NCC2-438
110991
25P.

INTELLIGENT SIGNAL ANALYSIS AND RECOGNITION : A report for the period 12/01/1986 through 11/30/1987

by

Robert Levinson - Computer and Information Sciences - Principal Investigator
Daniel Helman - Computer Engineering - Co-Investigator
Edward Oswalt - Computer and Information Sciences - Graduate Research Assistant

University of California at Santa Cruz
November 30, 1987

ABSTRACT

In this report, we describe our progress in the research and development of a self-organizing database system that can support the identification and characterization of signals in an RF environment. This report is for the first-year of a three-year plan to develop such a system. As the radio frequency spectrum becomes more crowded, there are a growing number of situations that require a characterization of the RF environment. This database system is designed to be practical in applications where communications and other instrumentation encounter a time varying and complex RF environment.

The primary application of this system is the guidance and control of NASA's SETI Microwave Observing Project. Other possible applications include selection of telemetry bands for communication with spacecraft, and the scheduling of antenna for radio astronomy are two examples where characterization of the RF environment is required. In these applications, the RF environment is constantly changing, and even experienced operators cannot quickly identify the multitude of signals that can be encountered. Some of these signals are repetitive, others appear to occur sporadically.

(NASA-CR-181531) INTELLIGENT SIGNAL ANALYSIS AND RECOGNITION Report, 1 Dec. 1986 - 30 Nov. 1987 (California Univ.) 25 p CSCL 05B

N88-13084

Unclas
G3/82 0110991

INTRODUCTION

The description of any pattern recognition task requires a means for developing patterns that characterize the input, and the means for recognizing or organizing these patterns. In the RF environment, the input to the pattern recognition and data characterization process is formed from the output of the receivers used for these signals. Thus, the pattern formation must be tailored to the specific application, although general principles for pattern development will apply across applications. The principles for classification, recognition or pattern organization present an even greater challenge. In many applications there has not been gathered sufficient data, nor has there been great enough human experience, to guide the development of classification or recognition. For this For this reason, "classical" expert system tools (e.g. Emycin) are not particularly promising for this task. The goal of this research is to develop an adaptive or self-organizing system to characterize these signals and to help track the (local) RF environment as it changes. Thus the desired system will gain "experience" over time and improve its ability to recognize particular signals.

This report has four major sections:

SECTION 1: THE THREE-YEAR PLAN

The three-year plan of proposed research, and the progress made in the first year.

SECTION 2: DESIGN OBJECTIVES

The design principles and objectives upon which the design of a prototype system has been based.

SECTION 3: PROTOTYPE DESIGN

The design of a prototype database system with two components:

a. Input Clustering

b. Self-Organizing Pattern Retrieval System for Signals

This is the main topic of this report.

SECTION 4: SOFTWARE AND GRAPHICS

Software and graphics programs have been combined to produce a demonstration program.

SECTION 1: THE THREE-YEAR PLAN

The following excerpt is from the proposal for this project:

> In the first year of the project, the investigators will produce a report documenting the results of research on the potential for application of intelligent systems to the automated characterization of RF environments. Detailed specifications of the requirements for this RF characterization, as well as an outline of the computational approach to be used by these modules will be provided. A primary objective of this first year effort would be the development of an appropriate knowledge representation scheme for signal data.
>
> Second year efforts will focus on the implementation and testing of prototypes for signal characterization. This stage is critical as it will provide data to determine if the program modules can efficiently and effectively process the data that is presented to them, and organize it in a way that is meaningful to applications requiring characterization of the RF environment.

In the third year it is expected that the primary focus of this work will be on issues associated with the design and implementation of the program modules in a form that meets the hardware, software and processing time constraints of a selected application. Hardware implementations of program modules will also be considered where appropriate.

With the completion of this report we have accomplished the proposal objectives for the first year. In addition to this written report we will be providing NASA with the software and graphics programs described below. Some of these programs were newly written at UCSC; others are debugged and modified versions of simulation programs provided by SETI. These programs form a miniature model of the database design and provide a useful way of demonstrating, clarifying and evaluating the design concepts and their interrelationship. Though care has been taken to select the most computationally efficient algorithms for these programs, the actual implementation does not have efficiency as the main objective.

In the second year, we will use the demonstration programs as a basis for developing a more complex prototype where efficiency and performance considerations will be given top priority. Also in the second year, the prototype will be thoroughly tested on "realistic" data sets.

The third year should proceed as planned.

SECTION 2: DESIGN OBJECTIVES

After reviewing the issues involved in constructing an RFI database system and the requirements of the SETI application, we determined that a practical self-organizing database system for RFI must meet the following criteria:

1. The system must be able to quickly (in real-time) determine whether or not it has seen a signal previously and return classification information about that signal. The SETI application requires a response time between 1 and 10 seconds.
2. The system must be able to adapt itself to different RF environments, antenna locations and survey strategies.
3. Under time pressure, the system must still be capable of returning reasonable (though not exact) responses. During busy periods the system may not have time to analyze each signal completely, but the system must return enough probabilistic information about the signal's classification so that a decision making system or human operator can make an intelligent choice with regard to the tracking of this signal.
4. The system should help to characterize the data that it has stored. This characterization should be understandable by humans and have a similar structure (and possibly a different content) to human classification schemes. It is quite possible that the system will recognize significant classes and patterns in signal data that a human would not. This requirement rules out the neural net model of associative pattern retrieval [Rumelhart 86]. The internal structure of the net is usually not in a form comprehensible to human users. Also note that a neural net requires accurate feedback from the start in order to be trained. This training data is not readily available or agreed on in

the SETI application, but may only come to be so after a large amount of data has been collected.

5. The system should be able to incorporate human feedback and heuristics into its scheme. Human expertise in RFI detection and signal analysis should be used where appropriate. Outside of training data, there is little mechanism for including human knowledge and heuristics into a neural net.
6. By using its data characterization capabilities the system should maintain good response time despite storing increasing amounts of data. In fact, the accuracy of its classifications should improve with larger amounts of data, since it has more information on which to base its decisions.

