General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

Produced by the NASA Center for Aerospace Information (CASI)

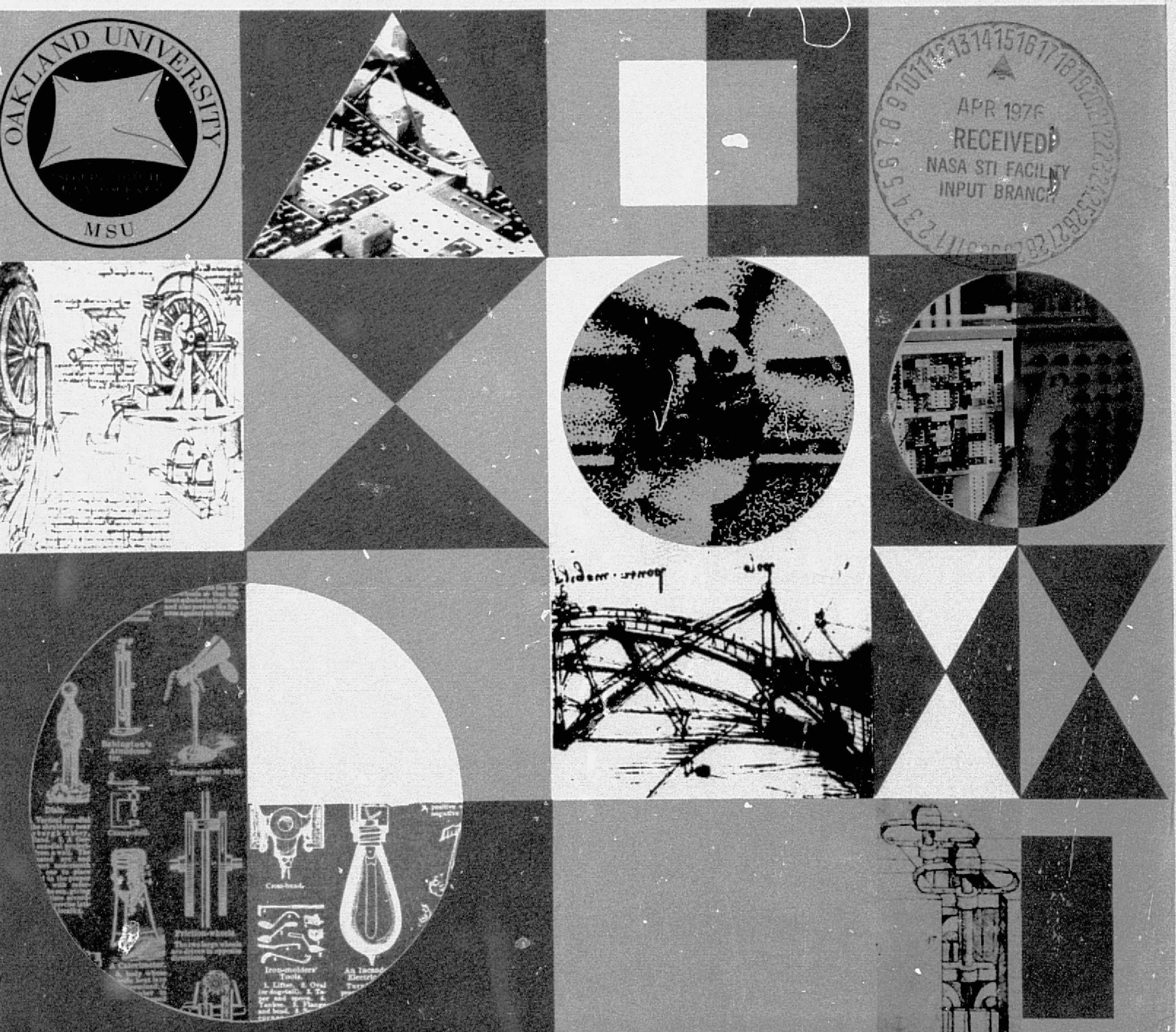
NASA CR-
14 7524

(NASA-CR-147524) INVESTIGATION OF CORRELATION CLASSIFICATION TECHNIQUES Final Report (Oakland Univ.) 114 p HC $5.50 CSCL 09B

N76-20862

Unclas
G3/61 21498

OAKLAND UNIVERSITY SCHOOL OF ENGINEERING



APR 1976
RECEIVED
NASA STI FACILITY
INPUT BRANCH

FINAL REPORT

NASA Contract NAS-9-14195

Investigation of Correlation Classification Techniques

Principal Investigator: Dr. Richard E. Haskell
School of Engineering
Oakland University
Rochester, Michigan 48063

For: Earth Observations Division
NASA Johnson Space Center
Houston, TX 77058

December 1975

Abstract

A two-step classification algorithm for processing multispectral scanner data has been developed and tested. The algorithm is carried out by two separate programs called CLUSTX and GROUPX. The program CLUSTX is a single pass clustering algorithm that assigns each pixel, based on its spectral signature, to a particular cluster. The output of the program CLUSTX is a cluster tape in which a single integer is associated with each pixel. This integer is the cluster number to which the pixel has been assigned by the program. The cluster tape is used as the input to the classification program GROUPX. Ground truth information is used in GROUPX to classify each cluster using an iterative method of potentials. Once the clusters have been assigned to classes the cluster tape is read pixel-by-pixel and an output tape is produced in which each pixel is assigned to its proper class. The classification algorithm can be operated in a hierarchical manner in which each ground truth datum is classified at various levels in a classification tree. In addition to the digital classification programs, a method of using correlation clustering to process multispectral scanner data in real time by means of an interactive color video display is also described.

ORIGINAL PAGE IS
OF POOR QUALITY

Table of Contents

ORIGINAL PAGE IS
OF POOR QUALITY

1. Summary and Overview

The research undertaken under this contract had as its goal the development and evaluation of various correlation techniques which might be useful in the processing of multispectral scanner data. This study is an outgrowth of work that was initially undertaken when the principal investigator was on sabbatical leave at the Johnson Space Center during the 1972-73 academic year.

At that time the principal investigator developed a single-pass clustering algorithm called CLUSTD[1] that could be used as a nonsupervised classifier. In addition, the possibilities of using coherent optical methods in the processing of multispectral scanner data were also studied.[2] Considerable progress has been made under the present contract in clarifying the potential role of these techniques and significant advances in developing and evaluating these methods have been achieved.

The major accomplishments of the current research effort include the following:

1) The overall digital processing of multispectral scanner data has been separated into two separate tasks. The first is to associate every pixel with a particular cluster by using a single-pass correlation clustering algorithm. The clusters are made small enough so that (nearly) all pixels in a given cluster will have very similar spectral signatures and therefore can be associated with the same class. The second task is to classify each cluster using ground truth information and thus, by association, to classify each pixel in the flight line. This separation of the processing tasks means that only a relatively

few spectral signatures need to be classified by the classifier (usually less than 200, corresponding to the spectral signatures associated with each cluster). As a result very powerful, non-linear, nonparametric classifiers can be used to classify these clusters. A more detailed description of this overall processing method is given in Section 3.

2) Two new single-pass correlation clustering algorithms have been developed. These algorithms have replaced the original method used in CLUSTD that was based on a transformation of the spectral signature into a binary signature in which the elements were either +1 or -1. The improved algorithms accomplish the same task without the need for this transformation. (This transformation was originally invented for an optical implementation in which it is required.) These single-pass clustering algorithms are grouped under the general name of CLUSTX and are described in more detail in Section 4.

3) The single-pass clustering algorithms CLUSTX have been extensively studied. The goal is to generate enough clusters so that all of the pixels in a given cluster will belong to a single class. This can obviously be achieved in the limit of one cluster per pixel. We have found, however, that with fewer than 200 clusters, over 99% of all pixels in a given cluster will, on the average, belong to one class. The best results are achieved when the physical separation of pixels associated with the same cluster is not allowed to become too great.

ORIGINAL PAGE IS
OF POOR QUALITY

4) A new, nonparametric method of classifying the clusters based on an iterative method of potentials has been developed. This algorithm is described in more detail in Section 5. In one version of the program, the training data for the classifier is taken to be the clusters that have been assigned to classes based on the costmatrix (i.e., by simply counting ground truth pixels in each cluster). This works well when large quantities of ground truth are available. For example, we have achieved an overall classification accuracy of approximately 94% when applying this method to the 12-channel aircraft scanner data of the C-1 flight line. The program GROUPX has been further improved by modifying it in such a way that the classifier based on the method of potentials can be trained directly from the ground truth pixels. This means that many fewer ground truth pixels are needed in order to effectively train the classifier.

5) A new hierarchical classifier called CHIMP (for Classification Hierarchy using an Iterative Method of Potentials) has been developed. This classifier allows ground truth information to be stored in the form of a classification tree with various levels of detail. For example, the class corn could be stored simultaneously as land, agricultural land, cultivated agricultural land, and corn. Classification of all pixels can occur at any level. In addition, ground truth information can be entered at any level and used for classifying all higher levels. For example, a pixel may be known to be forest but the particular type of trees may be unknown. This pixel could be used as ground truth

for classifying, for example at a forest, urban, agriculture level. This type of flexibility could greatly reduce the cost of acquiring ground truth by making maximum use of such things as aerial photographs. That is, all information that is known, at whatever level of detail can be handled by the classifier. A description of this new classifier is described in Section 6.

6) New methods of processing large quantities of multispectral scanner data have been studied. This was the original motivation for investigating the possibilities of applying various optical techniques.[2] In order to make a major advance in the use of multispectral scanner data the goal should be to develop a real-time processing system in which the human operator can interactively control the regions of feature space that are being observed. Such a real-time processor will require considerable parallel processing capabilities and optical methods seem to offer a possible choice. However, after an extensive study of the current optical processing technology it was concluded that the development of a real-time, interactive, color processing system is beyond the present state-of-the-art. Alternate technologies were then investigated and the preliminary design of a real-time processing system using a hybrid digital/analog system has been completed. This system, which could have a major impact on the usefulness and applications of multispectral scanner data, is described in Section 7.

2. <u>Recommendations</u>

As a result of the work done under this contract the following specific recommendations are made:

A. Software

1. It is recommended that the program CLUSTX be made operational at NASA-JSC after the following improvements and modifications have been made.

   a) A preprocessing procedure should be included that will sample the data in order to determine the optimum window size (the threshold parameter) such that clusters are generated at an appropriate rate.

   b) A feature should be added that will start generating new clusters on a new file when the maximum number of clusters is reached or when a certain number of scan lines has been processed. This will minimize the problem of different classes that are widely separated on the ground but might have similar spectral signatures. In addition, it will allow an entire data tape to be processed at one time.

   c) The COSTMATRIX procedure should be provided as an option in CLUSTX for evaluating the effectiveness of the clustering operation when ground truth information is available. This information should be stored on the cluster tape.

   d) The linear correlation measure and the rectangular correlation measure should be provided as alternate correlation measure options.

