

Hierarchical clustering algorithms are best suited for clustering RT's. They are best in the sense that no vector operations (such as those required to define a cluster center) are required except distance between samples. The result of an hierarchical clustering procedure is a tree known as a dendrogram [8]. The frontier of the tree is made up of clusters containing a single sample, and the root represents the entire sample set. A parent node represents a cluster which is the union of the clusters represented by its immediate descendants. Clusters are grouped according to some distance criterion such as nearest-neighbor. A typical two-dimensional sample set

and its corresponding dendrogram are shown in figure 1.4. The samples are shown grouped according to nearest neighbor criteria.

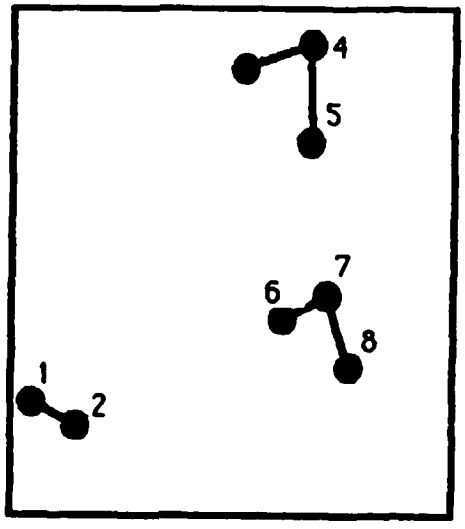


Figure 1.4 a
The Entire Sample Space

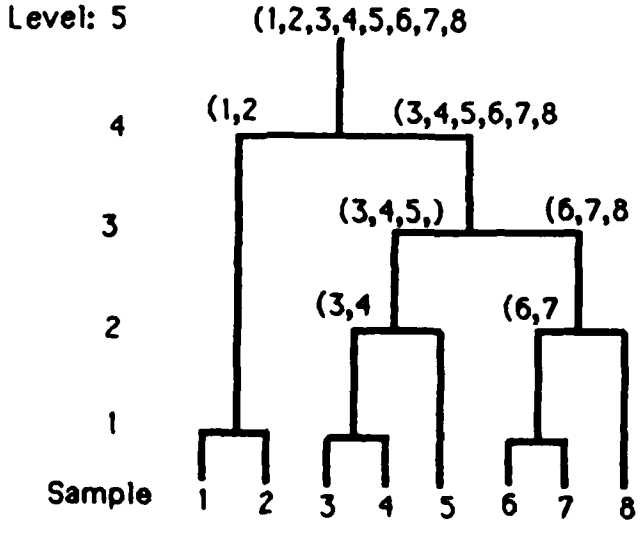


Figure 1.4 b
The Hierarchical Clustering Dendrogram

2.2 Syntactic/Semantic Clustering Techniques

Lu and Fu [7] proposed a cluster-seeking procedure for syntactic patterns. A grammar is inferred for each cluster in a sample set. If the distance from a sample to any cluster is above an arbitrary threshold, a new cluster is started. Otherwise, the sample is assigned to the nearest cluster and a new grammar is inferred for that cluster to account for the new sample. This is a very time consuming process since several new grammars must be infered each time a new sample is added to the set. Given a tree metric, any of the established minimum-distance clustering and classification techniques could be used. Lu [5] used a tree metric involving node substitutions, insertions, and deletions to hierarchically cluster handwritten characters. No attempt was made by Fu to evaluate or optimize clusters which were formed.

2.3 Sample Classification

Once the data are clustered, incoming samples may be classified according to a minimum distance criterion. This usually means assigning a sample to the cluster of its nearest neighbor. A variation on this is the k-nearest neighbor technique which assigns a sample to the cluster where a plurality of its k nearest neighbors lies. It may be that the resulting clusters are not ideal for classifying incoming samples. Individual clusters might be too scattered, or adjacent clusters too close together, or both. When this happens, we would like some way of modifying the clustering scheme so that each cluster contains the same samples, but with new clusters which are compact and well-separated (CWS). To this end, a criterion is needed which will indicate whether or not the clusters are CWS. The following sections

describe a transform method for modifying tree clustering, and an objective function for assessing the performance of such a transform.

Chapter 3

A New Tree Clustering System

3.1 Problem Statement

First consider the case where waveform segments are uncontaminated by noise. The problem we wish to address here involves a given a set of signals of known origin with similar peak and valley structures. These signals may have undergone non-uniform stretching or squeezing along the domain axis, but they can still be grouped together. Given another signal of unknown type, how can we determine whether or not the unknown signal belongs to the previously known set?

This paper proposes a solution to this problem which makes use of *relational tree structures and traditional pattern* recognition. The following sections show that the concepts of formal tree automata theory may be combined with the data structures and algorithms introduced in the previous chapters. This combination will produce a waveform recognition system which is insensitive to monotonic transformations of the horizontal axis.

3.2 Clustering Relational Trees

The previous section stated that relational trees partition the set of waveform segments into equivalence classes. If this is so, relational trees might provide a mechanism for classifying one-dimensional signals. Such signals include image scan lines, speech, or ECG records. Furthermore, a tree distance can be utilized to compare trees which are not identical. While there are ways to classify waveforms without resorting to syntactic/semantic

techniques, the special properties of relational trees may be exploited when certain types of signals are encountered.

When identifying patterns which can be described by some feature vector, this problem is commonly solved by cluster analysis. There is a large body of work pertaining to clustering patterns in an Euclidian vector space. Many of the published algorithms make use of matrix operations in this vector space. Since the samples considered here exist in a tree space, the only operation available is some prespecified distance function. This makes most traditional cluster analysis techniques less than useful. It is possible, however, to use the same concepts while modifying the specific technique somewhat. The methods discussed here are those that will have an application to clustering relational trees.

A relational tree space in which clusters are formed may be thought of as a directed graph. Each node represents a different tree. An edge exiting a node represents an elementary operation on that tree. The distance between two trees is the minimum path length on the directed graph from one tree to another tree. The subspace consisting of trees T1-T6 with metric $d_{id}(x,y)$ is shown in figure 3.1. Path "a" has length four, whereas path "b" has length two. The distance between trees T1 and T3 is therefore two edge traversals.

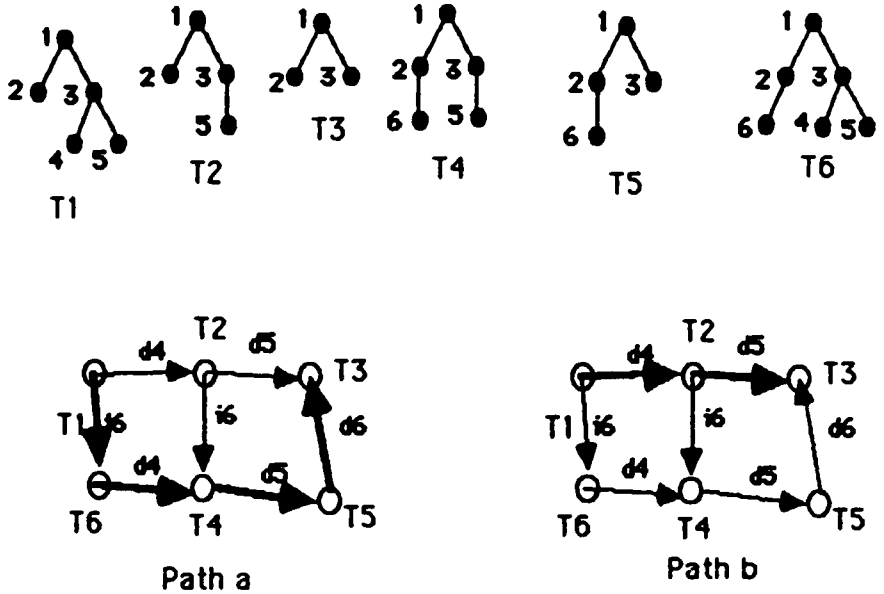


Figure 3.1
A Tree Subspace

Since the path from one tree to another is dependent on the particular type of elementary tree operation chosen, it is clear that the tree distance algorithm selected will determine the nature of clusters formed.

3.3 Theorems Concerning Relational Trees

In order to apply the concepts of tree automata theory to waveforms represented as relational trees, we have to formally assert certain facts about them. Among these facts are the existence of a relational tree, and if it can be recognized by a tree automaton.

Existence theorem:

For all one-dimensional, unipolar, bounded functions with a finite number of maxima and minima in a finite interval, there exists a relational tree description for that interval.

Proof:

Construct a tree by induction over M, the number of peaks in the segment.

Base case: M=1. This is a segment consisting of only one peak. Its relational tree desciption is a single node $t_1=p$.