SECTION 3: PROTOTYPE DESIGN

In this section we present the design of a self-organizing database system for RFI that we feel meets the design objectives above . The design is for a system with two major components. The first component is an input clusterer or preprocessor. The second component is the database itself. [1]

Section 3.1: Component I - Input preprocessor

The goal of the input preprocessor is to recognize individual signals in the current input. These signals are then sent to the RFI database to be compared with previously encountered signals and classified. Two types of signals, pulsed and continuous wave (CW), can be found. Signals are characterized by their usual attributes: frequency, power, duration, etc.

Section 3.1.1: Input

In the SETI system output from the radio telescope's antenna is sent to a Multi-Channel Spectrum Analyzer (MCSA)[Cullers 85] [2] The MCSA can simultaneously track ten million 1 Hz wide frequency channels. In other words, once per second it outputs the average signal strength found in each 1 Hz band over a 10 MHz bandwidth. It can also simultaneously compute the spectral energies at coarser resolutions, e.g. 5

---

[1] An alternative, yet similar, scheme for self-organizing retrieval [Omohundro 87] involving the data structure known as K-D trees was brought to our attention only a few weeks before the end of this reporting period. With some work, K-D trees can be adapted for nearest neighbor classification and dynamic database reorganization. It is not clear that K-D trees provide the flexibility of the scheme presented here or that they produce realistic intermediate data classes. In the second year of the project period we will explore the possibility of adapting the K-D tree scheme for RFI recognition. Even if a move to K-D trees took place, most of the same data structures, design concepts and mechanisms would remain the same. The only difference may be somewhat faster retrieval. At any rate, we are encouraged to know that another researcher has been thinking along similar lines.

[2] Actually, two systems, for both left and right circular polarization, are planned. For the purposes of initially detecting signals their outputs would be combined.

million 2 Hz bands twice a second, 2.5 million 4 Hz bands four times a second, etc. In SETI, several resolutions from 1 KHz to 1 Hz will be used.

Because of the tremendous amount of data involved (greater than one value every 100 nanoseconds) the data will be thresholded so that only significant "hits" need to be processed. Broadband RFI will be recognized in the low resolution data, so that the corresponding range of channels can be ignored at higher resolutions.

Section 3.1.2: Output

We wish to detect continuous and pulsed signals found in observation periods that are approximately 15 minutes long. Signals will be characterized by their frequency, bandwidth, power, slope (frequency drift due to relative motion of sender and receiver), and duration. In addition, pulsed signals also have a pulse frequency attribute.

Other information such as the date, time and location of the observation is, of course, relevant to characterizing a signal, but this information is only used by the database. The database takes the physical data found by the input preprocessor together with other relevant information and tries to match this data with the results of previous observations.

Section 3.1.3: Signal Detection

The detection algorithms take the data from the MCSA, which are points in a three dimensional (frequency, time, power) space. For the highest resolution observations frequency is quantized into Hz, time to seconds, and power is always greater than a threshold. One can imagine a grid with 10 million frequency channels along the horizontal axis, 900 seconds (observations) along the vertical axis, and each grid square containing a hit colored with a shade proportional to the observed power.

The task is to find line segments (signals) on the graph created by the darker blocks. It is assumed that the signals in question will not drift faster than one Hz per second. This means that the magnitude of the slope of the lines will be greater than or equal to one. Pulse trains are like dotted lines on the graph. Continuous signals can be thought of as pulses that occur on every one second sample. Bandwidth is the thickness of the lines.

Since their slope is limited, signal lines are restricted to very small portions of the graph (one ten thousandth). To find signals we use algorithms supplied by NASA-Ames. All possible pulse trains (of an appropriate slope) are tracked from each hit. When a train is finished (either expected pulses didn't occur, or at the end of an observation period) it is reported if its total power meets a threshold.

Each actual signal (together with noise) can cause many trains to be reported. For example, a signal with a two second pulse period can be reported a two second pulse period signal, two different four second pulse period signals, three different six second pulse period signals, etc. (if it is powerful enough). Noise together with a strong signal can cause many different trains with slightly varying slopes. Crossing signals also can also cause problems.

These many false trains can be weeded down by sorting signals according to their duration and pulse frequency. Accept long duration, high frequency nonoverlapping

signals. Long trains that overlap in slope and frequency can be discarded if their pulse frequency is a multiple of the already recognized signals. Short trains whose center frequency lies on the path of an existing signal are also suspect. Intersecting trains can be accepted only if their slopes are sufficiently different.

The signal detection algorithms are by no means perfected. Continuous signals with non-integral slopes (a staircase appearance) can be detected as overlapping pulse trains. Pulsed signals that do not have an integral pulse period will not necessarily be detected, and if they are it may be as a very sparse train. Signals that cross channel boundaries may not meet the threshold requirement for either channel. Wide signals may swamp the detection algorithm by causing it to track many different trains.

However, it is expected that most signals will be terrestrial and/or noise. Since terrestrial signals will not drift, and are usually wide band continuous waves, they should be easy to detect. If thresholds are set properly, false trains due to noise can also be mitigated.

Section 3.1.4: Output Format

The database wants its input signals to be in the form of a hyper-rectangle in the attribute space, i.e. each attribute is a range of values rather than a single point. The purpose is to be able to use a similar representation for individual, groups and hierarchies of signals. In addition, using a range for each dimension allows us to recognize that two observations are the same signal even if the equivalence is inexact due to the vagaries of the train clustering algorithm.

In the previous section we have discussed weeding out the pulse trains derived from an observation to find the representative signal(s). What we need now is a method for converting the data into hyper-rectangles and a method for combining hyper-rectangles themselves so that two observations of one source will produce equivalent signals.

We propose the following solution. The range of each dimension will be based on the minimum and maximum values (of that attribute) of all the pulse trains that were considered to be part of the same signal, subject to an overall minimum and maximum for each attribute set by the operator. These minima and maxima can be set based on the sensitivity of the equipment and heuristic knowledge of the current experiment.

By choosing ranges in this way we ensure that any individual pulse train from a future observation will be considered equivalent to a stored signal if, had it occurred when the signal was first formed, it would have been part of it. Of course, the validity of this and the other grouping algorithm that follows will need to be borne out by experimental data.

Section 3.1.5: Uncertainties

Each of the dimensional attributes have their own elements of uncertainty. Pulse frequency measurements may not be accurate if the source frequency is not an integral number of seconds per pulse. Power measurements may vary somewhat from site to site due to antenna differences. Duration should be fairly reliable, in any case the signals we

are interested in finding would have to continually observable. Short duration signals must be assumed to be RFI. Bandwidth should be consistent.