ORIGINAL PAGE IS
OF POOR QUALITY

2. It is recommended that the program GROUPX be implemented at NASA-JSC after the following improvements and modifications have been made.

   a) An algorithm should be implemented that will automatically use all ground truth information within the particular area corresponding to the data on the cluster tape. In addition ground truth from an area on either side of the region being processed should be used. With this modification multi-file cluster tapes can be processed with new ground truth information always being added from in front of the flight path while old ground truth corresponding to areas behind the flight path are being discarded.

   b) Modifications should be made that will allow the program to be compatible with multi-file cluster tapes. These modifications would produce multi-file output tapes.

   c) An option should be provided for classifying the clusters using either an Iterative Potential Function Method or a Gaussian Maximum Likelihood Method.

   d) The hierarchical classifier CHIMP that can classify at various levels of detail should be incorporated as an option in the program.

   e) An option that will produce a line printer classification map should be included.

   f) An option that will produce a PMIS-DAS tape output should be provided.

   g) The capability of inputing ground truth test data and producing an error matrix for testing the classification accuracy should be provided.

B. Hardware

It is recommended that a prototype interactive color display system as described in Section 7 of this report be built and tested. The major parts of the system would include

1. A high density magnetic disk assembly with 32 fixed head transducers,
2. A tape drive and processor suitable for loading the fixed head refresh disk,
3. A specially designed interactive analog processor incorporating high speed D/A converters,
4. A color TV monitor.

3. Processing Multispectral Scanner Data Using Correlation Clustering and Nonparametric Classification Techniques

The classification algorithm developed under this contract is a two-step process carried out by two separate programs called CLUSTX and GROUPX. The functions of these two programs are illustrated in the block diagram of Fig. 1. The input data tape contains multispectral scanner data in the form of 8-bit integers representing, for each pixel, the reflectance measured in each of several spectral channels. Thus, associated with each pixel on the input data tape are NCHAN integers (ranging in value from 0 to 255) where NCHAN is the number of spectral channels.

The program CLUSTX is a single pass clustering algorithm that assigns each pixel, based on its spectral signature, to one of NCLUST clusters. The maximum value of NCLUST is MAXCLUST (typically MAXCLUST=200). However, the actual value of NCLUST is variable and is determined by two variable parameters in the program. The output of the program CLUSTX is a cluster tape in which a single integer is associated with each pixel. This integer is the cluster number to which the pixel has been assigned by the program.

The clustering program CLUSTX can be considered to be a data reduction and preprocessing step in the classification algorithm. Thus, for example, whereas the original problem might be to classify each of say 40,000 pixels as one of 4 classes, CLUSTX reduces the problem to one of classifying each of a maximum of MAXCLUST clusters. The assumption is that enough clusters are chosen so that all pixels assigned to a particular cluster have very similar spectral signatures and thus belong to the same class. A spectral signature is associated with each cluster. This signature is the average signature of all pixels that have been assigned to the cluster. A detailed description of the program CLUSTX is given in Section 4.

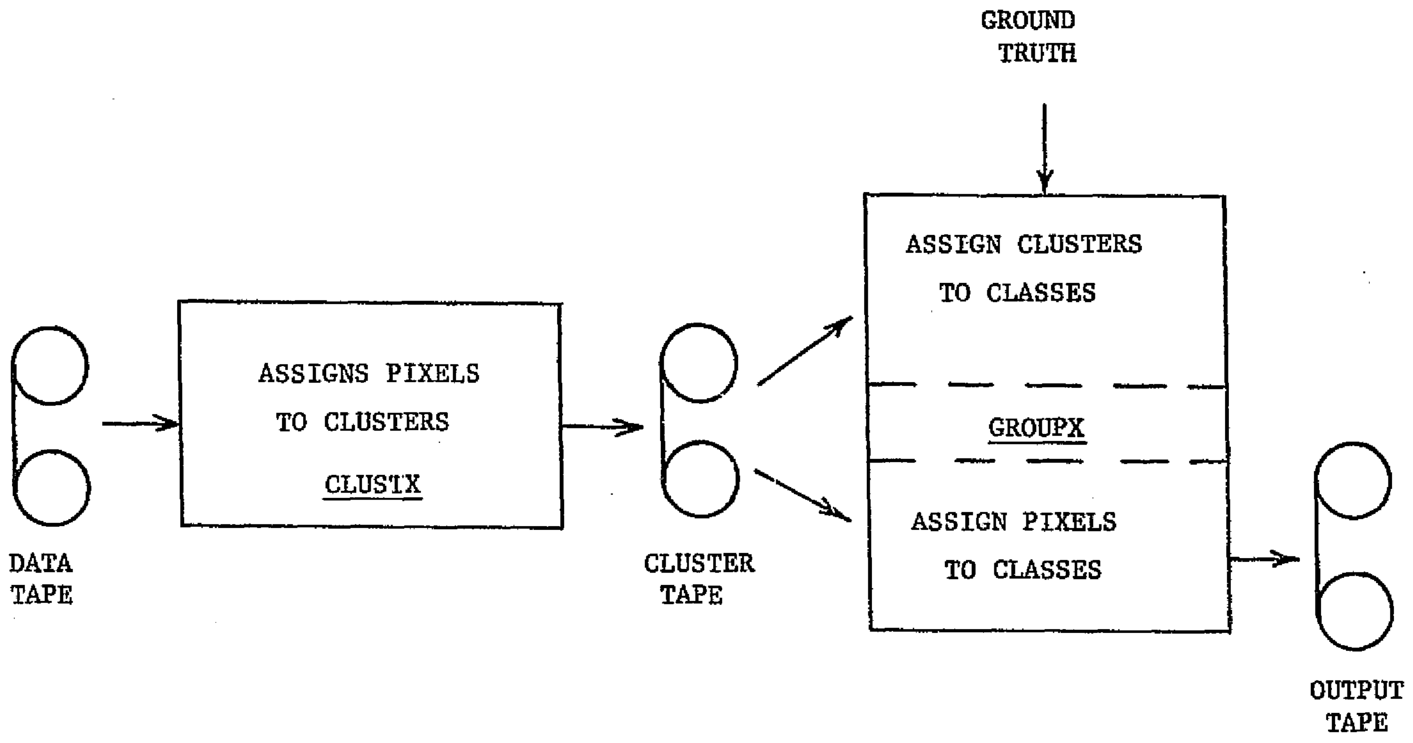


Fig. 1 Flow Diagram for Processing Multispectral Scanner Data Using Correlation Clustering and Nonparametric Classification Techniques.

The cluster tape which is the output of the program CLUSTX is the input to the classification program GROUPX. Ground truth information is used in GROUPX to classify each cluster as one of a small set of classes. Since the maximum number of possible clusters is 200 the number of items to be classified is relatively small. However, once the clusters have been assigned to classes the cluster tape is read pixel by pixel and an output tape is produced in which each pixel is assigned to its proper class. This output classification tape can then be used directly to produce classification maps, compute acreage of different classes, or test the accuracy of the classification method by comparing the results with additional ground truth.

Ground truth information is used to train the classifier that will classify each pixel. This classifier creates nonlinear decision surfaces based on the method of potentials. Two types of training are possible. If the ground truth is limited then the spectral signatures from each pixel are used to construct the decision surfaces. On the other hand, if a large quantity of ground truth is available, then it can be used to produce a costmatrix giving the number of pixels in each cluster that belongs to each of the various classes. These numbers are used to estimate the a posteriori probabilities of a particular cluster belonging to a particular class. The cluster is then assigned to the class for which this a posteriori probability is a maximum. The clusters classified in this manner serve as the training data for constructing the decision surfaces using the potential functions. The remaining clusters are then classified using the method of potentials. A more complete description of the method of potentials is given in Section 5.

ORIGINAL PAGE IS
OF POOR QUALITY

The advantages of this classification algorithm include the following:

1) The classification method is entirely nonparametric and thus avoids the errors that are inherent in estimating parameter vectors in parametric methods. This should lead to a more effective utilization of all of the information when data from a large number of channels is used. In particular, multimodal distributions of particular classes cause no problem.

2) Changes in the spectral signature of a particular class along the flight line cause no problem as long as representative ground truth is available, since the result will simply be the generation of new clusters. These clusters will then be assigned to the proper class in GROUPX.

3) If new ground truth information is obtained only GROUPX needs to be run to produce a new output tape.

4) The clustering can be done before the ground truth is obtained and the results of the clustering can be used as an aid in selecting ground truth areas.

4. Data Reduction Using A Single Pass Correlation Clustering Algorithm

The program CLUSTX is a single pass clustering algorithm that uses a correlation function as a similarity measure for assigning each pixel to a particular cluster. This correlation function is a measure of similarity between the spectral signature of a new pixel and the spectral signatures associated with previously generated clusters. Let $\underset{\sim}{x}$ be the N-channel spectral signature associated with a particular pixel. That is, $\underset{\sim}{x}^T = [x_1, x_2, \ldots, x_n]$. Let $\underset{\sim}{y}^{(i)}$ be the average of the spectral signatures of all pixels that have previously been assigned to cluster number i. Let $\phi_j(x_j - y_j^{(i)})$ be a weighting function associated with channel j whose value is a maximum at $x_j = y_j^{(i)}$ and whose value becomes small as $|x_j - y_j^{(i)}|$ increases. A possible example of the functions $\phi_j(x_j - y_j^{(i)})$ for the case of 4-channel data is shown in Figure 2.

The correlation function $C^{(i)}$ associated with the ith cluster is defined as

$$C^{(i)} = \sum_{j=1}^{N} \phi_j(x_j - y_j^{(i)})$$

From the properties of the function ϕ_j it is clear that the maximum value of $C^{(i)}$ is equal to

$$C_{\max}^{(i)} = \sum_{j=1}^{N} \phi_j(0)$$

and will occur when the spectral signature $\underset{\sim}{x}$ is equal to the spectral signature $\underset{\sim}{y}^{(i)}$. It is also clear that a large value of $C^{(i)}$ will occur when the spectral signatures $\underset{\sim}{x}$ and $\underset{\sim}{y}^{(i)}$ are similar, while a small value of $C^{(i)}$ will occur when $\underset{\sim}{x}$ and $\underset{\sim}{y}^{(i)}$ are dissimilar. Thus, $C^{(i)}$ can be used as a similarity measure to determine if the pixel with a spectral