Inductive Hypothesis: M=m. A tree describing a segment with m peaks can be written $t_m=\sigma(t_{m1},t_{m2})$, where t_{m1} contains m_1 peaks and t_{m2} contains m_2 peaks, and $m_1+m_2=m$.

Inductive Step: A new segment with m+1 peaks is formed by adding a peak p_{m+1} and a new valley on to the left of the segment with m peaks. The relational tree description t_{m+1} can be described recursively as follows:

If the new valley is lower than any of the valleys in tm, then

$$t_{m+1}=\sigma'(\sigma(t_{m1},t_{m2}),p_{m+1}).$$

else

$$t_{m+1}=\sigma(t_{m1},t_{m2+1}).$$

Corollary:

A relational tree representing a waveform segment with M maxima will have 2M+1 nodes.

If a tree is recognizable by a finite tree automaton, it is possible to perform certain operations on that tree via finite state machines. Later in this paper, we will want to transform relational trees according to some predefined node operations. The effects of such a transformation are well known if the tree being operated upon is recognizable.

Recognizability Theorem:

If relational tree descriptions are limited to binary trees as described in Erich and Foith's paper [1], then the forest of relational trees is a subset of those $\sum$, X trees which are recognizeable by a finite tree automaton.

Proof:

Prove by constructing a regular tree grammar for describing the relational trees. Since a $\sum X$-forest T(G) is formed by a regular tree grammar G, and $T \in Rec(\sum,X)$ for every $\sum X$-forest, the resulting trees will be recognizable. The grammar G is as follows:

$G=(N,\sum,X,P,a_0)$, where:

$N = \{a\}$, a non-terminal alphabet

$\sum = \sum_2 = \{\sigma\}$, a ranked alphabet

$X = \{p\}$, a terminal alphabet

$P = \{a \Rightarrow p,\ a \Rightarrow \sigma(a,a)\}$, a set of productions

$a_0 = a$, a starting symbol

Example:

A leftmost derivation of the relational tree shown in figure 1.1b is:

$$a \Rightarrow \sigma(p1,a) \Rightarrow \sigma(p,\sigma(a,a)) \Rightarrow \sigma(p,\sigma(\sigma(a,a),a)) \Rightarrow \sigma(p,\sigma(\sigma(a,a),a))$$
$$\Rightarrow^* \sigma(p,\sigma(\sigma(p,p),p)$$

If the nodes are labeled as shown in figure 1.1, the tree grammar becomes context-sensitive and is no longer recognizable. We use the term context sensitive because a the assignment of a sequential peak label requires knowledge of the surrounding peak labels which have already been fixed. To

avoid such context sensitivity, we will adopt a node labeling scheme which is based on peak height rather than sequence.

3.4 Tree Cluster Improvement:

We wish to find a transformation $T(x| x\epsilon X)$ on the set of sample trees X such that the probability of sample misclassification is minimized. In order to do this, we will define an objective function which is to be maximized. A tree transform mechanism and two methods for finding the objective function, $J_1(X,T)$ and $J_2(X,T)$, are presented in the next section.

3.5 A Tree Transformation

In traditional feature space clustering, the objective function is often optimized by finding an appropriate linear transformation on the feature space. This can be reduced to a simple unconstrained minimization problem. Since we lack such tools as matrix multiplication when dealing with trees, finding the proper transformation to improve cluster seperation becomes a search problem. A tree transformation is based on a tree tranducer. The following formal definition of a tree transformation lays the foundation for ameliorating clusters in a tree space. A frontier-to-root tree transducer U is a seven-tuple [11]:

$$U=(\sum, X, A, \Omega, Y, P, A')$$

$\sum$, Ω are ranked alphabets
X, Y are frontier alphabets
A is a state set
A' is a set of final states
P is a finite set of productions or rewriting rules of the following type
i) $\{x=>a(q) \mid (x\epsilon X, a\epsilon A, q\epsilon F_{\Omega}(Y))\}$

ii) $\sigma(a1(S1), \ldots, am(Sm)) \Rightarrow$
$a(q(S1, \ldots, Sm))$.
$\sigma \in \sum m$, $m \geq 0$,
$a1, \ldots, am, a \in A$,
$q(S1, \ldots, Sm) \in F(Y \cup \Xi m)$,
$\Xi m = \{S1, \ldots, Sn\}$ = a set of auxilliary variables that indicate the occurance of a subtree in a tree.

T_U, then, is the tree transformation induced by U:

$$T_U = \{ (p,q) \mid p \in F_{\sum}(X),\ q \in F_{\Omega}(Y),\ p \Rightarrow^* aq,\ a \in A' \}$$

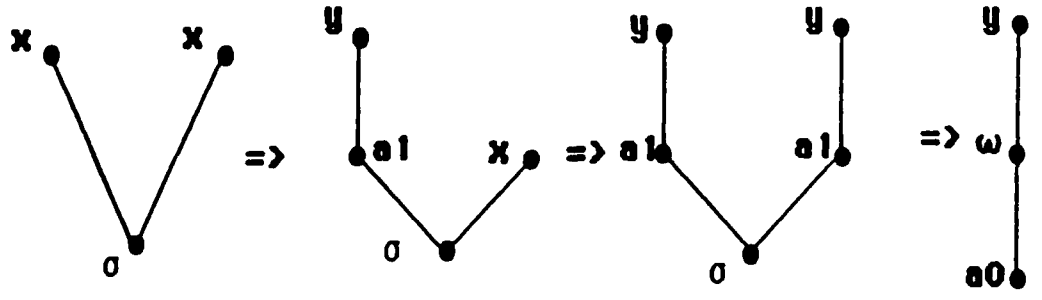
The rewriting rules are the variables of the transformation. A transformation is tailored to a specific aplication by choosing th proper rules. These rules represent mappings between subtrees in the input forest, $F_{\sum}(X)$, and the output forest, $F_{\Omega}(Y)$. States are placed in the tree during transformation to guide the rewriting rule application.

Example of a Tree Transformation

$$U = (\sum, \{x\}, \{a_0, a_1\}, \Omega, \{y\}, P, \{a_0\})$$

$$\sum = \sum_2 = \{\sigma\},\ \Omega = \Omega_1 = \{\omega\}$$

$$P: x \Rightarrow a_1 y,\ \sigma(a_1(S_1), a_1(S_1)) \Rightarrow a_0 \omega(S_1)$$



A more complex and useful tree transform example is shown in the next chapter.

3.6 A Tree Clustering Objective Function

Once the data undergo a transformation, the effect on cluster compactness and separateness must be assesed. Both objective functions presented here will utilize the ratio of between-class scatter to within-class scatter.

Method 1

For the two class problem, we define the within-class scatter for a transformed cluster of size $|Y_i|$ as:

$$s_i = \frac{1}{|Yi|} 2 \sum_{y \in Yi} \sum_{x \in Yi} d(y,x)$$

and the between-class scatter as,

$$S_B = \frac{1}{(|X1||X2|)} \sum_{x1 \in X1} \sum_{x2 \in X2} d(x_1,x_2)$$

X_i and Y_i are transformed clusters of trees. $|X_i|$ is the number of sample trees in X_i. $d(x,y)$ is some metric between trees. The objective function for the two class case is:

$$J_1(X,T) = \frac{S_B}{(s_1 + s_2)}$$

In order to generalize to c clusters, we must define the total within-class scatter as:

$$S_W = \frac{1}{c|X_i|^2} \sum_{X_i \in X} \sum_{y \in X_i} \sum_{x \in X_i} d(x,y)$$

and the between-class scatter as:

$$S_B = \frac{1}{c|X_i|(|X|-|X_i|)} \sum_{X_i \in X} \sum_{x_i \in X_i} \sum_{x \in X, x \notin X_i} d(x_i,x)$$

where X is the entire tree sample set after applying the transformation T. The objective to be maximized for the case where there are more than two clusters is:

$$J_1(X,T) = \frac{S_B}{S_W}$$

Method 2

As an alternative method, consider the following definition of a cluster center for syntactic patterns [10].