Drift and frequency measurements are the most tentative. They involve assumptions about the relative motions of the source and receiver of the signals. If, during the fifteen minute observation period, the signal source has an appreciable acceleration relative to our earth based antenna, the observed signal will drift. In addition, if scans are made at different times of the year, the perceived frequency from an extraterrestrial source could be quite different due to the changing relative velocities.

We will make certain simplifying assumptions:

1. Most of the signals to be stored in the database are RFI from earth based or geosynchronous sources.

2. Besides database retrieval (the fastest and easiest method), there are other ways of determining the signal type (RFI, ET) during the observational period. These include matching with information from the omnidirectional antenna and changing the direction of the main antenna to check if the observed signal is coming from the direction of the star being watched.

3. We have access to signal data both in raw form, and modified to correct for the motion of our antenna relative to an observer at the star we are scanning. We will call these two the signal and the corrected signal.

Extraterrestrial signals may or may not be corrected at their source for their own motion relative to the earth. They certainly cannot be corrected for our motion. If the received signal is appreciably different from the corrected signal, and the source is as yet undetermined, then both must be stored in the database. If the relative positions of the observed star and ourselves are such that no signal correction is necessary, we will have to rely more heavily on alternative methods of classification (second assumption).

For the rest of the paper we will only consider signals with stationary attributes (earth based or corrected stationary ET) as candidates for classification by database retrieval. The very few (first assumption) that can not be handled this way will have to be dealt with by an operator.

Section 3.1.6: Combining Hyper-rectangles

It will be clear from the following sections that the main purpose of using hyper-rectangles is to speed database retrieval over the brute force method of computing the distance metric between a candidate signal and every stored signal. Therefore, it is preferable to err by making the hyper-rectangle a bit too large (inefficient retrieval) rather than too small (misclassification).

When signal samples are found to be equivalent and are to be stored as a single hyper-rectangle in the database, we get the dimensions of the result as follows. The center point of the new hyper-rectangle is easily computed to be the average of the centers of all signals that have been combined to form the new one. An attribute's range is taken to be + or - three standard deviations of that attribute from the mean. This way over 99 percent of those observations (if they are normally distributed) would be included in the resulting hyper-rectangle. As before, these ranges should be subject to a minimum and maximum set by the operator.

Section 3.2: Component II - Database system.

Component II of the design has four subcomponents, described in the following four subsections. These four subcomponents are:

1. A representation scheme for signals and classes of signals.
2. A global data structure for representing the interrelationship of signal classes.
3. A retrieval/insertion algorithm and distance metric.
   --- Real-time processing of a recently observed signal.
4. Organization/reorganization (clustering) algorithms.
   --- Off-line restructuring to improve retrieval efficiency and classification accuracy.

Note that the database is to operate in two modes:

1. *Real-time query mode.* Given a new signal instance, rapidly determine which, if any, previous signal it corresponds to and add it to the database.
2. *Off-line restructuring.* Where appropriate, recluster (reclassify) parts of the database that have had new insertions, in order to improve the performance of mode 1.

Section 3.2.1: Representation scheme.

It is important to have a uniform scheme, so far as is possible, for representing individual patterns(signals) and pattern (signal) classes. This uniformity facilitates fast retrieval as it allows for a reasonably simple retrieval mechanism and a database with a high degree of mutual information (structure). Due to factors such as noise, overlapping signals, moving sources, antenna angles and the lack of established knowledge and experience in this area, the RFI recognition task is currently an inexact one. Two identical signals produced by the same source will produce attribute values that differ in not always predictable ways. We have designedour representation scheme for signals and signal classes to be a uniform one that takes the inexactness of the recognition task into account. We have also selected a distance metric (see Section 3.2.3) that is appropriate for this representation scheme.

We now present our representation scheme for both individual signals and signal classes. The main representation is a hyper-rectangle in an attribute vector space. The hyper-rectangle is defined by closed segments for each attribute. The attributes chosen for both individual signals and signal classes are:

a. frequency
b. bandwidth
c. pulse frequency
d. slope
e. power
f. duration
g. full-duration-flag: The closed segment will be either [1,1] or [0,0] depending on whether or not the signal lasted the full duration of the scanning session.

Thus, the hyper-rectangles will enclose a region in n-space (7-space for exactly the attributes above). Note that because of the inexactness of the signal matching task, even individual signals will be represented by regions rather than single points.

We have chosen the hyper-rectangle representation rather than a hyper-sphere representatioin scheme, for two reasons. First, as will be seen in the next section, determining whether a point belongs to a hyper-rectangle can usually be done much more efficiently than determining whether it belongs to a hyper-sphere (represented as a center and radius). Second, we feel it is much easier for a human user to understand what class of signals are represented by a hyper-rectangle than a hyper- sphere in which the bounds for each attribute vary with the values for other attributes.

In addition to the hyper-rectangle, the representation scheme for an individual signal is also to include the following fields:

i. center point: The center of the region enclosed by the hyper-rectangle.
j. classification: For example, RFI,TV, satellite, etc.
k. weight:The number of instances of this signal that have been seen.
l. other info: (one for each instance) date, time and location.
m. superclasses: (See description in the next paragraph.)
n. special-action: (See description in the next paragraph.)
o. update-flag: TRUE if and only if an instance has been added to this class since the last reclustering phase. This flag indicates those parts of the database that may need to be reclustered.
p. comments: These could include specific information about the signal source if known (e.g. Continental Flight 784 Minneapolis-Denver).

The representation scheme for a class of signals is to contain the following fields in addition to the hyper-rectangle:

i. standard deviations: (one for each of the seven attribute dimensions) These will be used by the distance metric.
j. total weight: The sum of the weights (occurences) for each of the individual signals contained in this rectangle.

k. # of occurences for each classification:
e.g., RFI - 2 signals 6 occurrences
TV - 1 signal 2 occurrences
AIR -1 signal 1 occurrence
This information will be used by the retrieval algorithm (Section 3.2.3) to produce probabilistic classifications if necessary.

l. expected retrieval time: Given that we know a new signal (query) falls in this signal class (hyper-rectangle), how much more time is required to determine which if any individual signal hyper-rectangles include it? This information is supplied and used by the retrieval evaluation algorithm (Section 3.2.4).

m. optimal retrieval time: The minimum retrieval time given a perfect clustering distribution.

n. subclasses: Pointers to signal classes (hyper-rectangles) immediately contained in this signal class. That is, classes z such that z is contained in this class and is contained in no other class that is contained in this class.

o. superclasses: Pointers to signal classes that immediately contain this signal class. That is, classes z such that z contains this class and z contains no other class that contains this class.

p. special-action: An action that the retrieval/insertion algorithm is to take, if ever a signal is found to be in this class. This mechanism will allow the retrieval mechanism to interrupt normal processing to handle special cases.

q. modify-flag: TRUE iff this signal class may be modified.

r. comments: The subclasses and superclass fields are used to efficiently store the global data structure described in the next section.