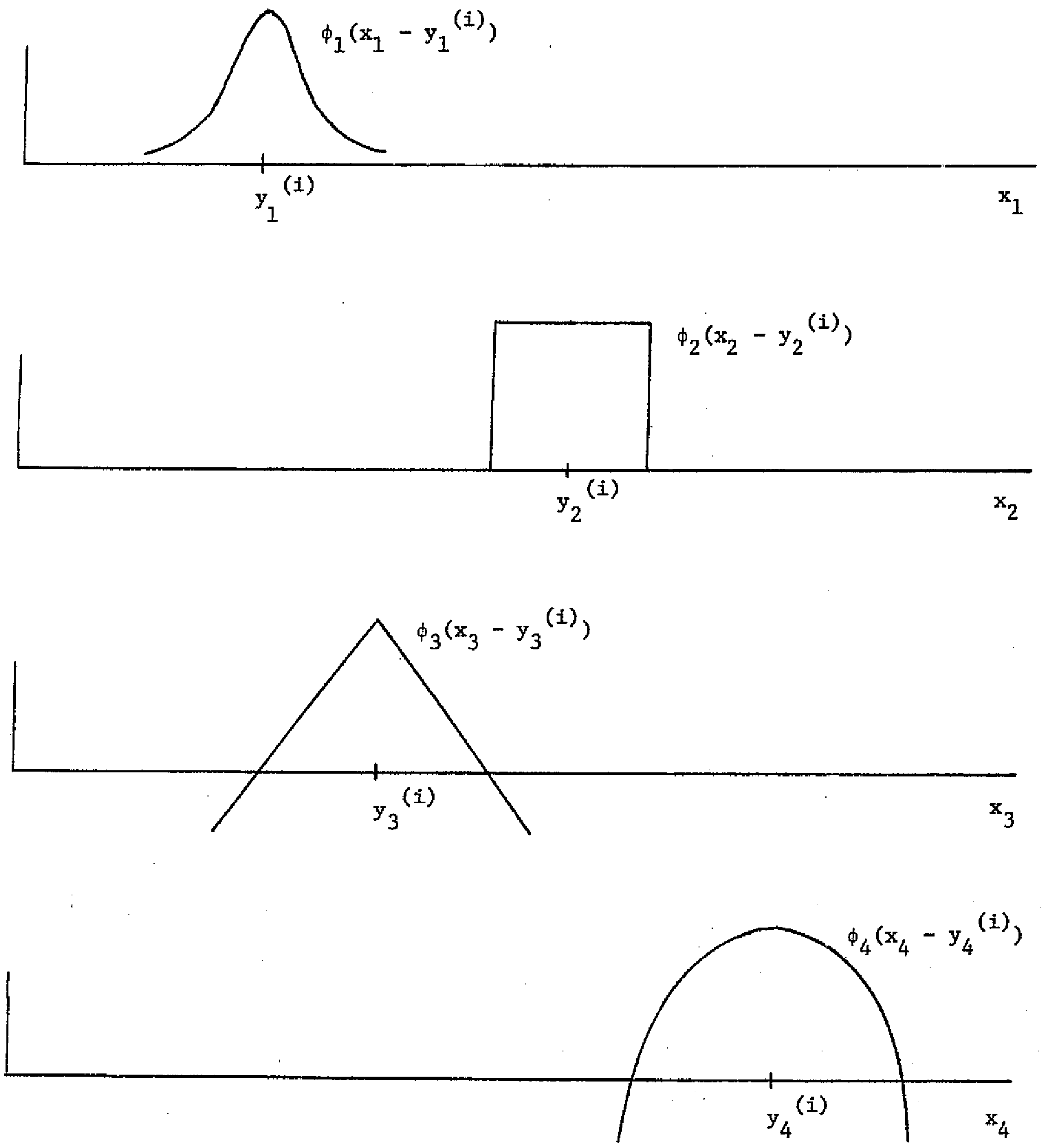

$\phi_1(x_1 - y_1^{(i)})$
$y_1^{(i)}$
x_1
$\phi_2(x_2 - y_2^{(i)})$
$y_2^{(i)}$
x_2
$\phi_3(x_3 - y_3^{(i)})$
$y_3^{(i)}$
x_3
$\phi_4(x_4 - y_4^{(i)})$
$y_4^{(i)}$
x_4

Fig 2. An Example of Possible Weighting Functions $\phi_j(x_j - y_j^{(i)})$ for the Case of 4-Channel Data.

signature $\underset{\sim}{x}$ should be assigned to the cluster whose average spectral signature is $\underset{\sim}{y}^{(i)}$. The criterion used will be to assign $\underset{\sim}{x}$ to cluster i if $C^{(i)} \geq C_{min}$ where C_{min} is a variable threshold level. In the interest of efficiency the $C^{(i)}$'s will be computed in the inverse order of cluster generation and the pixel will be assigned to the first cluster encountered for which $C^{(i)} \geq C_{min}$. If this condition does not hold for any of the clusters, then a new cluster is generated with the pixel as its first member.

The algorithm thus consists of the following steps:

```
     1)  Assign first pixel with spectral signature $x$ to cluster
         number 1. Let $\underset{\sim}{y}^{(i)} = \underset{\sim}{x}$ and set i = NCLUST = 1.
NEXT 2)  Consider next pixel with spectral signature $\underset{\sim}{x}$. When pixels
         run out, STOP
LOOP 3)  If $i \geq 1$
         Then  Compute $C^{(i)} = \sum_{j=1}^{N} \phi_j(x_j - y_j^{(i)})$
               If $C^{(i)} \geq C_{min}$
               Then assign pixel to cluster number i and update
               cluster signature $\underset{\sim}{y}^{(i)}$ .  GO TO NEXT
               Else let $i = i - 1$ and  GO TO LOOP
         Else create a new cluster by letting NCLUST = NCLUST + 1,
               i = NCLUST, and setting $\underset{\sim}{y}^{(i)} = \underset{\sim}{x}$.  GO TO NEXT.
```

In practice this algorithm may be modified so that instead of checking all of the clusters only the NBACK most recently generated clusters are checked before a new cluster is generated.

Two different versions of this clustering algorithm have been implemented.[3] One uses the linear correlation weighting function shown as the third example in Fig. 2. The second implementation uses the rectangular weighting function shown as the second example in Fig. 2. Both implementations produce satisfactory clustering results.

5. Pattern Classification Using an Iterative Method of Potentials

The goal of computer-aided pattern recognition is to automatically classify objects into distinct classes or states of nature.[4-6] If there are M such classes for a given problem and ω_i, i=1, M represents the ith class, then let $P(\omega_i)$ be the a priori probability of an object belonging to class i. If this was the only information available then the best decision rule is to always guess that an object belongs to the class for which $P(\omega_i)$ is a maximum. This rule will result in the minimum probability of error.

However, one normally has more information available with which to make a decision. This information will be assumed to be in the form of a measurement or feature vector $\underset{\sim}{x}$ where $\underset{\sim}{x}^t = [x_1, x_2, \ldots, x_n]$. The components of this vector represent measurements on the object to be classified. For example, in multispectral scanner data the components of $\underset{\sim}{x}$ represent the reflectance in each of N different spectral channels.

Having made an observation $\underset{\sim}{x}$ the a posteriori probability $P(\omega_i|\underset{\sim}{x})$ that the object belongs to class ω_i given that $\underset{\sim}{x}$ was measured is given by Bayes rule [4]

$$P(\omega_i|\underset{\sim}{x}) = \frac{p(\underset{\sim}{x}|\omega_i)\ P(\omega_i)}{p(\underset{\sim}{x})} \tag{5-1}$$

where $p(\underset{\sim}{x}|\omega_i)$ is the state conditional probability density of $\underset{\sim}{x}$ and $p(\underset{\sim}{x})$ is the total probability density

$$p(\underset{\sim}{x}) = \sum_{i=1}^{M} p(\underset{\sim}{x}|\omega_i)\ P(\omega_i)\ . \tag{5-2}$$

The decision rule is now to assign an object to class i if $P(\omega_i|\underset{\sim}{x}) > P(\omega_j|\underset{\sim}{x})$ for all $j \neq i$. Points in the feature space of $\underset{\sim}{x}$ for which

ORIGINAL PAGE IS OF POOR QUALITY

$P(\omega_i|\underset{\sim}{x}) = P(\omega_j|\underset{\sim}{x})$ lie on a decision boundary which separates class regions in the feature space. The decision rule can be generalized by introducing a loss matrix L_{ij} representing the loss associated with choosing ω_j when the actual class is ω_i. A classifier that minimizes the total expected loss is called a Bayes classifier.[4] The effect of various loss functions is to shift the decision boundaries in feature space so as to give more or less weight to a given decision.

It is common practice in statistical pattern recognition to assume that all classes have equal a priori probabilities $P(\omega_i)$ and that the loss matrix L_{ij} is equal to 0 if $i = j$ (no loss for choosing the correct class) and is equal to 1 if $i \neq j$ (a unit loss for making a mistake). Under these assumptions the Bayesian decision rule is to choose ω_i if $P(\omega_i|\underset{\sim}{x}) > P(\omega_j|\underset{\sim}{x})$ or $p(\underset{\sim}{x}|\omega_i) > p(\underset{\sim}{x}|\omega_j)$ for all $j \neq i$.

Alternatively, any monotonically increasing function of $P(\omega_i|x)$ can be used as a discriminant function $g_i(\underset{\sim}{x})$. The decision rule is then to choose class ω_i if $g_i(\underset{\sim}{x}) > g_j(\underset{\sim}{x})$ for all $j \neq i$. The logarithm of $P(\omega_i|\underset{\sim}{x})$ is often used as a discriminant function.[4]

In general, the state conditional probability densities $p(\underset{\sim}{x}|\omega_i)$ are not known. One common practice is to assume that $p(\underset{\sim}{x}|\omega_i)$ is a multivariate normal distribution and labeled training samples are used to compute maximum likelihood estimates of the mean vector and covariance matrix for each class.

There are two major potential pitfalls to this approach. First of all, if the training data for a particular class is not really normally distributed then serious errors can occur. This is particularly true if the data is multimodal and precautions (such as applying preliminary clustering algorithms) have not been taken to discover this fact. Secondly, and possibly more serious, is the fact that the number of samples needed to obtain reasonably good estimates of the mean vector and covariance matrix increases dramatically as the number of features goes up. Thus, while one would expect that adding new features to the measurement vector would increase class discrimination it is a common practical result that classification performance often deteriorates as the number of features increases beyond a certain point. This phenomenon can usually be traced to the fact that there are not enough training samples to provide an accurate estimate of the probability density parameters.

In order to overcome this problem of too many dimensions in the feature vector a wide variety of feature selection algorithms have been developed.[4-6] The goal of these algorithms is to reduce the dimensions of the feature space while at the same time trying to maintain the best possible discrimination between classes. However, the class discrimination can never be as good as when all features are used. This situation has led to the search for nonparametric methods in which the problems associated with statistical parameter estimations would be alleviated.

Nonparametric techniques have been used to estimate both the state conditional probability density $p(\underset{\sim}{x}|\omega_i)$[7] and the a posteriori probability $P(\omega_i|\underset{\sim}{x})$.[8] Alternatively, methods have been developed that determine the discriminant functions $g_i(\underset{\sim}{x})$ directly from the labeled training samples. The most popular of these techniques are the linear discriminant functions which divide the feature space into class regions by means of hyperplanes.[9] The main problem with linear discriminant functions is that there are many classification problems in which the classes may be separable with nonlinear discriminant functions but are not separable with linear discriminant functions.

The final goal of any of the classification schemes is to associate every region in feature space with a particular class (or a probability of belonging to a class) in such a way that the best possible classification accuracy is achieved in practice.

The classifier described in this section is a nonparametric classifier that produces nonlinear decision surfaces or discriminant functions by means of an iterative method that continually warps the decision surfaces in such a way that all labeled training samples remain correctly classified. When classifying an unknown object with a feature vector $\underset{\sim}{x}$, the M discriminant functions $g_i(\underset{\sim}{x})$, $i = 1,M$, are computed and the object is assigned to the class i for which $g_i(\underset{\sim}{x}) > g_j(\underset{\sim}{x})$ for all $j \neq i$.