First, define the β-metric for a tree x_{ij} in cluster X_i to be:

$$\beta_{ij} = (1/|X_i|)\sum_{l=1 \text{ to } |X_i|} d(x_{ij},x_{il})$$

Let x_{ij} be the cluster center of X_i, denoted by m_i, if its β-metric is minimum over the cluster, i.e.

$$\beta_{ij}=\min_l\{\beta_{il} \mid 1<l<|X_i|\}$$

Using this definition of a cluster mean, let the within-cluster scatter be the sum of β-metrics for each cluster center squared:

$$S_W = \sum_{i=1}^{c} \beta_{ij}^2, \qquad m_i = x_{ij}, \text{ for all } i$$

Define the total mean m to be the median sample tree over the entire sample set:

$$m = \{x \mid \sum_{y\in X, y\neq x} d(x,y) \text{ is minimum } \forall\, x\in X\}$$

The between-class scatter is:

$$S_B = \frac{1}{c} \sum_{i=1}^{c} d(m_i,m)$$

Now maximize the objective function $J_2(X) = S_B/S_W$ using these expressions for S_B and S_W.

3.7 Transform Application

Assuming the input and output alphabets of the transformation are fixed, an optimal transformation may be found over different combinations of rewriting rules. The transform designer searches for an optimal set of rewriting rules. The search is performed over a candidate rule set. Candidate rules are application specific and determined a priori. In the following chapter we show how such rules are formulated. In a step by step fashion, the waveform clssification algorithm is:

1) Obtain a training set of waveforms,
2) Convert to their relational tree representations,
3) Cluster these trees in a metric space (directed graph),
4) Find a transformation on these trees such that the objective function described previously is minimized,
5) Convert candidate waveforms to their relational tree representation,
6) Apply the transformation found above to the candidate trees,
7) Classify the candidate trees by a nearest neighbor technique.

The block diagram shown in figure 3.2 depicts such a waveform classification scheme.

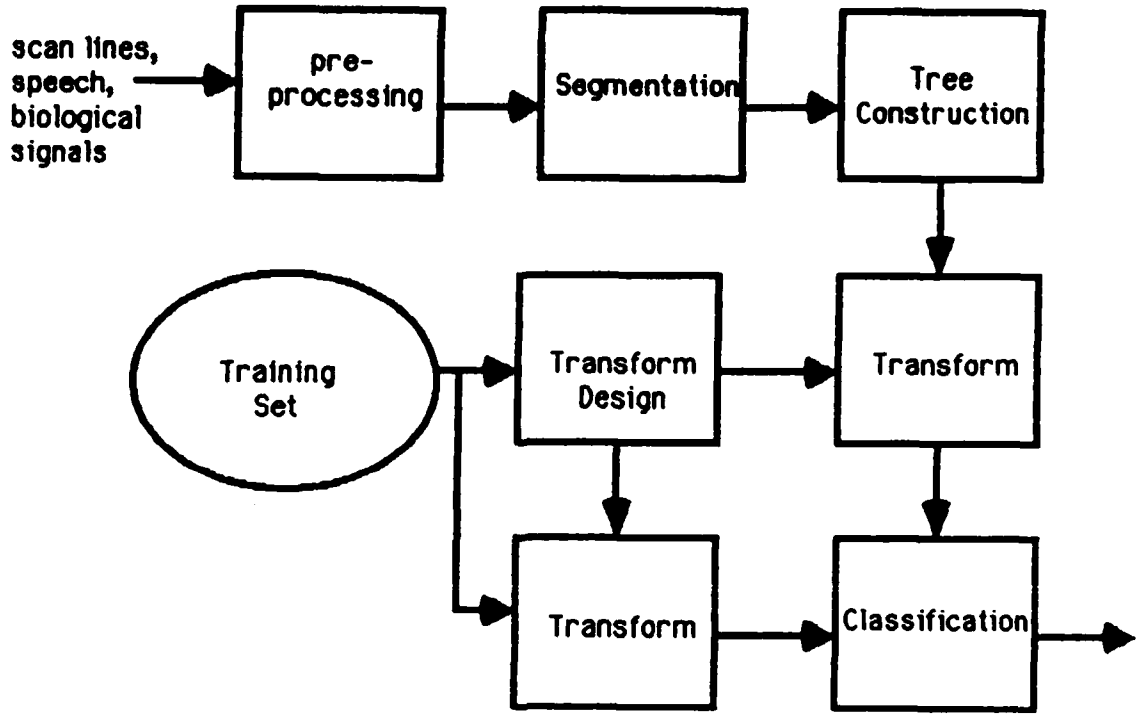


Figure 3.2
Waveform Recognition System

Chapter 4

Tree Transform Implementation

The examples of structures and algorithms discussed in the preceding sections represent simple and ideal cases. When adapting these techniques to actual problems, some additional constraints and modifications are necessary. Specifically, we need to discuss relational tree labeling, and the details of the transform and optimization algorithms.

4.1 RT Node Labels

The node labeling convention shown in figure 4.1 is to identify a peak by its relative sequential postition. This is unsatisfactory when small errors in segmentation occur. Two waveforms may be structurally similar, but adding a single unwanted peak to the front of one waveform will cause its entire relational tree to be labeled differently from the other. This leads to a falsely large distance between the two waveforms.

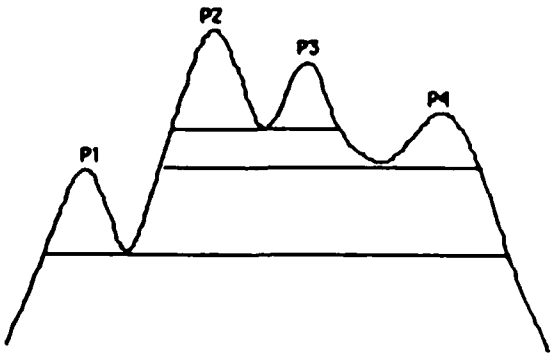


Figure 4.1
Sequential peak labels

The node labeling convention adopted for this research is to label peaks by their relative size within the waveform segment (see figure 4.2). Peak heights are scaled and quantized to L levels. When labeled in this manner, all relational trees will have root label P_{L-1}. The smallest peak in a segment will have label P_0. This labeling scheme leads to non-unique labels for peaks.

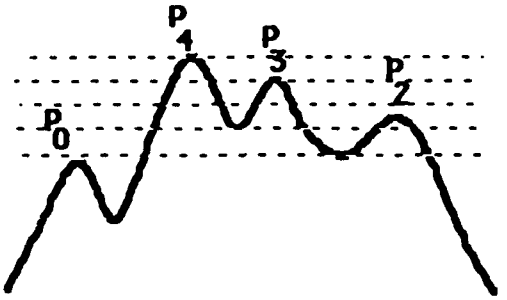


Figure 4.2
Modified labeling scheme.

This type of labeling requires a new and more complicated grammar. Such a grammar G for L peak quantization levels is as follows:

$$G=(N,\Sigma,X,P,a_0)$$

$$N = \{\alpha_n \mid n = 0, 1, \ldots, L-1\}.$$

$$\Sigma = \{P_n \mid n = 1, 2, \ldots, L-1\}.$$

$$X = \{p_n \mid n=0, 1, \ldots, L-1\}.$$

$$P = \{\ \alpha_n \Rightarrow P_n(\alpha_n,\alpha_m),\ 0<m<n \mid$$
$$P_n(\alpha_m,\alpha_n),\ 0<m<n \mid$$
$$p_n\ \}.$$

$$a_0 = \{\alpha_{L-1}\}.$$

An example derivation is shown in figure 4.3.

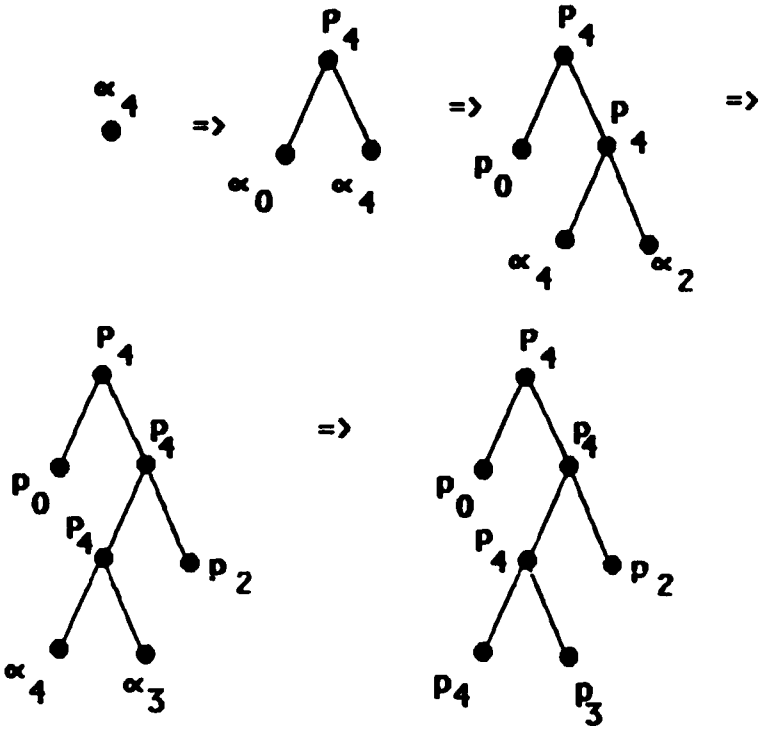


Figure 4.3
A sample relational tree derivation.