Thus, the data structures for individual signals and classes for signals both use the hyper-rectangle to define their class. Additional fields are used in both, since individual signals and signal classes are created and used for different purposes. Individual signal structures are created for every unique signal the system believes has been reported. Signal classes are created by the system to improve retrieval efficiency and to provide informative probabilistic responses when necessary. Signal classes may also be defined and created by a user who would like the system to recognize a particular type of signal.

Section 3.2.2: The global data structure

It is important to structure the database in such a way that that the interrelationships (overlaps and containments) between signal classes can fully be exploited. We have chosen to store the database as a partial ordering based on the relation "superclass-of". By partial ordering we mean that the following properties are true of the relation "superclass-of": (superclass-of(A,B) denotes that A is a superclass of B, where A and B are any two classes.

1. superclass-of(A,A) - reflexivity.
2. A <> B and superclass-of(A,B) => not superclass-of(B,A) - antisymmetry.
3. superclass-of(A,B) and superclass-of(B,C) => superclass-of(A,C) - transitivity.

For the type of database presented here, we will actually use irreflexivity instead of reflexivity. That is, superclass(A,A) will always be false. Hence A will not be included in its own subclass and superclass lists. As described in the section above we will store pointers from each signal class to its immediate superclasses and immediate subclasses. One useful way to view the database is as a tangled hierarchy. "Tangled" since a node in the hierarchy can have more than one parent. We can imagine the tangled hierarchy as starting with the universal superclass at the top, and leaf nodes at the bottom representing individual signals. (See Figure 1.) The universal superclass is the signal class that contains all other signal classes and signals contained in the database.

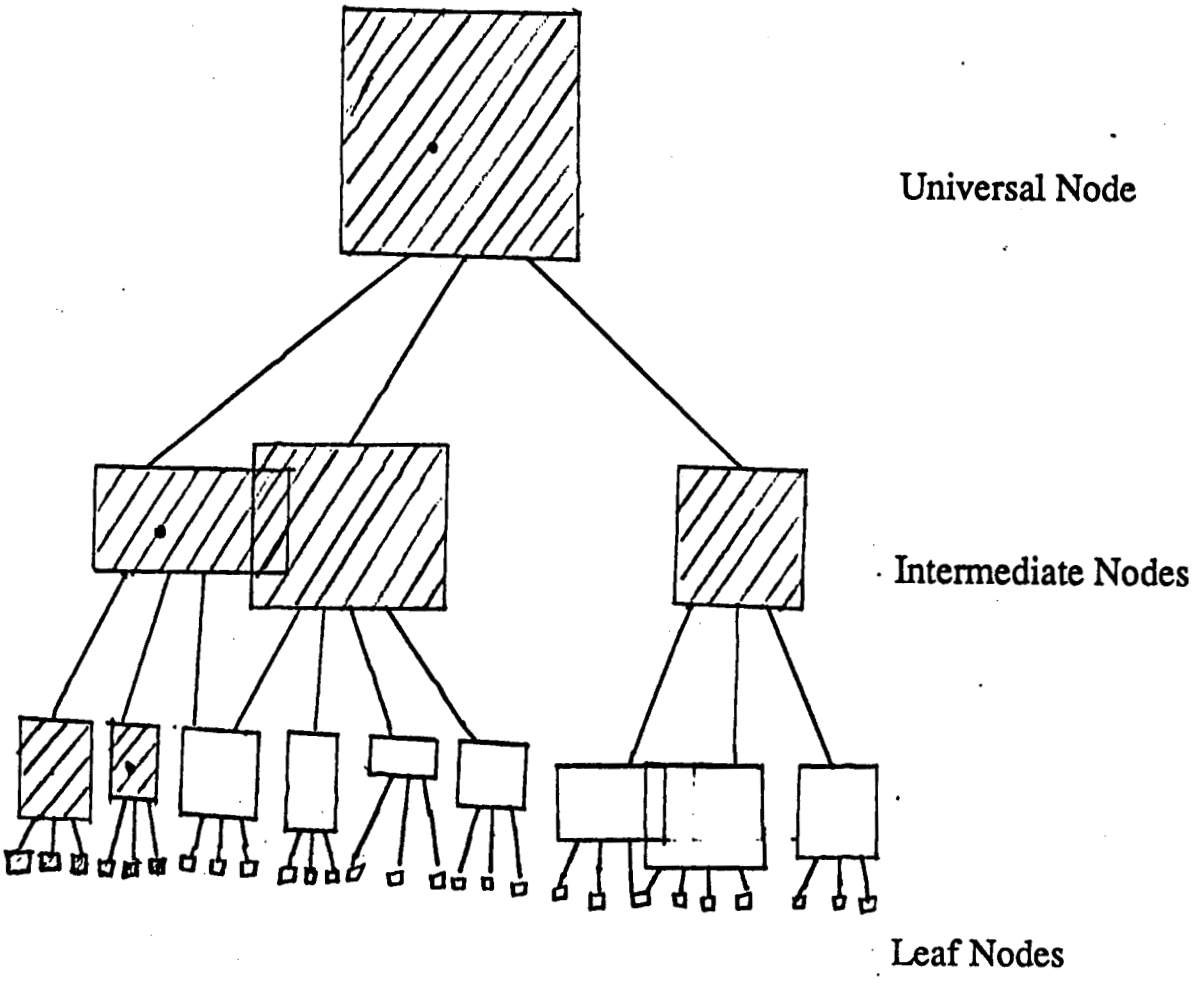

Figure 1

A partial ordering, with the path of a retrieval shaded.

 : Nodes which are checked for point containment.

 : Nodes containing the point.