This classifier is related to a class of methods referred to as the method of potentials.[5,10,11] In all such methods an interpolating or potential function is associated with labeled sample points. The cumulative sums of such potential functions form the discriminant functions used for classification. In the most common version of this method a potential function is added to the discriminant function only when a labeled samples is misclassified by the discriminant functions formed up to that point.[12-14] This recursive algorithm for forming the discriminant functions is applied interatively until all labeled samples are classified correctly.

The advantage of this method of potentials is that only those samples that are misclassified need to be stored to compute the discriminant functions. However, although all training samples are classified correctly there is no reason to believe that the resulting discriminant functions are related in any way to the a posteriori probabilities $P(\omega_i|\underset{\sim}{x})$ and thus there is no reason to believe that good classification results will occur with test data.

The classifier described in this section uses a modified approach in which a potential function is associated with each labeled training sample.[15,10] This approach is similar to the use of Parzen windows for the estimation of probability densities.[7] However, an iteration technique is used in which a positive weighting factor is applied each time a labeled sample is misclassified. In this way the resulting cumulative discriminant functions continually warp themselves until all labeled training samples are correctly classified.

Although this method has been recognized as a very general and powerful classification technique, it has been criticized in the past for its computational problems and excessive storage requirement.[16] However, these problems have been largely overcome in the classifier described here.

If a number of labeled samples belonging to the same class have feature vectors that are very close together in feature space then for the purpose of forming a cumulative potential function or discriminant function these many feature vectors may be replaced by a single "potential center" located at the mean of the vectors being replaced and the new single potential function is given a weight equal to the number of labeled samples that it represents. In this way the storage requirements can be kept to manageable proportions. For example, a resulting discriminant function that was formed from, say, 100 potential centers could represent an extremely complex, non-linear decision surface.

The classifier described in Section 6 checks each labeled training sample as it is presented to the classifier to see if it can be combined with an existing potential center. It does this by using a correlation clustering algorithm. In this way an efficient, but very powerful classifier is achieved.

Another unique feature of the classifier described in Section 6 is the hierarchical manner in which the training data is stored in the computer. Each labeled sample can be assigned to a class at a number of different levels of specificity. For example, bad corn could be simultaneously classified as land, agricultural land, cultivated agricultural land, corn, and bad corn. All training data can then be stored as a classification tree in which more and more detail is achieved by going further down the tree. The classifier is able to classify an object at any level in the classification tree.

Classifiers based on the method of potentials have been recognized as being superior to statistical classifiers when the amount of training data is limited.[16] This is often the case when processing multispectral scanner data since the cost of acquiring reliable ground truth can be very high. When this cost is taken into account then a more powerful classifier that can work well with a limited amount of ground truth may be more economical even if its processing time is longer.

The main advantages of the classifier described in this report can be summarized as follows:

1) It is a nonparametric classifier that works well with multimodal data and whose performance should continue to improve as the dimension of the feature vector is increased.
2) It is trained iteratively in such a way that all training data are correctly classified by the classifier.
3) It can equally well handle a large amount of training data (by using correlation clustering to reduce the number of potential centers) or a small amount of training data (by using each training sample as a potential center).
4) It can classify at various levels of detail by storing the training samples in the form of a classification tree.
5) It can be trained over a period of time, getting better and better as additional ground truth information becomes available.

5.1 Discriminant Functions Formed by an Iterative Application of Potential Functions

The method of potentials uses labeled training samples to form non-linear discriminant functions that can be used to classify test data. Let $\underset{\sim}{x}_k^i$ be the feature vector associated with the kth sample of class i. An

ORIGINAL PAGE IS OF POOR QUALITY

interpolating, or potential function $K(\underset{\sim}{x},\underset{\sim}{x}_k^{(i)})$ is defined to be a function that is maximum when $\underset{\sim}{x} = \underset{\sim}{x}_k^{(i)}$ and decreases monotonically as $|\underset{\sim}{x} - \underset{\sim}{x}_k(i)|$ increases. Specific potential functions that have been used include

$$K(\underset{\sim}{x},\underset{\sim}{x}_k^{(i)}) = \frac{1}{1 + \alpha||\underset{\sim}{x} - \underset{\sim}{x}_k^{(i)}||^2} \tag{5-3}$$

and

$$K(\underset{\sim}{x},\underset{\sim}{x}_k^{(i)}) = \exp(-\alpha||\underset{\sim}{x} - \underset{\sim}{x}_k^{(i)}||^2) \tag{5-4}$$

An estimate $\hat{p}(\underset{\sim}{x}|\omega_i)$ of the state conditional probability density $p(\underset{\sim}{x}|\omega_i)$ can be obtained by erecting a potential function $K(\underset{\sim}{x},\underset{\sim}{x}_k^{(i)})$ at each of the N_i samples of class i, adding all of these functions and dividing by N_i. That is,

$$\hat{p}(\underset{\sim}{x}|\omega_i) = \frac{1}{N_i} \sum_{k=1}^{N_i} K(\underset{\sim}{x},\underset{\sim}{x}_k^{(i)}) \tag{5-5}$$

The division by N_i in Eq.(5) accounts for the fact that there may be different numbers of samples in different classes. If all classes have equal a priori probabilities then, from Eq. (5-1), $\hat{p}(\underset{\sim}{x}|\omega_i)$ would also be proportional to an estimate of the a posteriori probability $P(\omega_i|\underset{\sim}{x})$. One might then consider using a discriminant function $G_i(\underset{\sim}{x})$ equal to $\hat{p}(\underset{\sim}{x}|\omega_i)$ given by Eq. (5-5) and then classify objects according to the following decision rule: Assign an object characterized by the feature vector $\underset{\sim}{x}$ to class i if $G_i(\underset{\sim}{x}) > G_j(\underset{\sim}{x})$ for all $j \neq i$.

On the other hand if the training data is obtained by randomly sampling all objects to be classified, then the number of training samples obtained for each class is, in some sense, related to the a priori probabilities of class membership. In particular if N_i is assumed to be proportional to $P(\omega_i)$ then by comparing Eqs. (5-1) and (5-5) it is clear that an estimate $\hat{p}(\omega_i|\underset{\sim}{x})$ of the a posteriori probabilities $P(\omega_i|\underset{\sim}{x})$ can be taken to be

$$\hat{P}(\omega_i|\underset{\sim}{x}) = A \sum_{k=1}^{N_i} K(\underset{\sim}{x},\underset{\sim}{x}_k^{(i)}) \tag{5-6}$$

where A is some proportionality constant. A useful discriminant function for class i might therefore be taken to be

$$G_i(\underset{\sim}{x}) = \sum_{k=1}^{N_i} K(\underset{\sim}{x},\underset{\sim}{x}_k^{(i)}) \tag{5-7}$$

The decision rule is to assign an object to class i if $G_i(\underset{\sim}{x}) > G_j(\underset{\sim}{x})$ for all $j \neq i$.

The location of the decision boundaries generated by the discriminant functions of Eq. (5-7) will depend on the number of training samples of each class N_i. As has been noted this shifting of the decision boundaries is meant in some sense to account for the a priori probabilities. However, there is no guarantee that the discriminant functions given by Eq. (5-7) will classify all of the labeled training samples correctly. This situation can be corrected by using an iterative error-correcting scheme that adds a weighting factor to the potential function $K(\underset{\sim}{x},\underset{\sim}{x}_k^{(i)})$ each time that the labeled sample $\underset{\sim}{x}_k^{(i)}$ is misclassified by $G_i(\underset{\sim}{x}^{(i)})$. The discriminant functions $G_i(\underset{\sim}{x})$ given by Eq. (5-7) are therefore modified by the following error-correcting algorithm.

For each labeled sample $\underset{\sim}{x}_k^{(\ell)}$, the discriminant functions $G_i(\underset{\sim}{x}_k^{(\ell)})$ are computed for all classes. If $G_\ell(\underset{\sim}{x}_k^{(\ell)}) > G_i(\underset{\sim}{x}_k^{(\ell)})$ for all $i \neq \ell$, then $G_\ell(\underset{\sim}{x})$ is left unchanged and the next labeled sample is considered. On the other hand if, for any i, $G_\ell(\underset{\sim}{x}_k^{(\ell)}) \leqq G_i(\underset{\sim}{x}_k^{(\ell)})$ then G (x) is changed to $G_\ell(\underset{\sim}{x}) + \lambda K(\underset{\sim}{x},\underset{\sim}{x}_k^{(\ell)})$ where λ is a constant. This rule is applied iteratively to all labeled samples until all of the labeled samples remain correctly classified. After convergence the resulting discriminant functions are thus given by

$$G_i(\underset{\sim}{x}) = \sum_{k=1}^{N_i} (1+\lambda C_{ik})\ K(\underset{\sim}{x},\underset{\sim}{x}_k^{(i)}). \tag{5-8}$$

where C_{ik} is the number of times that the labeled sample $\underset{\sim}{x}_k^{(i)}$ caused a change.

The discriminant functions given by Eq. (5-8) are used to classify test data by assigning an object to class i if $G_i(\underset{\sim}{x}) > G_j(\underset{\sim}{x})$ for all $j \neq i$.

5.2 Using the Potential Function Classifier in the Program GROUPX

As described in Section 3 the purpose of the program GROUPX is to classify each of the clusters that have been created by the program CLUSTX. The program has a procedure called COSTMATRIX that computes a cost matrix using ground truth data. Fig. 3 shows an example of the costmatrix that is produced by the procedure COSTMATRIX. In this figure the rows correspond to cluster numbers and the columns correspond to class numbers. The numbers within the costmatrix are equal to the number of pixels belonging to a particular cluster that are known from ground truth to belong to a certain class. Since it is desirable that all pixels in a given cluster belong to the same class one would hope that only one column in each row of the costmatrix is nonzero. In any event the cluster is assigned to that class corresponding to the column containing the largest number of pixels. This selection is indicated on the right-hand side of each row together with a percentage indicating what percentage of the total of each row this maximum number represents. A figure of 100% means that all ground truth pixels in that cluster belong to one class. This is obviously the desired state of affairs.

Printed at the bottom of each column of the costmatrix is the number of ground truth pixels that belong to clusters that have been assigned to that class. The ratio of this number to the sum of all pixels in that column is also printed as a percentage and is a measure of the percent correct classification.