4.2 Transform Implementation

The purely theoretical transform example given in the previous chapter is not sufficiently complex for a practical application involving relational trees. Some correlation must exist between the transform rewriting rules and the effects those rules have on the underlying waveform. The user of a waveform classification system is not concerned with tree manipulations as long as clusters are formed that reflect structural similarities between waveforms. Rewriting rules cannot be considered as individuals due to the interaction amongst them. These rules may be grouped according to their combined effects on a waveform. A subset of states and rewriting rules which are intended to affect one particular subtree, will

henceforth be referred to as an operation. As an example, consider the following transform:

Locate and replace all occurrences of segments consisting of the two peaks p_4, p_5, with the single peak p_5. The operation to perform this on a relational tree is the set of states Q:

$$Q = \{q_4, q_5\}$$

and rewriting rules P:

$$P = \{p_4 \Rightarrow q_4 p_4,\ p_5 \Rightarrow q_5 p_5,\ P_5(q_4, q_5) \Rightarrow q_5 p_5\}$$

Figure 4.4 shows the results of applying this transform to the tree $P_5(p_4, P_5(p_4, p_5))$.

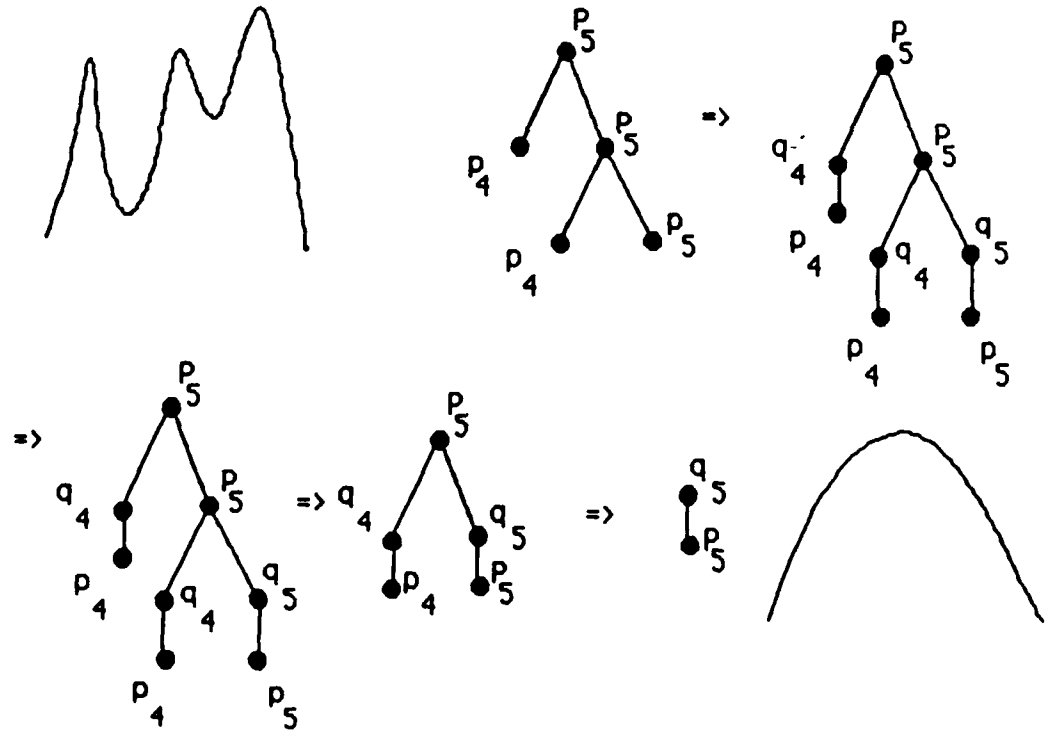


figure 4.4
A transform involving one operation.

4.3 Tree Transform Rule Selection

Rules must be chosen so that they are consistent and non-ambiguous. For example, individual operations must not interfere with each other. It is also necessary that no two rules in a transform share the same left-hand side. This constraint prevents ambigous transforms. When transforming relational trees, the proper peak dominance relations must remain intact. To accomplish this, any relational tree transform must include rewriting rules that maintain peak dominance conventions when a node's descendants have been altered. These rules should relabel the node so that the proper peak is dominant.

The tree transform algorithm as implemented here accepts input from the user in the form of states and rewriting rules. Beginning with the leaves of the tree, each node is compared to the left-hand sides of the rule list. If a match is detected, the proper action is taken. If no match is detected, the node is checked for peak dominance. This step frees the user from having to specify rules to maintain proper peak dominance. In practice, the system reserves two states: CHANGE and NOCHANGE. If no rule is matched at a node, and either the node is a leaf or all the states descendant from the node are NOCHANGE, then the subtree is propagated intact with state NOCHANGE. If one of the descendant states is CHANGE, peak dominance is checked and the subtree is propagated with state CHANGE. Care must still be taken that user specified rules preserve peak dominance.

4.4 Search Algorithm

The algorithm which searches for an optimum transform is a critical aspect of the classification scheme. This is a very time consuming procedure, and any modifications which reduce the algorithm's complexity will result in great cost savings.

If there are n compatible operations, the result is a worst case search time complexity of O(M), where

$$M = \sum_{i=0}^{n-1} \frac{n!}{i!(n-i)!}$$

different combinations of operations.

Unless there is some way to predict the outcome of a tree-matching (tree distance) algorithm, J(X,T) must be computed for each combination of operations. If the algorithm can be implemented on parallel processors, the search time could be reduced.

Tree distance computations are the most costly portion of the search algorithm. In the case where particular trees are known a priori to be the desired cluster centers, the number of distance computations nescessary at each search node may be reduce considerably by using the objective function $J_2(X,T)$. If there are n samples, the objective function $J_1(X,T)$ requires T_1 tree distance computations for each combination of operations, where

$$T_1 = \frac{n(n-1)}{2}$$

With $J_2(X,T)$, only $T_2 = n+c$ distance calculations are necessary, where c is the number of clusters. It is simple to show that for all n greater than 5, $T_2 < T_1$. Therefore, a search using the objective function $J_2(X,T)$ is faster than one utilizing $J_1(X,T)$, so long as no cluster center computations are required.

4.5 An Alternate Implementation

A sequential language such as Fortran, C or Pascal is well suited for implementing the numerical computations one encounters in signal processing. Researchers are accustomed to thinking sequentially and the field of waveform recognition has grown up around this sequential and deterministic paradigm. So far, we have assumed the proposed relational tree waveform recognition system is to be implemented in a traditional Von Neumann computer language. Since this is the "natural" way to program the current generation of computers, such an approach is probably appropriate. However, the operations involved in our waveform classification system are radically different from typical signal processing tasks. Our techniques are symbolic rather than numerical. They are reasoning rather than computational. It is awkward to express reasoning with a Von Neumann language. In addition, operations on trees need not be sequential. Instead of transforming a waveform by operating on one digitized sample at a time, as in conventional signal processing, tree transformations possess inherent parallelism. While computers are currently available to execute the tree transformation in parallel, resulting in considerable time savings, parallelism is difficult to express in a Von Neumann-type language. With these considerations in mind, a more convenient way to implement our waveform classification system is sought.

Since we are trying to imitate the human capacity for detecting structural similarities in waveforms, the relational tree classification system may be characterized by the catch-all phrase "Artificial Intelligence". Workers in this field have found two alternatives to Von Neumann languages that are well-

suited to expressing this type of computation. There are functional languages such as Lisp, and logic programming languages such as Prolog. Lisp operates solely on lists. Prolog, however, has more flexible typing, making it easier to represent trees with greater than binary branching factor. Prolog is also attractive in that it operates in a similar fashion to our tree transformation. In Prolog, knowledge is represented descriptively in a database which may be easily modified, while a separate inference engine uses that knowledge in a methodical manner. The inference engine is the same for all applications. In our system, the tree transform rewriting rules are expressed as facts in a user modifiable text file. These facts change according to the application. The tree transform procedure, itself a type of inference engine, is never altered. Finally, Prolog seems the best choice for an implementation language since it is quite easy to express parallelism. Parallel implementations of Prolog, such as Parlog, already exist [12].