Section 3.2.3 Retrieval/insertion algorithm and distance metric.

In this section we define a distance metric for comparing two signal vectors for similarity, and a simple yet efficient algorithm for retrieving and inserting signals given the partially-ordered database.

Section 3.2.3.1 The distance metric.

Let signals have n attributes.

Let sd(i) = the standard deviation of the ith attribute in the lowest level signal class involving all the points.

Define the distance between two signals s1 and s2 (represented as points) as $d(s1,s2) = |s1(1) - s2(1)| / sd(1) + |s1(2) - s2(2)| / sd(2) + \ldots + |s1(n) - s2(n)| / sd(n)$. If sd(i)=0 take the ith summand to be 0, since the two signals match on this attribute.

This definition of d satisfies the following three defining properties of a metric and thus induces a metric space: For any three signals a,b and c,

1. d(a,b)= 0 iff a=b.
2. d(a,b)= d(b,a)
3. d(a,b) <= d(a,c) +d(c,b), since it is true for absolute value.

This distance metric can be computed efficiently since the reciprocals of the (non-zero) standard deviations can be calculated and stored ahead of time.

The purpose of including the standard deviations in the distance metric is to weight the attribute distances appropriately. These standard deviations will vary from class to class, so by choosing the lowest level signal class containing the points involved the attributes, the same attribute will be weighted differently in different parts of the database. If we determine empirically or otherwise that one attribute is more critical to similarity than other attributes, we can include this information by, in addition, assigning weight parameters along each attribute dimension.

The effect of the distance metric on the overall performance of the database should not be underestimated. The structure of the metric presented here should be coupled with empirical tests and human input to produce a metric that consistently conforms to human judgement. For example, we think it would be a good idea to weight the contribution of the slope attribute (see Section 3.2.1) by the minimum of the durations of each signal.

It may be that a simple linear formula as above is not sufficient, but that a short algorithm involving various special cases is necessary. Such a metric can only be developed after extensive experience with realistic data and user feedback. Development of an intelligent metric is a major objective of the second year of the project.

Section 3.2.3.2: Retrieval/insertion algorithm.

Given a new signal instance Q we need to rapidly determine the previously stored signal it corresponds to, if any, and insert it where appropriate. The algorithm works in two phases.

In Phase I, we use a breadth-first top-down elimination algorithm. This corresponds to Phase I of the retrieval algorithm first presented in [Levinson 84]. The purpose of Phase I, is to quickly determine all individual signal hyper-rectangles a signal Q

(considered as a single point in the attribute space) falls into.

How does Phase I work? It attempts to find all the leaf nodes containing Q while using as few hyper-rectangle comparisons as possible. Informally, the idea is to never check whether a hyper-rectangle X contains Q, unless all hyper-rectangles containing X contain Q. This leads to the following breadth-first, top-down elimination algorithm:

List all hyper-rectangles in the database from largest to smallest in volume. (Note that this insures that no hyper-rectangle will be preceded in the list by any hyper-rectangle that it contains.)

With each member of the list assign a number "superclass count" equal to the number of its immediate superclasses.

```
While there is an unexplored member X of the list with superclass count = 0 do
  If Q fits in X
    Mark X as a hit.
    If X has a special-action specified, perform the action.
    Subtract 1 from the superclass count of each immediate subclass of X.
  Else
    Mark X as a miss.

  Return all leafnodes that have been marked as hits.
```

Note that a superclass count of a class Y goes to zero only if all of Y's immediate superclasses have been marked as hits.

Phase I can be terminated early in two special cases. First, a special-action may preempt the retrieval process for some other special activity, or it may simply cause a termination if, for example, the signal is clearly earth-based in origin.

Secondly, due to time pressure (many signals to process) Phase I may be stopped before completion. If some leafnodes had been marked as hits, we would send those on to Phase II. However, what if we didn't get as far as the leaf nodes? In this case, we can only produce a probabilistic classification. This is done by finding the set of signal classes that have been marked as hits and contain no other hit class. These will be the most specific signal classes that have been hit. These classes provide input to a procedure that determines the probability that the signal belongs to a particular classification (RFI,TV,etc.) The sum of these probabilities subtracted from 1, give the probability that it is a new signal that does not meet previous classifications.

For example, suppose the class X with the following occurrence data is the only smallest signal class with a hit:

RFI - 3 signals 7 occurrences
TV - 2 signals 3 occurrences
AIR - 1 signal 1 occurrence

Then 7-3 of the 11 occurrences have been previously seen RFI, 3-2 have been previously seen TV, and 1-1 have been previously seen AIR, thus (11-(4+1+0)) or 6 out of 11 have been new signals. If more than one class (besides X) is involved, we first find the

cumulative signal and occurrences data for each classification and then proceed as above.

If the retrieval process has been interrupted, we do not have enough information available to know where to insert the new signal. In this case, the signal is logged and is inserted during the off-line restructuring mode.

If Phase I completes normally, Phase II begins. The purpose of Phase II, is to return the classification of Q and insert it in the proper place in the database as follows:

```
If Phase I returns no individual signals
   Return "Signal not found"
   Insert Q as a new individual signal (leaf node) in the database
   The size of the hyper-rectangle associated with Q, will be based on output from the input clusterer. (S
   Set update-flag(Q) to TRUE.
Else
   Find the individual signal X with closest center to Q.
   Return X and its classification.
   Update leaf X, to include the new instance Q.
   Set update-flag(X) to TRUE.
```

Note that if there are only a few intersecting classes in the database, maintaining a superclass count in Phase I may not be necessary, since most classes will have only one immediate superclass. In this case, whenever X is found as a hit, we can immediately check if any of its immediate subclasses are hits.

This retrieval system seems especially suited for real-time signal recognition since the code required for the retrieval algorithm is short and easy to implement, and thus may lend itself easily to an efficient implementation in hardware. Since the algorithm resembles the standard data flow evaluation algorithm [Dennis 74], a parallel implementation may also be possible.

Section 3.2.4: Organization/reorganization algorithms and evaluation strategies.

In order to enhance retrieval efficiency, intermediate signal classes are created by the system in the initial off-line restructuring phase. This is done by taking a standard statistical clustering algorithm and adapting it for the purpose of improving retrieval efficiency. During later off-line restructuring phases, based on an analysis of the system's expected retrieval performance given the new database insertions, the clustering algorithm is applied to "recluster" local parts of the database.