Those rows in the costmatrix corresponding to clusters containing pixels for which no ground truth exists will contain all zeros. These clusters must be classified using the method of potentials. It is also possible to train the potential function classifier directly from the ground truth for individual pixels and to then classify all clusters using the method of

ORIGINAL PAGE IS
OF POOR QUALITY

```
CLUSTER/CLASS     1     2     3     4     5        TOTAL     CLASS    PERCENT
      1           0     0     0     0     0          0.        0        0.00
      2           0     0     0     0     0          0.        0        0.00
      3           0     0     0     0     0          0.        0        0.00
      4           0     0     0     0     0          0.        0        0.00
      5           0     1     3     0     0          4.        3       75.00
      6           0     1     0     0     0          1.        2      100.00
      7           0     0     2     0     0          2.        3      100.00
      8           0     0     0    12     0         12.        4      100.00
      9           0     0     0     0     0          0.        0        0.00
     10           0     0     0     0     0          0.        0        0.00
     11           0     0     0     0     0          0.        0        0.00
     12           0     1     0     0     1          2.        2       50.00
     13           0     1     0     0     0          1.        2      100.00
     14           0     0     3     0     0          3.        3      100.00
     15           0     0     0     0     0          0.        0        0.00
     16           1     1     0     0     0          2.        1       50.00
     17           4     2     2     0     2         10.        1       40.00
     18           0     0     0     0     0          0.        0        0.00
     19           0     0     0     0     1          1.        5      100.00
     20           0     0     0     0     0          0.        0        0.00
     21           0     0     0     0     0          0.        0        0.00
     22           4     0     0     0     0          4.        1      100.00
     23           1     0     0     0     0          1.        1      100.00
     24           0     0     0     0     0          0.        0        0.00
     25           0     0     0     0     0          0.        0        0.00
     26           0     0     0     0     0          0.        0        0.00
     27           0     0     0     0     0          0.        0        0.00
     28           0     0     0     0     0          0.        0        0.00
     29           0     0     0     0     0          0.        0        0.00
     30           0     0     0     0     0          0.        0        0.00
     31           0     0     0     0     0          0.        0        0.00
     32           1     0     0     0     0          1.        1      100.00
     33           0     0     0     0     0          0.        0        0.00
     34           1     0     0     0     0          1.        1      100.00
     35           0     0     0     0     0          0.        0        0.00
     36           0     0     0     0     0          0.        0        0.00
     37           0     0     0     0     0          0.        0        0.00
     38           0     0     0     0     0          0.        0        0.00
     39           0     2     0     0     3          5.        5       60.00
     40           0     0     0     0     0          0.        0        0.00
     41           2     0     0     0     0          2.        1      100.00
     42           0     0     0     0     0          0.        0        0.00
     43           0     0     0     0     2          2.        5      100.00
     44           0     0     0     0     0          0.        0        0.00
     45           1     1     0     0     0          2.        1       50.00
     46           0     0     0     0     0          0.        0        0.00
     47           0     0     0     0     0          0.        0        0.00
     48           0     0     0     0     0          0.        0        0.00
     49           0     0     0     0     0          0.        0        0.00
     50           0     0     0     0     0          0.        0        0.00
     51           0     0     0     0     0          0.        0        0.00

   TOTAL         15.   10.   10.   12.    9.        56.

  CORRECT        15.    3.    8.   12.    6.        44.

  PERCENT     100.0  30.0  80.0 100.0  66.7      78.57
```

ORIGINAL PAGE IS
OF POOR QUALITY

Fig. 3 COSTMATRIX Output

potentials. In this case the costmatrix is used only for informational purposes as an indication of how well the clustering algorithm CLUXTX performed.

The discriminant functions given by Eqs. (5-8) and (5-3) are used to classify all of the unlabeled samples. Thus, if $\underset{\sim}{y}$ is the spectral signature associated with an unclassified cluster then $\underset{\sim}{y}$ is assigned to class i if $G_i(\underset{\sim}{y}) > G_j(\underset{\sim}{y})$ for all $j \neq i$.

After all of the clusters have been assigned to a class, the input cluster tape is read and the cluster number associated with each pixel on the cluster tape is translated into a corresponding class number on the output tape.

The effectiveness of the algorithm GROUPX that uses a Potential Function Method (PFM) classifier can be demonstrated by comparing its performance to that of a Gaussian Maximum Likelihood (GML) classifier. Both types of classifiers have been used to classify ERTS data containing agricultural fields in Fayette County.

An area containing 12,726 pixels was selected of which a total of 297 pixels of ground truth was available. This ground truth consisted of six classes and was divided between training data and test data according to the following table.

TABLE OF GROUND TRUTH

	Class	No. of Training Pixels	No. of Test Pixels
1.	Soybean	116	25
2.	Corn	40	10
3.	Wheat	14	3
4.	Woods	52	16
5.	Bare Soil	13	4
6.	Clover	3	1
		238	59

When the GML classifier was applied to this data, the covariance matrix for class 6 was found to be non-positive definite (the matrix was singular) and thus this class could not be included in the classification. The remaining 58 test pixels were classified by the GML classifier and the results are summarized in the error matrix of Fig. 4. These results show that all of the test pixels are classified as either class 3 or class 4. This is undoubtedly due to the fact that an insufficient quantity of training data yields inaccurate estimates of the mean vectors and covariance matrices.

The PFM classifier GROUPL was then applied to this same data. The procedure COSTMATRIX classified the following number of clusters that were used as potential centers for training the potential method classifier:

	CLASS	No. of TRAINING POTENTIAL CENTERS
1.	Soybean	17
2.	Corn	5
3.	Wheat	1
4.	Woods	7
5.	Bare Soil	1
6.	Clover	0

A value of $\alpha = 5.0$ and $\lambda = 1.0$ in Eqs. (5-8) and (5-3) resulted in training convergence after a single iteration. A total of 51 test pixels were then classified and the resulting error matrix is shown in Fig. 5.

Both the GML and PFM classifiers were used in similar version of GROUPX. ALGOL listings of both programs and given in the Appendix. The original scanner data had been clustered into 137 clusters using CLUSTX. The overall performance of the costmatrix using the training data was about 90%. A higher percentage could have been achieved by generating more clusters. Thus, relative to this upper limit a more accurate measure of the GML classifier would be an overall accuracy of 17.24/90 = 19.2%, while the overall accuracy of the PFM classifier is 84.31/90 = 93.6%. The total processing time for the version of GROUPX containing the GML classifier and for the version containing the PFM classifier was 1 min. 17 sec. and 1 min. 29 sec. respectively. All programs were run on a Burroughs B-5500 computer.

Another version of GROUPX was tested that used a PFM classifier that trained on individual pixels rather than cluster centers obtained from the COSTMATRIX. In order to keep the number of potential centers or training data to a minimum (an actual advantage in PFM classifiers!) the test data used previously was used for training and the original training data was classified. A total of 136 pixels were classified using 79 different pixels as training potential centers. The resulting error matrix is shown in Fig. 6.

The same data was used to train the GML classifier and the result of classifying the same 136 test pixels used in Fig. 6 is shown by the error matrix in Fig. 7. The covariance matrix for class 7 was singular and therefore the three text pixels for that class could not be classified.

The results given above indicate the superiority of the new PFM classifiers over the GML classifiers. The main reasons for this improved performance include:

1) Even limited quantities of ground truth can be effectively used by the PFM classifiers while the same ground truth may yield covariance matrices that are either singular or so grossly in error as to be meaningless.
2) The pre-clustering of the training data (by using the COSTMATRIX) results in a manageable number of potential centers that represent a faithful sampling of all available training samples.
3) When the training data is not normally distributed or is even multi-modal the PFM classifiers have no problem in forming accurate decision boundaries. On the other hand, the GML classifier may produce totally inaccurate results in these instances.

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

```
                         TEST   ERROR   MATRIX

                              CLASSIFIED                        SUM      PERCENT
ACTUAL
                  1       2       3       4       5       6
  1               0       0      25       0       0       0      25        0.00
  2               0       0      10       0       0       0      10        0.00
  3               0       0       3       0       0       0       3      100.00
  4               0       0       9       7       0       0      16       43.75
  5               0       0       0       0       0       0       0        0.00
  6               0       0       4       0       0       0       4        0.00


SUM               0       0      51       7       0       0      58

PERCENT        0.00    0.00    5.88  100.00    0.00    0.00              | 17.24 |
```

Fig. 4   Error Matrix for GML Classifier

TEST ERROR MATRIX

ACTUAL	CLASSIFIED 1	2	3	4	5	6	SUM	PERCENT
1	19	0	0	0	0	0	19	100.00
2	2	4	0	0	0	0	6	66.67
3	0	0	0	0	2	0	2	0.00
4	0	0	0	18	0	0	18	100.00
5	3	0	0	0	2	0	5	40.00
6	0	1	0	0	0	0	1	0.00
SUM	24	5	0	18	4	0	51	
PERCENT	79.17	80.00	0.00	100.00	50.00	0.00		$\boxed{84.31}$

Fig. 5 Error Matrix for PFM Classifier.
Potential Centers Obtained from COSTMATRIX Clusters.

ORIGINAL PAGE IS OF POOR QUALITY

TEST ERROR MATRIX

ACTUAL	CLASSIFIED 1	2	3	4	5	6	7	SUM	PERCENT
1	51	1	0	0	0	0	0	52	98.08
2	3	5	0	5	0	0	0	13	38.46
3	0	0	0	5	0	0	0	5	0.00
4	0	0	0	52	0	0	0	52	100.00
5	0	0	0	0	0	0	0	0	0.00
6	8	0	0	0	0	3	0	11	27.27
7	0	0	0	3	0	0	0	3	0.00
SUM	62	6	0	65	0	3	0	136	
PERCENT	82.26	83.33	0.00	80.00	0.00	100.00	0.00		$\boxed{81.62}$

Fig. 6 Error Matrix for PFM Classifier
Potential Centers Obtained from Individual Pixels

TEST ERROR MATRIX

ACTUAL	CLASSIFIED 1	2	3	4	5	6	SUM	PERCENT
1	0	0	52	0	0	0	52	0.00
2	0	0	13	0	0	0	13	0.00
3	0	0	5	0	0	0	5	100.00
4	0	0	7	45	0	0	52	86.54
5	0	0	0	0	0	0	0	0.00
6	0	0	11	0	0	0	11	0.00
SUM	0	0	88	45	0	0	133	
PERCENT	0.00	0.00	5.68	100.00	0.00	0.00		$\boxed{37.59}$

Fig. 7 Error Matrix for GML Classifier

ORIGINAL PAGE IS
OF POOR QUALITY

6. A Nonparametric, Hierarchical Classifier

In many classification problems, the taxonomy in which objects are to be placed is intrinsically hierarchical. Figure 8 illustrates a possible taxonomy for use in classifying areas of a photograph of the earth's surface. At the lowest level of classification (level 1) a pixel is classified as either land or water. The next level of classification further subdivides the first level. The taxonomy shown indicates no further classification of water (for example into warm or cold) but a pixel classified as land at level 1 might further be classified at level 2 as urban, woods, bare soil, or agricultural. The algorithm described in this section is able to classify an object to any desired level in such a hierarchical classification system.[17]

We assume a set of labeled training data, each of which has been classified. In order to classify an unknown object at level 1 (water or land) we need two sublists of our training data, one of all training data labeled water, the other of all training data labeled land. Using the iterative method of potentials described in Section 5.1, the discriminant function for each of these sublists is evaluated at the point in feature space corresponding to the spectral signature of the unknown pixel. Then the unknown pixel is determined to be either water or land according to which sublist yields the largest discriminant function value*. To further classify the unknown pixel at level 2, we would need four other sublists of labeled training data: urban, woods, bare soil and agricultural. A training datum classified as, say, corn (Fig. 8) would be included in four sublists.