Although this paper is not intended as an introduction to Prolog, a brief description is given here. a more thorough treatment may be found in Clocksin and Melish [13]. Prolog is a system for doing inference using clausal logic. A Prolog program consists of rules and facts in the form of Horn clauses. These clauses reside in the Prolog database. The user presents queries to Prolog, also in the form of Horn clauses. These queries are known as goals. Prolog decides whether or not a goal clause is consistent with the rules and facts in the database, and instantiates whatever variables are required to make the goal true. The inference method used by Prolog to make these decisions is known as resolution.

The waveform recognition system presented here has been implemented in the programming language C. The tree transform program as written is a complex procedure with many address comparisons and multiple indirections. The Prolog implementation of a tree transform is much simpler than its counterpart in a sequential language. In addition, the Prolog version has the flexiblity to work "backwards". For instance, one could instantiate the new tree and ask Prolog to find the tree transform which led to it. The suggested Prolog program is listed below. Tree nodes are represented by the function symbol node(Label, desclist), where desclist is the list of descendent nodes. Transform rules are in the database in the form:

rule(lbl,statelist,new_label, newstate,desclist),

where a node labeled lbl with descendent states given by statelist is transformed to a node with label new_label state new_state, and has descendants given by desclist.

nodexform(old node, new state, new node)
This predicate transforms a node and assigns it a state

```
nodexform(node(Lbl,Ndlist) ,State,node(Newlbl,Newndlist2)) :-
                        xformchildrn(Ndlist,Newndlist1,States),
                        rule(Lbl,States,Newlbl, State,Childlist),
                        selectchildrn(States,Newndlist1,Childlist,Newndlist2).
```

xformchildrn(list of nodes, new list of nodes, list of states)
This predicate transforms a list of nodes

```
xformchildrn([ ],_,_).
xformchildrn([Node|Nlist],[Newnode|Newnlist],[State|Slist]) :-
                                        nodexform(Node,State,Newnode),
                                        xformchildrn(Nlist,Newnlist,Slist).
```

selectchildrn(state list, node list, child list, new node list)
This predicate selects the new list of children for the transformed node given Clist from the transform rule.

```
selectchildrn(_,_,[ ],_).
selectchildrn([S|Slist],[N|Nlist],[C|Clist],[N|Newnlist]) :- S=D.
selectchildrn(Slist,Nlist,[C|Clist],Newnlist) :-
                                        selectchildrn(Slist,Nlist,Clist,Newnlist).
```

The program listed above expresses the parallelism inherent in the tree transform algorithm. The descendants of any node may be processed simultaneously, because they are not dependent on each other in any way. If, in the predicate xformchildrn(), the elements of Nlist could be processed in parallel, the complexity of the algorithm would be reduced substantially.

Chapter 5

Applications

In order to demonstrate the usefulness of the concepts and algorithms developed in the preceeding chapters, we will apply them to some real problems encountered in waveform recognition. The areas examined will be reflection seismic exploration for stratigraphic anomalies, identification and classification of electrocardiogram abnormalities, and the interpretation of stereo images.

5.1 Review

Let us reiterate the concepts which have led to the waveform recognition system. Just as patterns can be represented as strings over some grammar, waveforms can be described by a relational tree. Valley information is stored at non-terminal tree nodes, and peak information is stored at the leaves. A left-to-right traversal of the leaves yields a left-to-right description of the peaks in the waveform. An attractive feature of relational trees is that they are insensitive to scaling along the horizontal axis of the waveform . If certain restrictions are imposed on relational trees, they may be represented by a tree grammar. The set of all relational trees is the forest of trees described by such a grammar.

Just as the Levenstein distance is used as a metric between strings, there is a tree distances by which we can examine the similarity of two trees. Such a distance allows the relative positioning of relational trees in a tree space. A tree space is a directed graph where each relational tree occupies

a unique node. The distance between two trees is the path along the graph from one to another. Using this concept, we can "cluster" relational trees in such a space. Hopefully, similar waveforms would generate relational trees which lie close to each other in tree space. Using the well-known techniques of cluster analysis, we can treat relational trees much as a vector in pattern space. This immediately provides us with two very useful operations. The first, cluster-seeking, may be used to segment a set of unknown waveforms into classes. These classes should reflect some kind of underlying structural similarity. The second, classification, allows us to recognize individual unknown waveforms as belonging to one of a number of known classes.

Two criteria for assessing clustering performance have been introduced. They are intended to convey the degree to which clusters are compact and well-separated. Both rely on the ratio of between-class scatter to within-class scatter. The first requires many costly tree comparisons, but requires no knowledge of a cluster center. The second makes use of a prototype tree which serves as a cluster center. This reduces the complexity of the objective function computation substantially.

The tree transformation may be used to improve cluster performance. A tree transformation maps the forest of relational trees to another forest which is a subset of the relational trees. The desired subset has clusters which are compact and well-separated. A tree transformation is sought which improves a given clustering criterion. The tree transformation relies on a set of node rewriting rules. It is by varying these rules that tree transforms are tailored to a specific application.

A waveform recognition system which exploits the properties of relational trees might be constructed as follows: Training sets of waveforms are converted to relational trees and then transformed such that the clusters are compact and well-separated. Candidate waveforms are transformed using the same set of rewriting rules, and then classified based on their proximity to an existing cluster.

5.2 Seismic Example

Figure 5.1 shows a typical seismogram of a thin sand imbedded in a shale.

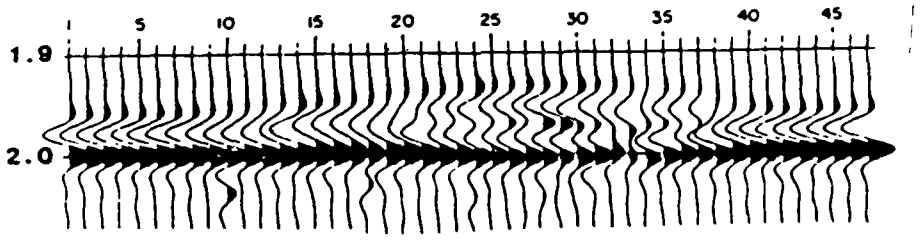


Figure 5.1
Seismogram over a sand channel (after Neidell [14])

This particular stratigraphic configuration forms an oil trap and therefore has great economic significance. An expert seismic interpreter can easily spot such an anomaly. A wavelet with a single peak and a trough becomes a doublet over the sand lens. It is not so easy for a machine to make such a qualitative judgement. Due to varying frequencies, noise contamination, changing bed thickness, and segmentation errors, a purely numerical algorithm technique, such as a matched filter, may not succeed in identifying those traces that contain sand. However, this is an ideal two-class relational tree clustering problem.

A seismogram over a known sand lens will serve as a training set of waveforms. From the corresponding relational trees, a tree transform can be found which improves clustering performance based on the objective functions described earlier. Seismic traces from unknown geology may then be compared to the two clusters and classified as belonging to the cluster of their nearest k neighbors. Figure 5.2 depicts schematically the two waveforms in question and their relational trees.

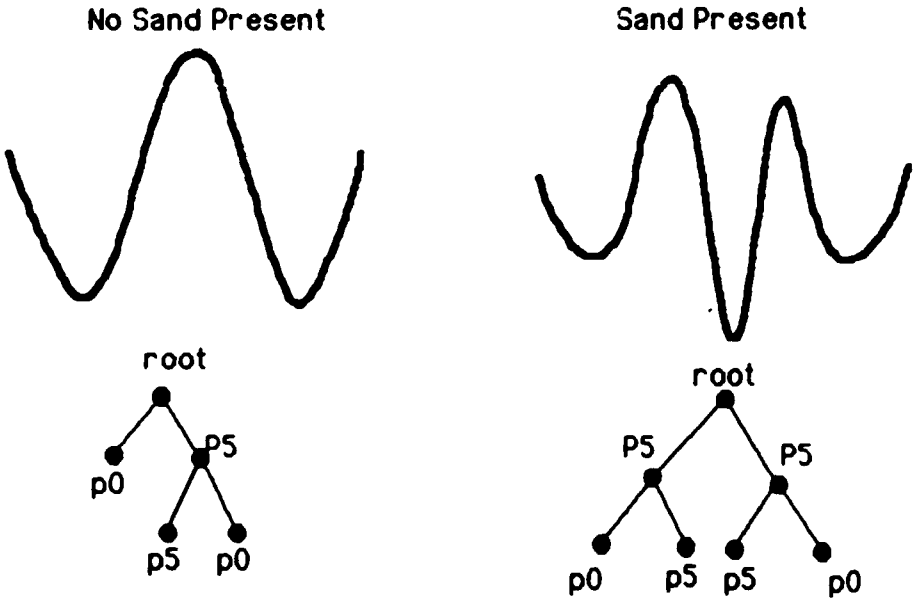


Figure 5.2
Seismic Waveforms and Relational Trees

Variations of these trees will occur due to noise, varying geology, etc.. The tree transformation will be designed to eliminate these inhomogeneities as much as possible. The rates of successful classification will be compared before and after transforming the tree clustering space by simulating the waveforms, distorting them, and adding noise, then applying the above procedure.