We explored three clustering algorithms and their hierarchical variants. Due to studies of the output from these programs and recent theoretical results [Pitt 87], we recommend the use of a special adaptation of the hierarchical agglomerative algorithm. In this section we present:

1. The three clustering algorithms.
2. The use of clustering to construct a hierarchy.
3. How to analyze retrieval efficiency.

4. How the hierarchical agglomerative algorithm can be used to produce a hierarchy which enhances retrieval efficiency.
5. How restructuring (reorganization) of the database is done.

Section 3.2.4.1: The three clustering algorithms.

The purpose in general of a clustering algorithm is to group together similar data points. In this case "similar" means that the vectors representing data points are close to each other according to the distance metric (Section 3.2.3.1). [Anderberg 73] is a general survey of clustering algorithms. The three clustering algorithms we have explored are the Forgy method, nearest neighbor transitive closure, and agglomerative algorithm. These algorithms were originally designed to produce hyper-spheres, but they can be modified to produce hyper-rectangles.

Section 3.2.4.1.1: Forgy algorithm.

In the Forgy method, the user specifies the number of clusters into which the points are to be grouped. Good candidate cluster centers are found from among all the points which have high densities of points nearby, but are not near other candidate centers. Starting with these initial centers, the algorithm iterates according to this rule: each point is assigned to the nearest center, and centers of clusters are re-computed as the average of the points in the cluster. The algorithm terminates when no points change centers in an iteration. There is a variant due to Jancy which doubles the distance cluster centers move each iteration. Since cluster centers often tend to drift along a roughly straight line, this variant is thought to reduce the required number of iterations. Our studies using flat distributions fail to show this.

Section 3.2.4.1.2: Transitive closure algorithm.

The transitive closure algorithm computes a binary matrix which is 1 in the (i, j) position if points i and j are nearest neighbors, otherwise 0. The transitive closure of this matrix is then computed and points in the same equivalence class are clustered together. One feature of this algorithm is that the number of clusters is decided by the algorithm itself. An obvious disadvantage is that a dense cluster of points which naturally may be considered a single cluster is likely to be divided into several by the presence of very tight clusters within it. With flat distributions, the algorithm found 17 clusters in 60 points, 31 in 100 points, 124 in 400.

Section 3.2.4.1.3: Agglomerate algorithm.

We believe the most appropriate algorithm for this application is the agglomerative algorithm. Starting with all points as singleton clusters, successively combine the pair of clusters which are closest to each other into a single cluster until the desired number of clusters is reached. [Pitt 87] shows that this simple algorithm produces optimal (according to certain reasonable criteria) clusterings efficiently (in polynomial time). In particular, it desirable to have both a high density of points in each cluster (the cluster

boundaries are tight around a group of close points), and a large minimum distance between clusters (where the distance between clusters is the distance between their closest points). The agglomerative algorithm will maximize the minimum of these two quantities: the average density of clusters and the minimum distance between clusters.

Section 3.2.4.2: The construction of a hierarchy.

Any of the three algorithms described above may be used to form a hierarchy of clusters. This is often desirable in taxonomic classification schemes. The hierarchy is constructed by subdividing clusters to form a lower level of clusters (we call this top-down), or by combining existing clusters into larger ones (bottom-up). Top-down does not work in the nearest neighbor algorithm (clusters can not be subdivided), but both types of hierarchy are possible with the other algorithms, giving a total of five combinations.

Once the hierarchy is constructed in this way, the system can go through the hierarchy bottom-up "pruning" (i.e., leaving unclustered) any class whose expected -retrieval-time is less than if it was left unclustered.

Section 3.2.4.3: Evaluating retrieval efficiency.

In this section we present methods for analyzing the relationship between a hierarchical clustering of data points and the expected performance of the retrieval/insertion algorithm described in Section 3.2.3.2. Certainly, in order to analyze expected retrieval performance some assumptions must be made about the distribution of future queries/insertions to the database. The following alternatives seem reasonable, though being only assumptions they can be only approximations of the truth:

1. Future queries to the database correspond exactly to signals already in the database.
2. Future queries correspond to the lowest-level non-leaf classes in the database and their weights.
3. Future queries correspond to the volumes of the lowest-level non-leaf classes in the database.
4. Recency assumption. Number leaf-nodes in the database from least recent (1) to most recent (N) and apply these as weights in any of the above schemes.

For this analysis (and in the demo system) we make assumption 1. This is analogous to the assumption usually made when designing optimal data structures [Overmars 83]. Similar, though somewhat more complex analysis can be done using another assumption.

For any query we can measure retrieval time by the number of class inclusion tests in Phase I and the number of metric calculations in Phase II. Since the costs of these operations are likely to be nearly the same, we will define retrieval time as the sum of these.

How do we measure expected-retrieval-time for a class X of signals or for the database as a whole (the universal signal class)? First let's consider the case in which none of the subclasses in the database overlap. That is, the database is a tree. Suppose the class X has m subclasses and total weight N (that is, N signal instances) then ordering the subclasses

X.1 through X.m by decreasing weights and applying assumption 1 we get:
Expected-Retrieval-Time(X) (= ERT(X)) =
[|X.1|(1 + ERT(X.1)) + (|X.2|(2 + ERT(X.2)) + ... + |X.N|(N + ERT(X.N))] / N
since once a point is found to be in a class the inclusion tests may stop, since the classes do not overlap. We would expect this formula to be optimized when the subclass sizes are equal and the corresponding subtrees are equivalent, this gives:
ERT(X) = (m/2 (inclusion tests) + ERT(X.1).

Note that if we do no clustering at all ERT(X) = N. Analysis also shows that optimal clustering will produce branching factors of m=2 and ERT(X) = $\log_2(N) + 1$. on 4 or less points.

For the case where subclasses overlap, (which we expect to keep to a minimum) we have found that simulating the algorithm on each of the data points is the best way to estimate efficiency. Since two points or classes are equivalent with respect to retrieval if they are contained in the same superclasses, and we already know in which classes the points are included the simulation can be computed efficiently.

Section 3.2.4.4: Using the hierarchical agglomerative algorithm to improve retrieval efficiency.