---

*In the algorithm described below, a further requirement is made; namely that this largest discriminant function value must exceed some minimal threshold. Otherwise, the object is unclassifiable at that level.

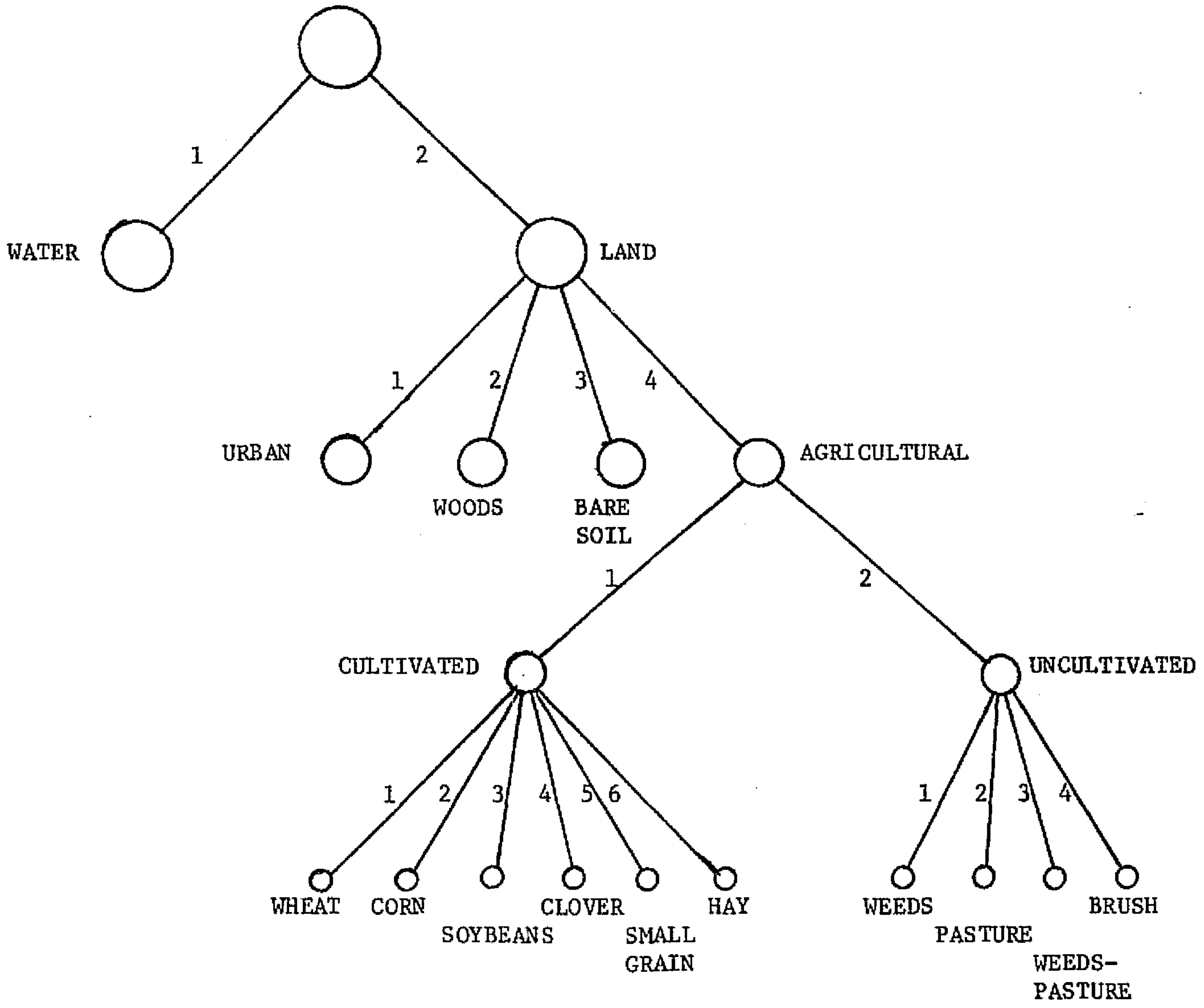


Fig 8 Sample Classification Tree for ERTS Multispectral Scanner Data

Clearly, to implement hierarchical classification we need rapid access to various sublists of the training data. Representing the training data as subsets of lists is facilitated through the use of a multi-linked data structure. Each of the labeled training data is stored in a node

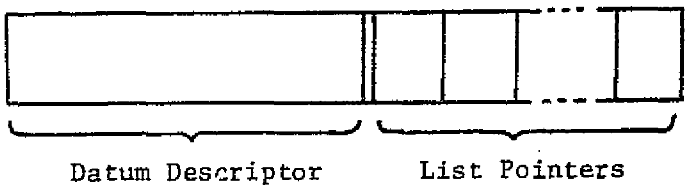


Fig. 9 Node for Storing Training Data

as shown in Fig. 9. The datum "value" is stored along with a set of links or pointers. These pointers are used to associate that datum with those sublists to which it belongs. For each node, a pointer is required for each level of classification. If a given datum is the only member of a sublist at, say, level 2, the corresponding pointer is set to null (zero); otherwise it is set to point to the most recent datum that is a member of that sublist. The classification tree of Fig. 8 has 4 levels; hence, the nodes for storing the training data will each have 4 pointers.

To facilitate the addition and deletion of training data nodes, it is convenient to include one more pointer with each node. This pointer is used simply to link all of the data into a list. If any training datum is to be discarded, this pointer can be used to link the unused node storage box to the free storage stack. When a new datum is required, the needed node storage box is removed from the free storage stack. This memory management technique insures that all available memory for training data storage is accessable.

The actual arrays used to implement the multi-linked list structures are now described. The i^{th} node box is composed of the i^{th} row (or element) of the arrays described in Table 1

Name	Dimension
FEATURE	NROWS x NCHAN
CLASS	NROWS x NLEV
CLINK	NROWS x NLEV
CLISTLINK	NROWS

Table 1 Arrays used for multi-linked list storage

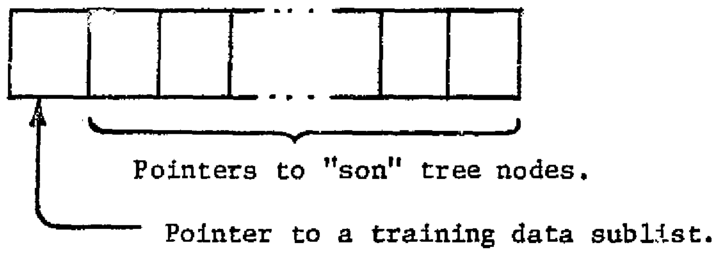
The array FEATURE is used to store the training data feature vectors, each of which is a vector of length NCHAN. The array CLASS is used to store the classification data. Each datum is classified according to the given classification tree, where the branches are labeled sequentially as shown in Fig. 8. NLEV is the depth of the classification tree and therefore represents the length of a classification vector. Each row of the array CLINK holds the pointers needed to link that node into each of the respective sublists. Finally, the array CLISTLINK is used to link all of these nodes together in one list.

The unused locations are also linked into a list (stack) by means of CLISTLINK. The location of the top of this stack is held in the variable CAVAIL.

The data structure described above is adequate for storing all of the sublists of the labeled training data. However, it does not inherently provide rapid access to each sublist. For example, if it is known that a given training datum is a member of the "land" sublist, then its level 1 pointer points to the "next" element of the "land" sublist. However, no mechanism is provided for locating the beginning of the "land" sublist. Yet another data structure is required to provide access to the beginning of any desired sublist.

The data structure used here to access the sublists is a tree having a form similar to that of Fig. 8. Each node of the tree contains the address of the beginning node of a training data sublist. By "climbing" through this tree any desired sublist can be located.

A typical tree node is shown in Fig. 10. Storage is



Pointers to "son" tree nodes.
Pointer to a training data sublist.

Fig 10 A typical tree node

provided for a pointer to the head of a sublist. Also provided are pointers which locate the "sons" of that tree node. If the tree node has no sons (i.e., a terminal or "leaf") then all of the son pointers are null (zero).

In the present implementation, a fixed number of son pointers is used; this number must equal the maximal "fan out" of the classification tree. The subclasses of a node are denoted by integers: 1,2,3,..; the absence of the i^{th} subclass (or a sublist) is denoted by a null (zero) value in the i^{th} son pointer position.

An array TNODE, with dimensions NODES x NSONS, is used to store the access tree. The i^{th} row of this array stores the i^{th} node box. The unused storage locations are linked together in a free storage stack using the first column elements of the array. The location of the top of this stack is held in the variable TAVAIL. The location of the root of the tree is held in the variable TROOT.

In summary, two data structures are used for storing and accessing the training data. A multi-linked list structure is used to represent the data as a set of mutually inclusive lists (or sublists). A tree structure is used to provide access to the various sublists. Together they provide the data storage and access needed to support the algorithm for hierarchical classification by the method of potentials.

6.1 The Classifier CHIMP

The algorithm described below is called CHIMP (for Classification Hierarchy using an Iterative Method of Potentials). There are three major phases in this algorithm: training data input, training, and classification. These phases of the algorithm are discussed separately. Reference is made to the listing of the algorithm in Appendix D.

6.1.1 Training Data Input

The function of this phase of the algorithm is to enter the training data (from TDFILE) into the multi-linked list structure and to construct the corresponding sublist access tree. As each labeled training datum is received by the procedure INPUT, the access tree is extended, if necessary, by the procedure SETTREE to accomodate that new datum. SETTREE leaves behind a TRAIL vector of tree node locations which contain pointers to the sublists in which the new datum will be included. (These sublists may, of course, be empty, in which case the new datum will be the first element in the lists.)

An important task carried out by the procedure INPUT is that of clustering the training data. Stated simply, training data that are sufficiently "close" to each other in feature space will be combined into a single datum located in feature space at the center of gravity of the included data. The number of data associated with each "cluster" or "potential center" is stored in a corresponding element of the vector WEIGHT (of length NROWS). A new datum will be clustered with an existing one if the new datum falls within the WINDOW of the existing one, which WINDOW is a hypercube centered on the existing datum with half width WINDOWSIZE. The procedure INWINDOW determines whether or not the new datum is to be clustered.

ORIGINAL PAGE IS
OF POOR QUALITY

Clustering is only done with data that match at all levels of classification. Therefore, after deploying SETTREE, the procedure INPUT searches the deepest sublist, to which the new datum would belong, to see if the new datum is in the WINDOW of any existing datum. If so, then they are clustered and INPUT terminates. If not, then the new datum is placed in a new storage node which is then linked into each of the sublists identified in TRAIL.

6.1.2 Training

The potential function used in this algorithm has the form

$$f_j(\underset{\sim}{x},\underset{\sim}{x}_j) = \mathrm{WEIGHT}(\underset{\sim}{x}_j)\ \frac{1 + \lambda \mathrm{COUNT}_j}{1 + \alpha\|\underset{\sim}{x}-\underset{\sim}{x}_j\|^2} \tag{6-1}$$

where $\underset{\sim}{x}_j$ is a feature vector of a training data cluster,

$\underset{\sim}{x}$ is the feature vector of the unknown datum,

λ and α are scalar parameters,

$\mathrm{WEIGHT}(\underset{\sim}{x}_j)$ is the number of training data associated with $\underset{\sim}{x}_j$,

$\|\cdot\|$ is the Euclidean norm,

and COUNT_j is a counter used in training as described below.