The number of peak quantization levels used was six. This provided reasonable identification of critical peaks, while keeping the tree grammar small enough to work with. The candidate rule set chosen for this application is shown in table 5.1.

```
operation:
p0 => a0(p0);
p1 => a0(p0);
p2 => a0(p0);
p3 => a0(p0);
p4 => a4(p4);
p5 => a5(p5);

operation:
P0(aA,aA) => a0(p0);
P1(aA,aA) => a0(p0);
P2(aA,aA) => a0(p0);

operation:
P3(a3,a0) => a8(P3(s0,s1));
P3(a0,a3) => a8(P3(s1,s0));
P3(a8,aA) => a8(P3(s0));
P3(aA,a8) => a8(P3(s1));
P3(aA,aA) => a3(p3);

operation:
P4(a4,a0) => a7(P4(s0,s1));
P4(a0,a4) => a7(P4(s1,s0));
P4(a7,aA) => a7(P4(s0));
P4(aA,a7) => a7(P4(s1));
P4(aA,aA) => a4(p4);

operation:
P5(a5,a0) => a6(P5(s0,s1));
P5(a0,a5) => a6(P5(s1,s0));
P5(aA,a6) => a6(P5(s1));
P5(a6,aA) => a6(P5(s0));
P5(aA,aA) => a5(p5);

operation:
P6(a0,a6) => aC(P6(s1,s0));

operation:
```

```
P6(a7,a6) => aC(P6(s1,s0));

operation:
P6(a8,a6) => aC(P6(s1,s0));
```

Table 5.1
Selected tree-transform operations

The effect of operations 2 through 6 was to propagate a desired subtree once it is identified. The state labeled aA, which appears in the right hand side of the rules, is the wild card state. Since the rule list is scanned from the top down, rules containing this state can be viewed as a default situation.

5.3 Seismic Classification

Waveforms were classified via a k-nearest neighbor scheme [8]. Training sets consisted of five samples of each waveform. A value of k=3 was found to be effective for classification. A single nearest neighbor scheme was found to be ineffective because an outlier from an incorrect cluster may be the closest to a given waveform.

5.4 Seismic Classification Results

Waveforms were simulated using cubic splines. The contaminating noise was bandlimited. The resulting trees before and after transforming are shown in appendix. Table 5.2 gives the results of transformation and classification for various signal to noise ratios. Signal to noise ratio is listed, followed by the objective function J_1 before transformation, the objective function J_1^* after transformation, and the observed error rate $Pr[\epsilon]$. The errors made were all misses, i.e. the classification of a sand trace as a trace without sand.

S/N	J1	J1*	Pr[ϵ]
30db	∞	∞	0
20db	0.097	1.11	0
10db	0.16	1.54	0.125
5db	0.18	1.38	0.187
0db	0.16	0.94	0.500
-5db	0.12	0.88	0.500

Table 4.2

5.5 Seismic Segmentation

Another useful clustering operation is one that divides a group of unknown patterns into clusters. A cluster-seeking system may be used to segment a seismic section by forming clusters of similar wavelet structure. A hierarchical clustering scheme such as the one discussed in previous chapters was utilized to form clusters from the simulated seismic data. The resulting dendrogram using nearest-neighbor clustering is shown in figure 5.3. The tree samples used were from the training set with 10db SNR. Samples 0-4 were signals from traces without sand, while signals 5-9 were from traces over sand lenses.

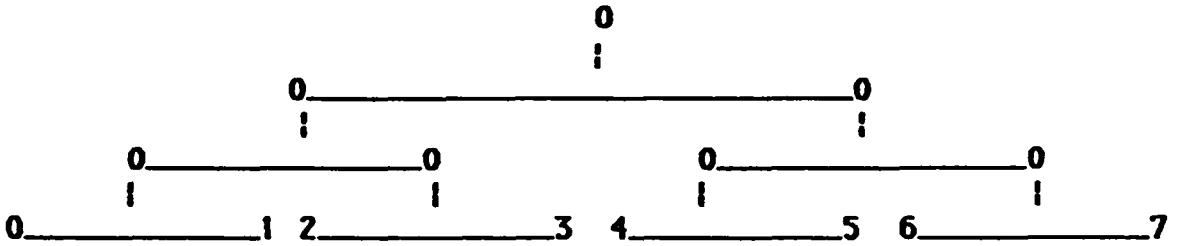


Figure 5.3
Dendrogram for Seismic Segmentation

5.6 ECG Example

We shall now address the problem of interpreting electrocardiogram (ECG) waveforms. An ECG may be divided into three parts: The P wave, the QRS complex, and the T wave [15]. A normal ECG waveform is shown in figure 5.4.

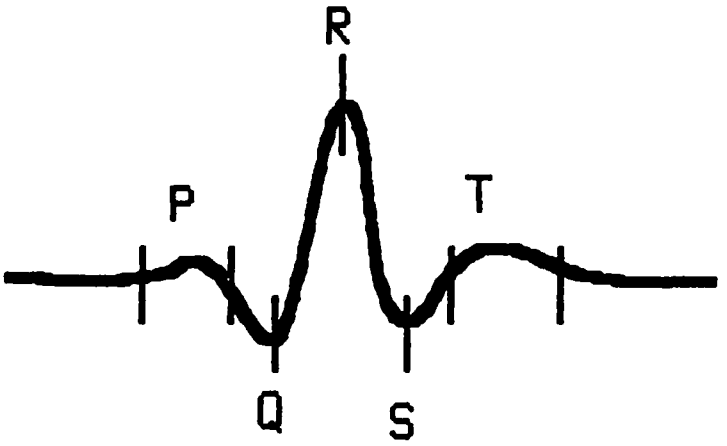


Figure 5.4
The Normal ECG

We wish to classify a subset of ECG abnormalities resulting from supraventricular arrhythmias. These are characterized by changes in the P

wave. Specifically, we wish to determine whether a given ECG exhibits an inverted P wave associated with premature atrial contraction (PAC), an absent P wave resulting from atrial tachycardia, or is normal.This is essentially a three-class detection problem. The waveforms in question are shown in figure 5.5.

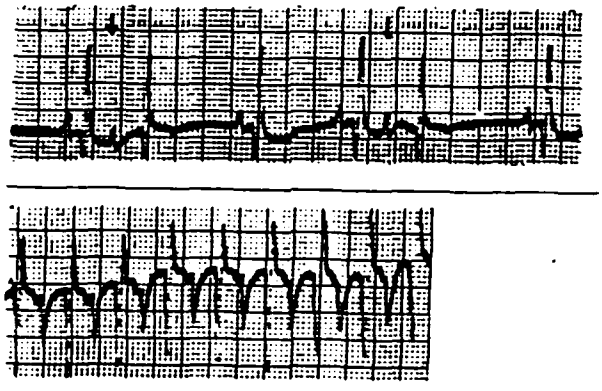


Figure 5.5
Abnormal ECG patterns (after Ganong [15])

5.7 ECG Classification

As in the previous example, waveforms were simulated with additive colored gaussian noise. In order to capture the inhomogeneities encountered when interpreting ECG's in real situations, variations were allowed in PQ interval, QRS duration, and ST interval. The training set was then transformed to improve the objective function J_1*. The set of transform rewriting rules which was found to accomplish this is given in table 5.3:

operation:
p0 => a0(p0);
p1 => a1(p1);
p2 => a2(p2);
p3 => a3(p3);
p4 => a4(p4);
p5 => a5(p5);

```
operation:
P0(aA,aA) => a0(p0);

operation:
P1(aA,aA) => a1(p1);

operation:
P2(a1,a2) => a6(P2(s0,s1));
P2(a6,aA) => a6(P2(s0));
P2(aA,a6) => a6(P2(s1));
P2(aA,aA) => a2(p2);

operation:
P5(a0,a5) => a5(p5);
P5(a5,a0) => a5(p5);
P5(a1,a5) => a7(P5(s0,s1));
P5(a2,a5) => a7(P5(s0,s1));
P5(a7,aA) => a7(P5(s0));
P5(aA,a7) => a7(P5(s1));

operation:
P5(aA,aA) => a5(p5);
```

Table 5.3
ECG transform rules

The unknown incoming ECG signals were transformed in the same way and classified by a k-nearest neighbor scheme with k=3. The resulting relational trees for examples of all three clusters before and after transform are shown in the appendix.