To optimize retrieval efficiency we want to minimize the expected number of inclusion and distance tests (as discussed above). But we also have a secondary objective of maximizing the average density (minimizing the volume) of the classes in the database. Density maximization increases the probability that early on in the retrieval process that we will recognize that the incoming signal corresponds to no previous signal. We would expect that as the database grows the average density will decrease. We would like it to decrease as little as possible. Also, density provides a good measure of how well the clustering algorithms have characterized the data points. Having higher density classes means that any probabilistic responses the system gives will be more accurate since they will be based on more specific classes. Note that if density maximization was our primary objective, we would simply put each data point in a cluster by itself. Note on the other hand that if retrieval efficiency was our only objective, we would have little concern for the volume of our classes as long as they contained a certain number of points.

How can we use the agglomerative algorithm to achieve the two objectives of minimizing ERT(X) for the database X (the universal superclass) of N points while minimizing the total volume of X's immediate superclasses? Our approach will be to adapt the algorithm to produce one level of m subclusters, and then to apply the algorithm recursively on each of the subclusters. Note that a good measure for ERT(X) where X has only one level of m clusters is m + entropy(X).

The entropy is the standard information theoretic definition [Shannon 48]:
$H(x) = P_1 Log(1/P_1) + P_2 Log(1/P_2) + ... + P_n Log(1/P_n)$
where the P_i are the probabilities of each of the N events. The probabilities here are interpreted as the probability of a point being in each subcluster, so that clusters whose sons contain nearly equal numbers of points have high entropy, which is desirable from the retrieval efficiency standpoint.

As has been shown [Pitt 87] the agglomerative algorithm is based on a goodness measure that is the minimum of two measures - tightness and distance. Tightness reflects the distance of points within a subcluster. For this we will use the total volume of the subclusters - in order to maximize their density. Distance reflects the distance between the subclusters, for this we will use [m-1 + entropy(X)] * minimum intercluster distance.(This will have to be normalized to be comparable to our tightness measure.) This distance measure corresponds to our retrieval
objective. These functions satisfy the properties necessary for the agglomerative algorithm to produce a clustering of optimal goodness. Note that since our goodness function is a mixture of both retrieval time and density, we can not claim that the resulting clustering is optimal with respect to just one or the other.

Section 3.2.4.5: Reorganization of the database.

After insertions are made to the database in real-time query mode (see Section 3.2.3), the database organization (taxonomy) may no longer be in accord with the performance objective of fast retrieval. Furthermore, class divisions in the database may no longer characterize the data points properly. Thus, we would like the system to be able to change its database structure to reflect the new information. Though reclustering the entire database is possible, it is a costly operation. Instead, the system locally reclusters only those classes of the database that no longer meet a predefined performance criterion based on expected retrieval time and optimal retrieval time. This type of database reorganization is analogous to one psychological explanation of dreaming. It is the process of maintaining order in our model of the world by rehashing and integrating the day's experiences and events.

How does the database system perform local reclustering? Recall from Section 3.2.4.3 that the expected-retrieval-time for a class can be defined recursively in terms of the expected-retrieval-time for each of its subclasses. Hence, as is done when the system is pruning a proposed clustering, we work our way bottom-up , starting with the leaf-nodes, determining the expected retrieval efficiency of a class and reclustering that class if it does not meet our performance criterion. Note that we need only recalculate retrieval efficiency for those classes with an update-flag = TRUE (since at least one leaf node has been added) , and also that we can not remove classes with a modify-flag = FALSE. This process ensures that we improve the retrieval performance of smaller classes in the database before considering the larger classes (or the entire database).

Another type of database reorganization technique that is possible, though not yet studied in depth, is to remove signals from the database that have not been repeated in a long while and only retain summary information about these signals. This would correspond to the recency assumption discussed in Section 3.2.4.3.

The statistical fields such as standard deviations and weights are all calculated for new classes and recalculated for modified classes during the reorganization phase.

SECTION 4: DEMONSTRATION AND TEST PROGRAM

The purpose of the demonstration program is to simulate the performance of the proposed data base scheme. In the absence of a large collection of radio signal data that could give insight into the actual distribution of the parameters that characterize the signals, this program generates data points according to a user specified distribution. Another possible approach is to interface this program to the output of the NASA signal generation and detection simulators. [3]
However, this merely passes the responsibility of the choice of input parameters to the signal generating program. We instead create random vectors directly to represent radio signals. The distribution may be the sum of a finite number of Gaussian distributions, or be perfectly flat. The sum of Gaussians distributions is specified by using a file that contains the center point and the covariance matrix for each distribution. The program is compiled for two-element data points (that is, for only two parameters for each signal), but a simple change and re-compile allows any number of attributes. The randomly constructed data points are converted into the partial ordering by "superclass-of" using a clustering algorithm. The user has a choice of the three clustering algorithms discussed in Section 3.2.4.1. For the Forgy and agglomerative algorithms, the user must specify the desired branching factor from one level to the next. In the two parameter case, graphics displays show the form of the hierarchy.

Given a hierarchical clustering, points are retrieved according to the algorithm described in section 3.2.3.2.

This program does not perform all of the functions that the final system will. This program gathers all data vectors, creates the partial ordering, and then gathers statistics. The final system will have the ability to add new points to the data base at any time, and to update statistics and re-organize as time permits. The re-organization scheme in the final system will include the ability to re-organize subgraphs of the entire data-base; this will require statistics to be gathered on the required retrieval times of points, given that their location in the intermediate levels is known. These conditional retrieval times reflect the efficiency of the hierarchical sub-structure rooted at the intermediate node in question. The final system will also require some criteria to decide, based on retrieval statistics, when re-clustering is worthwhile.

Section 4.1.1: Statistics.

Statistics are gathered to measure the comparative performance of clustering algorithms on data sets of various sizes and distributions. Statistics include the entropy of each branching, cost of retrieval, and densities of clusters.

---

[3] There are 2 programs by NASA that can be cascaded to supply vector data points to this program. The first of these accepts specifications for RFI signals of various types and parameters (e.g. frequency, modulation, strength, etc.), and produces a signal with noise added. The second simulates pulse detection hardware and software to retrieve the parameters from the signal embedded in the noise. The second requires some system between it and our demo program to collect multiple reports of the same signal into a single report. This intermediate system corresponds to the input preprocessor discussed in Section 3.1.