The discriminant function for a subclass (sublist of potential centers) is

$$D_k(\underset{\sim}{x}) = \sum_j f(\underset{\sim}{x},\underset{\sim}{x}_j) \tag{6-2}$$

where the summation is over all elements in the sublist corresponding to the subclass k.

An example will illustrate the use of these discriminant functions. Suppose we wish to classify an unknown object with feature vector $\underset{\sim}{x}$ at level 1 as either water or land (Fig. 8). Two discriminant functions are required: D_W and D_L. D_W is defined over the subclass of all training data labeled "water," and $D_L(\underset{\sim}{x})$ is defined over the sublist labeled "land." $D_W(\underset{\sim}{x})$ and $D_L(\underset{\sim}{x})$ are evaluated and $\underset{\sim}{x}$ is classified according to which discriminant function value is greater. If the unknown $\underset{\sim}{x}$ is classified at level 1 as land then it can further be classified at level 2 as "urban," "woods," "bare-soil" or "agricultural." To do this classification at level 2, four discriminant function values are needed, corresponding to the four training data sublists. Again, the unknown $\underset{\sim}{x}$ is classified according to which discriminant function value is greatest.

Implicit in this algorithm is the requirement that the potential centers, resulting from the training data, be themselves classified correctly by the discriminant functions. If the value of COUNT_j in Eq. (6-1) is zero it usually happens that some of the potential centers would not be correctly classified by the method. To overcome this shortcoming, COUNT is introduced into the potential function and adjusted until each potential center is correctly classified by the discriminant functions.

The adjustment of the COUNT_j's is an iterative procedure as described in Section 5. During each pass, the classification of each member of each sublist is checked by the procedure CLASSIFIEDCORRECTLY. If the classification is wrong, the value of COUNT_j for that sublist element is incremented. Note that since each potential center can belong to as many as NLEV sublists, then each potential center may have NLEV values of COUNT associated with it, one for each sublist to which it belongs. If, during the first pass, any sublist element were incorrectly classified, then the whole procedure of checking the sublist elements is repeated a second time to determine whether or not the new COUNT values were sufficient to yield accurate classification. This procedure is repeated until all sublist elements are correctly classified (or until a limit on the number of these iterations is reached). This completes the training phase of the algorithm.

The procedure TRAIN is the driving procedure for this phase. It executes training passes by deploying TREECHECKER until training is successful, or for a maximum of 20 times. Also, it reports on the results of each pass.

The procedure TREECHECKER traverses the sublist access tree and deploys CHECKSUBLIST for each sublist accessed by the tree. TREECHECKER returns a Boolean value indicating whether or not all of the elements in all of the sublists were classified correctly.

The procedure CHECKSUBLIST traverses a sublist and deploys CLASSIFIED-CORRECTLY to determine whether or not each list element is correctly classified. If a list element is not classified correctly then the corresponding COUNT is incremented. CHECKSUBLIST returns a Boolean value indicating whether or not all elements in the sublist are classified correctly.

6.1.3 Classification

The previous example on the use of the discriminant functions given in Section 6.1.2 describes the general technique for classification of an unknown. The procedure CLASSIFY carries out the classification of an unknown feature. The classification is carried out only to a depth specified by the parameter MAXLEVEL (but, of course, not to exceed NLEV - the maximal depth of the tree). The unknown, to be classified by CLASSIFY, is held in the vector NEWFEATURE (which is taken from a row of the input array SIG). The vector of classification results produced by CLASSIFY is held in NEWCLASS.

The operation of CLASSIFY can be understood by following it through one level of classification. A pointer P is set to the root of the access tree. The local variables BIGCLASS and BIGVALUE are set to zero at the beginning of each new level of classification. Each of the sublist discriminant functions is evaluated and compared in turn with the current value of BIGVALUE. If the discriminant function value is greater than BIGVALUE, then BIGVALUE is replaced and BIGCLASS is replaced by the sublist index. The sublists are accessed through the sublist pointers and those tree nodes that are the sons of the root node. TNODE [TROOT, 1+K] points to the Kth son of TROOT. Therefore, TNODE[TNODE[TROOT, 1+K],1]points to the sublist corresponding to the Kth son of TROOT.

When all of the sublist discriminant functions for the sons of TROOT have been compared to BIGVALUE, then BIGVALUE will contain the value of the largest discriminant function and BIGCLASS will contain the corresponding classification. However, before the assignment of BIGCLASS to NEWCLASS is made, it is required that BIGVALUE exceed THRESHOLD. The a priori assumption here is that if the greatest discriminant function value falls below THRESHOLD then no proper classification can really be made. In this event, -1 is placed at the appropriate level in NEWCLASS and the procedure terminates.

However, if BIGVALUE exceeds THRESHOLD then BIGCLASS is placed in the proper (first) element of NEWCLASS. Then the local pointer P is moved down the tree one level to the Kth son of TROOT, where K = BIGCLASS. This process is then repeated until MAXLEVEL is reached or until a terminal is reached.

In summary, the major advantages of the classifier CHIMP include:

1) Labeled samples can be represented by a hierarchical tree structure and unknown objects can be classified at any level of the tree. Labeled samples that are known at only a certain level can be used to train the classifier up to that level.

2) The classifier can produce good results with a limited amount of training data. This is in sharp contrast to parametric classifiers such as a Gaussian maximum likelihood classifier for which considerable training data is required, particularly for higher dimensional feature vectors.

3) The classifier can operate well when large quantities of training data are used. Previous attempts to use potential function methods with large amounts of training data have been plagued with computational difficulties. CHIMP incorporates an automatic clustering algorithm that reduces the training samples to a manageable number of potential centers. These potential centers represent a faithful sampling of all available training samples.

4) The classifier CHIMP can produce very general, nonlinear decision boundaries. These decision surfaces can be used to accurately classify multi-modal as well as unimodal data.

7. An Interactive Color Display for Multispectral Imagery Using Correlation Clustering

Two distinct but complementary approaches to the processing of multispectral scanner data have been followed. One approach focuses on digital processing and has as its goal the classification of each ground resolution element, or pixel, in a given area. Sections 3-6 of this report describe an example of this approach that uses correlation clustering and nonparametric classification techniques to classify each pixel. The second general approach uses a variety of techniques to produce color maps of the ground area that are suitable for visual inspection and interpretation by humans. One common method is to use the intensity of one color (red, green, or blue) to represent the intensity of the reflected energy in one of three channels. If these three color images are superimposed (either photographically or with a color video system) then a full color map is obtained.

There are a number of limitations to the color maps produced in this way. First of all, since one color is associated with one particular spectral channel of the data it is difficult to produce a map that uses data from more than three different spectral channels. On the other hand, multispectral scanners with up to 24 spectral channels have been built. Even if one uses data from multiple-passes of the 4-channel ERTS multispectral scanner, then 8, 12, or 16 effective channels of data (combinations of spectral and temporal) would not be uncommon.

In an effort to include information from more than three channels a number of digital processing techniques, including various clustering methods, have been developed. The results of such digital processing can be used to produce color maps with display systems such as NASA's PMIS-DAS system at the Johnson Space Center in Houston.

ORIGINAL PAGE IS
OF POOR QUALITY

Is it possible to process multispectral scanner data in an unsupervised manner and produce color classification maps interactively in real time? In this section we will describe the design of such an interactive color display system that uses correlation clustering techniques to produce color maps of multispectral imagery in real time.[18]

Fig. 11 illustrates how the system would be used. An operator sits at the color display screen and has access to a number of control knobs located on the console. The color display contains an image of a certain ground area made up of, say, 500 x 500 pixels. The operator can adjust the knobs such that the entire screen is a single color. Additional adjustment will produce a broad level classification map in which perhaps all water appears blue, agricultural land appears green and forests appear red. Further adjustments might result in only the agricultural fields appearing in color with different colors representing different types of crops. In other words, the operator can "tune in" to as much detail as he wishes using his own judgment to interact with the image causing it to change in real time.

It is important to understand that the processing that is going on is entirely unsupervised in the sense that no a priori ground truth information is used. On the other hand the operator "supervises" the processing in an interactive mode and may very well use a priori information that he has about the general nature of the area in order to produce a useful map.

Obtaining good ground truth information may well be the most expensive part of supervised digital pattern recognition systems. The color maps produced by the system described in this section could prove to be very useful in identifying meaningful ground truth areas. This is true because a particular color on the map represents a localized region in the N-dimensional feature space associated with the N-channel multispectral data.

ORIGINAL PAGE IS
OF POOR QUALITY

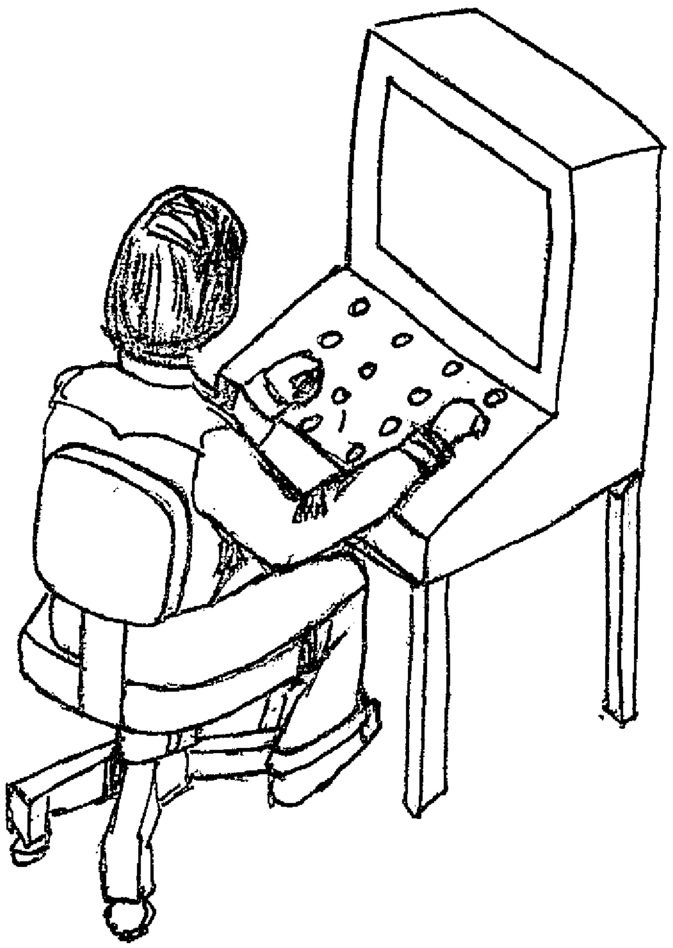


Fig. 11. Operator Processing Multispectral Scanner Data on Real-Time Interactive Color Display System

For many applications such as the production of land-use maps, the maps produced by this system may be the only type of processing of the scanner data that is required. In any event it seems clear that such a device would greatly increase the productive output of a group involved in the processing of multispectral scanner data.