5.8 ECG Results

The results of implementing the ECG recognition system are shown in table 5.4. The system performs well only at high signal to noise ratios.

S/N	J_1	J_1^*	Pr[ε]
20db	0.198	1.271	0
15db	0.174	0.478	0.081
10db	0.159	0.499	0.33
5db	0.179	0.302	0.581
0db	0.169	0.326	0.581

Table 5.4
ECG Classification Results

5.9 ECG Cluster Seeking

We may also use hierarchical cluster-seeking to segment a group of ECG waveforms. Application of a nearest-neighbor cluster seeking algorithm resulted in the dendrograms shown in figures 5.10 and 5.11. The 10db training set was used. Figure 4.10 shows a two-class example where normal ecg's are separated from those with PAC. Samples 0-3 were normal, samples 4-7 exhibited PAC.

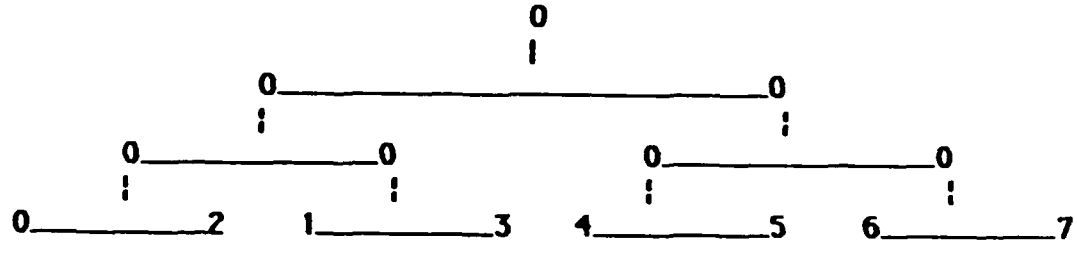


Figure 5.10
Hierachical clustering of normal and PAC ECG's

The dendrogram for a three class problem is given in figure 5.11. In this example, samples 0-3 were normal, samples 4-7 exhibited PAC, and 5-9 were tachicardic.

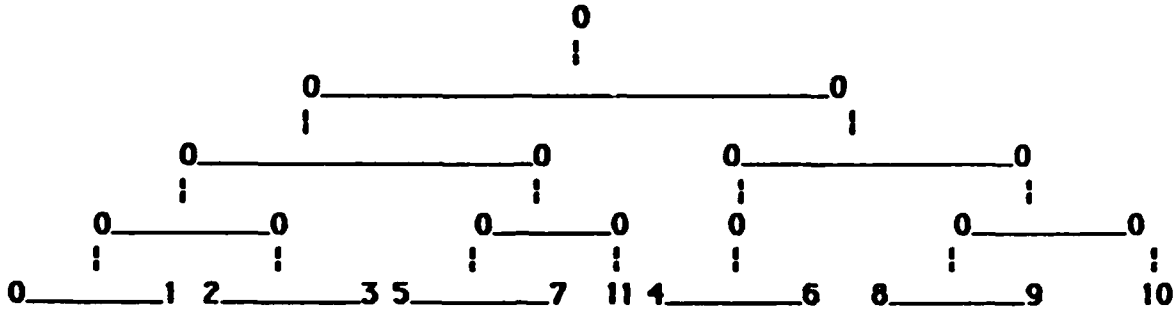


Figure 5.11

Three-class ECG hierarchical clustering dendrogram

5.10 Discussion of Results

The relational trees shown in the appendix demonstrate the filtering capabilities of the tree transform. The fluctuations in basic wave shape are removed by propagating the desired subtrees and eliminating those subtrees judged as superfluous. By storing time and amplitude information as node attributes, waveforms may be reconstructed from the transformed trees. The results of this experiment can be compared to the existing numerical techniques. Since within each signal class there are infinitely many variations in the waveform, an infinite number of matched filters would be required to accurately represent the problem. Assuming the signal set could be limited to a finite number of possible forms, a bank of matched filters would perform better than the technique presented here at low SNR's, but not as well for high SNR's. When the amplitude of the noise becomes large enough to distort the peak dominance relations in the tree representation, the method breaks down.

A further consideration is the complexity of the system. Banks of matched filters are difficult to implement, and require many fixed or floating point operations. The relational tree clustering technique, once the transform operations have been selected, requires only addressing operations and integer comparisons.

If a prototype tree can be constructed, system complexity is reduced linearly, keeping the number of tree comparisons to a minimum. In addition, the tree transformation speeds up the tree distance calculations by the square of the number of nodes removed since the distance computation runs in $O(n^2)$ time, where n is the number of nodes.

Comparing tables 4.2 and 4.4, it becomes apparent that the performance in terms of classification error is much worse for the ECG case than for the seismic case. The explanation for this lies in the increased complexity of the ECG waveforms over the seismic waveforms. The need to distinguish complex trees from one another, and to recognize and eliminate noise subtrees, calls for a carefully constructed set of tree transform rewriting rules. The effectiveness of the transformation is limited by the candidate rules supplied by the human user. A methodical rule-selection algorithm is worth investigating. Information for the automatic design of a transform rule set could be gathered as a byproduct of the distance calculation. Since the distance between two trees results from the best matching between individual nodes, a large amount of detail concerning the correlation between subtrees is simply being ignored. The precise method by which this additional information is put to use needs to be the subject of further research. Linking the transform design to the tree matching process would not only simplify the

rule selection, but would reduce the complexity of the entire system markedly.

In addition to improving the rule selection process, the rules themselves might be enhanced by taking semantic information into account. More information may be encoded in the rules annotatively. If the single attribute "valley height" was to be included at each non-terminal node in the relational tree, more intelligent filtering could take place. The decision to propogate a subtree as relavent information, or to classify it as noise and prune it, depends to some extent on the depth of the parent node's valley. It was this inability to distinguish absolute valley depths that limited the success of the ECG rule set. The inclusion of at least some semantic information is evidently crucial for the recognition of complex waveforms. The theory for handling semantic information in the tree transform needs to be thoroughly developed before any improvements can be made to the implementation.

Chapter 6

Conclusions

We have endeavored to construct a system which will identify and classify waveforms based on their underlying structural similarities. An elegant and convenient means to represent a signal's structure, without regard to absolute magnitudes or precise timing, is the relational tree. The relational tree is a computer data structure that represents a waveform by the relative placement of peaks and valleys.

We can treat the relational tree much as a vector in pattern space. Several pseudo-metrics have been developed for measuring distance between trees. Using a tree metric and many of the concepts from traditional cluster analysis, we have designed a waveform recognition system. A key element of the waveform recognition system is the tree transform. The reason for transforming the tree space is to improve the clustering configuration so that unknown candidate trees lie close to their prototypes in a sample training set. The actions of a tree transform are determined by the rewriting rules which are mappings between subtrees. By varying the rewriting rule set, a transform may be found which improves tree clustering performance. We have introduced two objective functions for measuring that performance. They vary in action and complexity.

After implementing the waveform recognition system and testing it on simulated reflection seismic and electrocardiogram data, the following observations may be made.

(1) The symbolic recognition system in its present form is only feasible if the tree complexity is low, i.e. the signal contains a small number of peaks and valleys.

(2) For these waveforms, the classification error is equal to or better than numerical techniques at low signal to noise ratios, but abruptly becomes worse as relative peak and valley heights are altered by noise.

(3) A system of practical complexity requires automated rule selection. Information needed to select these rules is available as a byproduct of the tree metric.

(4) The transform rule set could be greatly enhanced by adding semantic information, but the theory governing such a transform has yet to be developed.

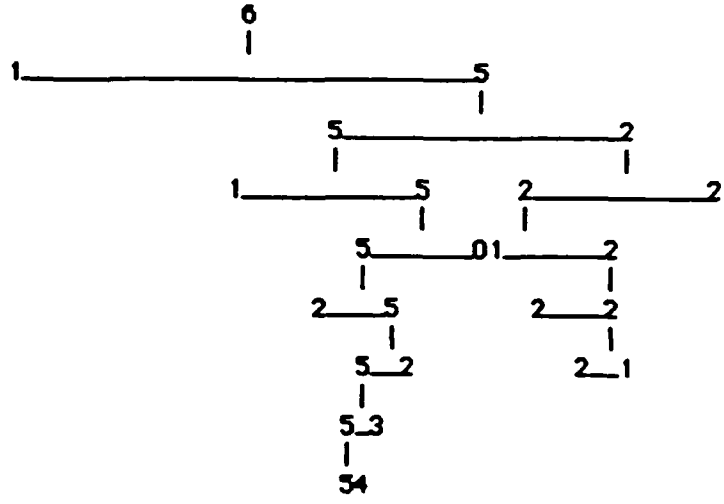
The decision to use relational tree waveform recognition must be made on the merits of each individual set of waveforms. Waveforms whose relational tree structures are similar will be difficult to distinguish in tree space. The strength of the RT method becomes apparent if the relational tree structures of opposing clusters lie far apart in tree space, as in the seismic example.