The cost of retrieval is simply a count of the rectangle inclusion and vector equality tests needed in the retrieval algorithm. The total cost figures given by the program assume each point in the data base is retrieved once, and each time the a point is retrieved it is the last one checked in its smallest enclosing rectangle (the worst case). This number is compared to n^2 which is the worst-case retrieval time for n points with no clustering, and to $n \log_2(n)$ which is the expected (optimum) total retrieval cost with perfect clustering. The size of each rectangle, the number of points each contains, and the density of each is also given.

Section 4.1.2: Graphics displays.

The "-g" option gives the user views of the hierarchy, and in the case of the Forgy algorithm, a view of the operation of the clustering algorithm. The option should be followed by the digit 1, 2, or 3 -- e.g. -g2. The larger the digit, the more graphs are shown. The option -g1 shows only the border of the rectangles and numbers in their corners showing the identification number of the cluster (used internally in the program) and (in parenthesis) the number of the level in the hierarchy the rectangle is in (Level 1 is immediatedly below the root (universal set), level 2 is their sons, etc. The option -g2 gives a more detailed account of the clusters by drawing lines between each pair of points in each cluster. The option -g3 shows individual points as they are created, and the trace of the centers in the Forgy algorithm. After each level's clustering is performed, the program stops so that the graphs may be seen. Hitting any key resumes operation.

Section 4.1.3: Running the program.

The graphics portion of the program uses the X package, which is relatively standard. There are many user options. Each involves the use of command line arguments. Following the word "demo", or "Xdemo" (with the X graphics routines), any of the following may be typed:
-gn : do graphics, n is a digit, the larger the more detailed graphing (try -g1)
-dn : dump information, n is a digit, the larger the more info
-a: use the agglomerative clustering algorithm
-z: use the nearest neighbor transitive closure algorithm
-f: use Forgy algorithm
-j: use Jancy variant of Forgy algorithm
-bn: use bottom up hierarchy with branching factor of n
-tn: use top down, separating each cluster into n sons
-pn: use n total data points

-p is mandatory, you should have exactly one of -t or -b.

The source code consists of the files called: demo.h, demo.c, display.c, nearneigh.c, agglom.c, and combine.c. Demo.h contains global declarations; demo.c the high-level routines and command line reading; display.c the routines related to display including graphics and gathering and printing of statistics; nearneigh.c, forgy.c, and agglom.c

implement the three clustering algorithms.

Section 4.1.4: Sample retrieval results.

The table below gives costs of retrieval for data bases of various sizes, constructed using each of the 3 clustering algorithms.

In each of these sample runs, a bottom-up hierarchy was constructed with a constant branching factor of 4. Each point in the data base was retrieved once. The number of inclusion tests is the total number of times in Phase I a test for a point inside a rectangle was required. The number of equality tests is the number of times in Phase II a point was checked for being identical to a point already in the data base. Also, these numbers are divided by the total number of points to give the average cost per point retrieved. Note that the agglomerative algorithm used here does not implement the techniques discussed in Section 3.2.4.4.

Number of points	Incl tests	Incl per pt	Eq tests	Eq per pt
Agglomerative Algorithm				
32	331	10.34	154	4.81
64	733	11.45	244	3.81
128	2198	17.1	376	2.93
256	5794	22.63	1245	4.86
Forgy Algorithm				
32	271	8.46	153	4.78
64	721	11.26	220	3.43
128	1776	13.87	577	4.50
256	4108	16.04	1081	4.24
512	10555	20.61	2488	4.86
Transitive Closure Algorithm				
32	241	7.53	313	9.78
64	614	9.59	417	6.51
128	1802	14.07	836	6.53
256	5958	23.27	1611	6.29
512	18331	35.80	3536	6.90

BIBLIOGRAPHY

[Anderberg 73] *Anderberg, Michael R.*
Cluster Analysis for Applications.
New York
Academic Press, 1973.

[Cullers 85] *Cullers, D.K., Linscott,I.R., Oliver,B.M.*
Signal Processing in SETI.
Communications of the ACM
Volume 28, Number 11, November 1985.

[Dennis 74] *Dennis,J.B.*
First Version of a Data Flow Procedure Language.
Lecture Notes in Computer Science
19: 362-376, 1974.

[Levinson 84] *Levinson, R.*
A Self-Organizing Retrieval System for Graphs.
In Proc. AAAI-84. 1984.

[Levinson 85] *Levinson, R.*
A Self-Organizing Retrieval System for Graphs,
PhD Dissertation, Univ. of Texas at Austin, May 1985.

[Michalski 83] *Michalski,R.S. and Stepp,R.E.*
Learning from observation: Conceptual Clustering.
In Michalski,R.S.,Carbonell,J.G.,and Mitchell,T.M.,ed.
Machine Learning: An Artificial Intelligence Approach.
Tioga Press, 1983.

[Mitchell 82] *Mitchell,T.M.*
Generalization as Search.
Artificial Intelligence
18:203-226, 1982.

[Omohundro 87] *Omohundro, Stephen M.*
Efficient Algorithms With Neural Network Behavior
U Illinois Tech Report UIUCDCS-R-87-1331.
April, 1987.

[Osherson 86] *Osherson, D.N., Stob, M., and Weinstein, S.,*
Systems That Learn: An Introduction to Learning Theory
for Cognitive and Computer Scientists,
MIT Press, 1986.

[Overmars 83] *Overmars, M. H.*
The Design of Dynamic Data Structures
Lecture Notes in Computer Science
Number 156
Springer-Verlag, 1983.

[Pitt 87] *Pitt, Leonard and Reinke, Robert E.*
Polynomial-Time Solvability of Clustering and Conceptual
Clustering Problems: The Agglomerative-Hierarchical Algorithm
U. Illinois Tech Report UIUCDCS R-87-1371
September 1, 1987

[Rumelhart 86] *Rumelhart, E.D., and McClelland, J.L..*

Parallel Distributed Processing
PDP Research Group
UC-San Diego
Vols. 1-2
MIT Press, 1986.

[Shannon 48] *C.E. Shannon.*
A Mathematical Theory of Information
Bell Systems Technical Journal 27, pp 379-423, 623-656.

[Van Melle 81] *Van Melle, W., Scott, A.C., Bennett, J.S., Peairs, M.A.*
The EMYCIN Manual
Heuristic Programming Project
Stanford University, 1981.