Appendix

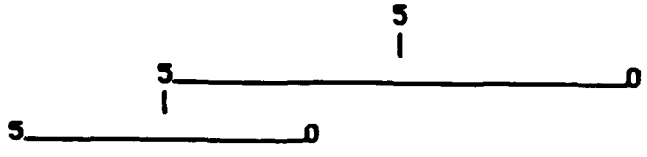
The following relational trees are taken from the seismic example with 5db signal-to-noise ratio. The trees are shown before and after undergoing transformation with the rewriting rules in table 4.1.

No Sand Present

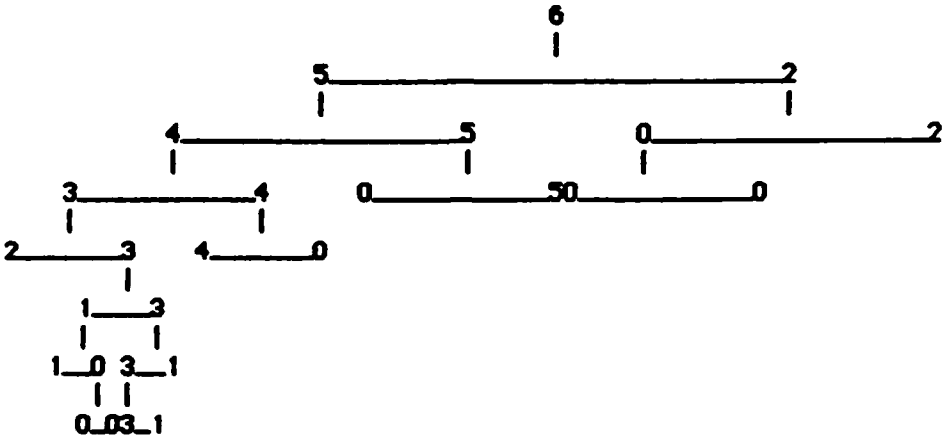
Before transform:



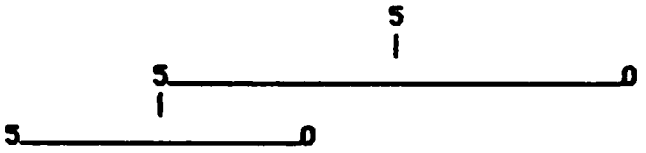
After transform:



Before transform:

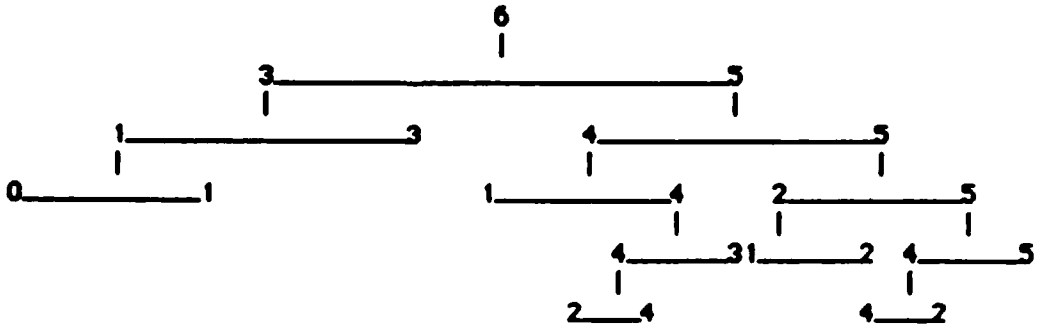


After transform:

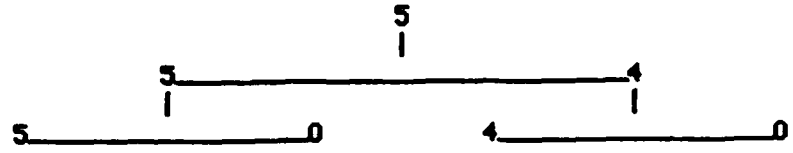


Sand Present

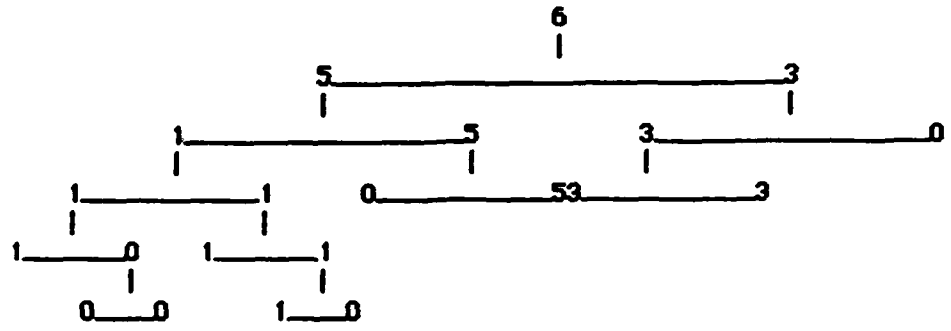
Before transform:



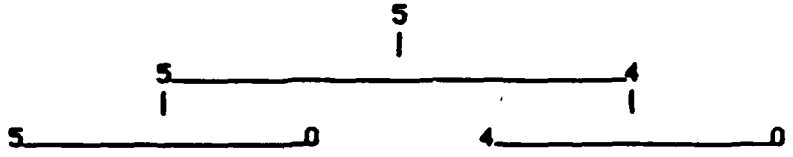
After transform:



Before transform:



After transform:



References:

[1] R. W. Erich , and J. P. Foith ,"Representation Of Random Waveforms by Relational Trees. " IEEE. Trans. Computers. Vol. c-25, pp. 725-736, (July 1976).

[2] Yao-chou Cheng, and Shin-yee Lu, "Waveform Correlation Using Tree Matching." Proc. IEEE Conf. PRIP. pp. 350-354, (1982).

[3] S. Y. Lu, and K. S. Fu, "Structure-Preserved Error-Correcting Tree Automata for Syntactic Pattern Recognition." Proc. IEEE Conf. Decis. Control. (Dec. 1-3, 1976).

[4] S.Y. Lu, and K. S. Fu, "Error-Correcting Tree Automata for Syntactic Pattern Recognition." Pattern Recognition. Vol. 2, No. 3, pp. 213-222, (1979).

[5] S. Y. Lu, "A Tree-To-Tree Distance and Its Application to Cluster Analysis." IEEE. Trans. PAMI ,Vol. 1, No. 2, pp. 219-224, (April 1979).

[6] S. Y. Lu, "A Tree-Matching Algoritm Based on Node Splitting and Merging." IEEE. Trans. PAMI. Vol. 6, No. 2, pp. 249-256, (March 1984).

[7] S. Y. Lu, and K. S. Fu,"A Cluster Procedure for Syntactic Patterns." IEEE. Trans. SMC. Vol. SMC-7, pp. 734-742, (May 1978).

[8] R. O. Duda, and P. E. Hart, Pattern Classification and Scene Analysis. John Wiley & Sons, New York, Ch. 4.

[9] J. T. Tou, and R. C. Gonzales, Pattern Recognition Principles, Addison-Wesley, Reading, MA. Ch. 3.

[10] K. S. Fu, Syntactic Pattern Recognition and Applications. John Wiley & Sons, New York, (1983).

[11] F. Gecseg, and M. Steinby, Tree Automata. Akademiai Kiado, Budapest, (1984).

[12] Keith Clark, and Steve Gregory, "Parlog: Parallel Prograsmming in Logic" Imperial College of Science and Technology Research Report DOC 84/4, April, (1984).

[13] W.F. Clocksin, C. S. Melish, Programming In Prolog. Springer-Verlag, New York, N.Y, (1981).

[14] Norman S. Neidell, Stratigraphic Modeling and Interpretation: Geophysical Principles and Techniques, (AAPG Course Note Series # 13), Amer. Assoc. Pet. Geol., Tulsa, OK,p. 86, (1984).

[15] W. F. Ganong, Review of Medical Physiology, Lange, Los Altos, CA. pp. 393-408, (1973).

END

7-87

DTIC