In Section 7.1 the general method by which correlation clustering techniques can be used to produce color maps will be described. Various technologies, including digital, optical, and analog, that might be capable of producing the color maps in an interactive and real time environment will be surveyed and evaluated in Section 7.2. A hybrid system in which the correlation clustering is accomplished with analog circuitry is described in Section 7.3. Finally, Section 7.4 presents conclusions and recommendations for future development.

7.1 Correlation Clustering Images from Multispectral Scanner Data

What does an N-channel multispectral image look like to a human observer? Or, alternatively, how can the information contained in N-channels of multispectral scanner data be presented in a form that is readily understood by a human observer? Inasmuch as the eye is able to distinguish a wide variety of color shades and hues it would seem advantageous to use a color display to present the multidimensional information contained in the scanner data. In particular, the goal will be to associate a given shade of color with a particular localized region in the N-dimensional feature space. The size of a particular localized region and the color associated with it should be under the interactive control of the operator.

The color c of a given pixel will be some combination of the three primary colors red, r, green, g, and blue, b. That is,

$$c = C_R r + C_G g + C_B b \tag{7-1}$$

where C_R, C_G, and C_B are the proportions of red, green, and blue respectively. (For a color video tube C_R, C_G, and C_B could be the voltages applied to the red, green, and blue guns respectively.) The values of C_R, C_G, and C_B are determined by the following correlation clustering method.

Let $\underset{\sim}{x}$ be the N-channel spectral signature associated with a particular pixel. That is, $\underset{\sim}{x}^T = [x_1, x_2, \ldots, x_n]$. Let $\underset{\sim}{y}^{(i)}$ be a reference spectral signature associated with the color i (i = R, G, or B). Let $\phi_j\ (x_j - y_j^{(i)})$ be a weighting function associated with channel j whose value is a maximum at $x_j = y_j^{(i)}$ and whose value becomes small as $|x_j - y_j^{(i)}|$ increases. (For a possible example of the functions $\phi_j\ (x_j - y_j^{(i)})$ for the case of 4-channel data, see Figure 2 in Section 4.)

The correlation function C_i associated with the color i (i = R, G, or B) is defined as

$$C_i = \sum_{j=1}^{N} \phi_j\ (x_j - y_j^{(i)}) \qquad (7\text{-}2)$$

From the properties of the function ϕ_j it is clear that the maximum value of C_i is equal to

$$C_i^{MAX} = \sum_{j=1}^{N} \phi_j(0) \qquad (7\text{-}3)$$

and will occur when the spectral signature $\underset{\sim}{x}$ is equal to the reference spectral signature $\underset{\sim}{y}^{(i)}$. It is also clear that a large value of C_i will occur when the spectral signatures $\underset{\sim}{x}$ and $\underset{\sim}{y}^{(i)}$ are similar, while a small value of C_i will occur when $\underset{\sim}{x}$ and $\underset{\sim}{y}^{(i)}$ are dissimilar. Thus, if the three reference signatures $\underset{\sim}{y}^{(i)}$ are well separated then, for example, a pixel with a spectral signature $\underset{\sim}{x} = \underset{\sim}{y}^{(R)}$ would appear red. Similarly, pixels with spectral signatures $\underset{\sim}{x} = \underset{\sim}{y}^{(G)}$ and $\underset{\sim}{x} = \underset{\sim}{y}^{(B)}$ would appear green and blue respectively. Other pixels with arbitrary spectral signatures x would have colors given by (7-1) and (7-2).

An example of the locations of the three reference signatures $\underset{\sim}{y}^R$, $\underset{\sim}{y}^G$, and $\underset{\sim}{y}^B$ for the case of 2-channel data is shown in Figure 12. In this figure a "region of influence" is shown as a solid curve surrounding each color center. The size of each region is representative of the "widths of the corresponding weighting functions ϕ_j. In a real time interactive system the operator would be able to vary both the color centers $\underset{\sim}{y}^{(i)}$ and the size of each "region of influence" surrounding each color center. In this way the operator can watch as the display changes in real time as the result of varying the different parameters. Large regions of influence corresponding to wide ϕ_j functions will result in color displays in which large areas with different spectral signatures will appear as (nearly) the same color. On the other hand, narrow ϕ_j functions can be used to isolate in a single color only those pixels with a particular spectral signature. By this interactive mode of operation it should be possible to extract the maximum amount of information from the multispectral data in a minimum amount of time.

The next section will consider a number of technologies that might be used to make the type of interactive color display system that has been described above.

7.2 Interactive Displays Using Digital, Optical, or Analog Systems

When thinking of an interactive color display that is to operate in real time one thinks first of a TV type of video display system. Assuming a 500 x 500 pixel display that must be refreshed every 1/30 sec., one sees that a 7.5 MHz data rate is required to refresh the video display. Such systems are available and in use today. However, this will simply display a single image and does not process the multispectral data in any way.

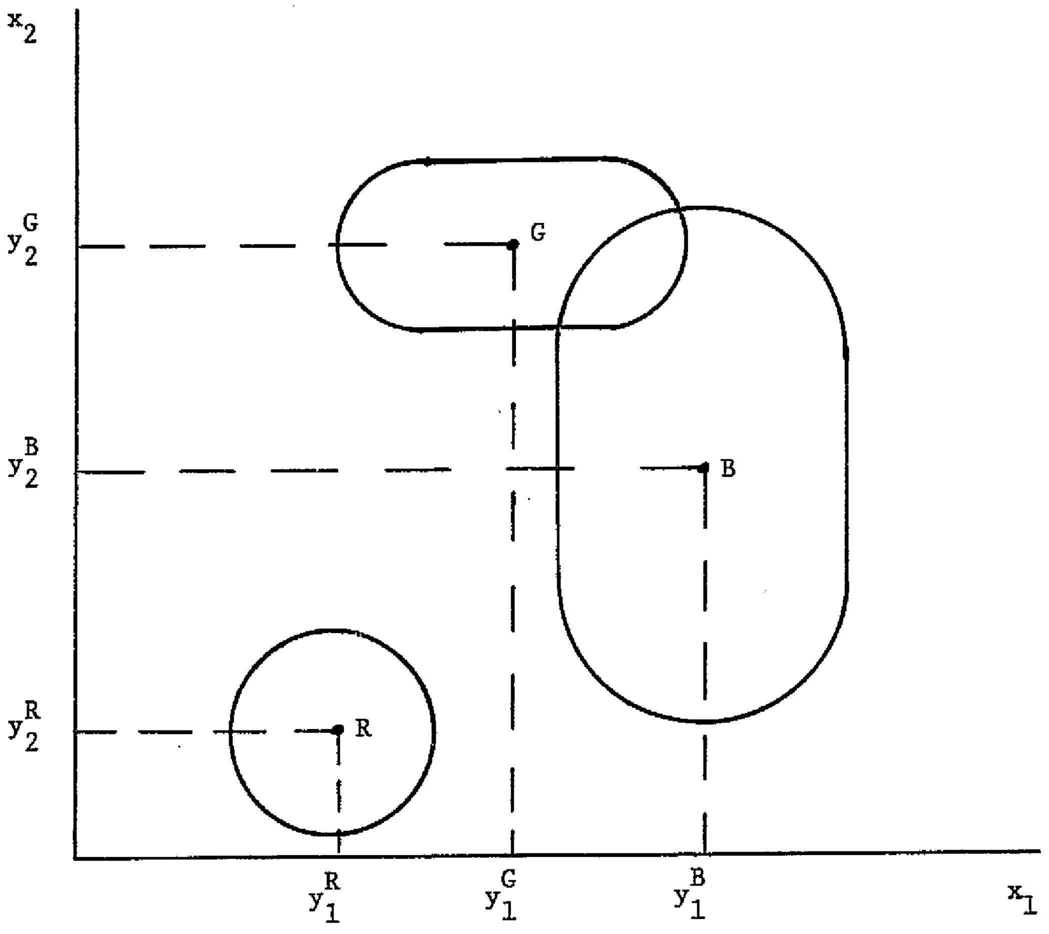


Fig. 12 Localized Color Regions in a Two-Dimensional Feature Space

What is desired is to be able to change the correlation functions C_i given by Eq. (7-2) in "real time" as observed by the operator. Suppose one tries to do this digitally. Assume that the calculation of a single value of ϕ_j requires only 5 basic operations each taking 1 μsec. For ERTS data this calculation must be done for each of the four channels and the results added (assume 1 μsec per add) to obtain C_i in Eq. (7-2). Thus, it would take 23 μsec to compute C_i for each of the three colors. Therefore, each of the 250,000 pixels would require 69 μsec of computation which means that it would take over 17 sec. to change the video picture. This is obviously not the real time operation that is desired.

The basic problem with digital computations is that there are too many pixels (250,000) and one can therefore afford to spend only about 1 μsec per pixel if the entire calculation is to be completed in some fraction of a second. This suggests that a substantial amount of parallel processing must be done if real time operation is to be achieved. Although digital computers with substantial parallel processing capabilities have been designed and built (such as the ILLIAC IV), there are major problems with their use and they would not be suitable for use in the small type of dedicated system envisioned here.

Optical processing in one sense offers the ultimate in parallel processing. The author[2] has previously suggested a method by which holographic correlation techniques could be used to produce classification maps of a type similar to those described in Section 7.1. In such a system all of the pixels are processed simultaneously at the speed of light. However, a real time system would require a real time input transducer capable of changing coded data for all pixels at video rates as well as a real time medium for recording the holographic filters. While a number of such real time devices and recording media are being developed in various laboratories, none at the present time possesses all of the properties that would be required for the type of interactive system being discussed here.

Additionally, in order to make a color display it would be necessary to construct an elaborate system containing lasers of three different colors. It is clear that such an interactive real time system using coherent optical processing is not within the current state of the art.

Returning then to the color TV video display, is there any way that the processing described by Eq. (7-2) can be done in real time? The next section will describe a hybrid system in which this interactive processing is accomplished with electronic analog circuits.

7.3 Design of an Interactive Correlation Clustering Color Display System

In this section an interactive system that will process ERTS multispectral scanner data in real time will be described. An overall block diagram of t [illegible] stem is shown in Fig. 13. The scanner data for an area of up to [illegible] pixels is transferred from magnetic tape to a high speed magnetic disk using a minicomputer which serves as a high speed buffer. Up to 250,000 bits can be stored in a single track on the disk. Thus, eight parallel tracks can store the 8-bit per pixel data for an entire TV frame for one of the spectral channels. Thirty-two tracks can then store the data for all four spectral channels. The disk rotates at 1800 rpm so that data for a complete TV frame is read every 1/30 sec.

The 32 bits representing the spectral signature for a given pixel are read from the disk in parallel with 32 fixed head transducers. This data is fed through four 8-bit digital-to-analog converters. Thus, four voltages (V_1, V_2, V_3, V_4) representing the spectral signature of a single pixel are available simultanously. These four voltages are fed into an interactive analog processor containing analog circuits that process the data. This processor contains the interactive controls that determines the nature of the processing. The output of this analog processor consists of three voltages that go to the three color guns of the video display.

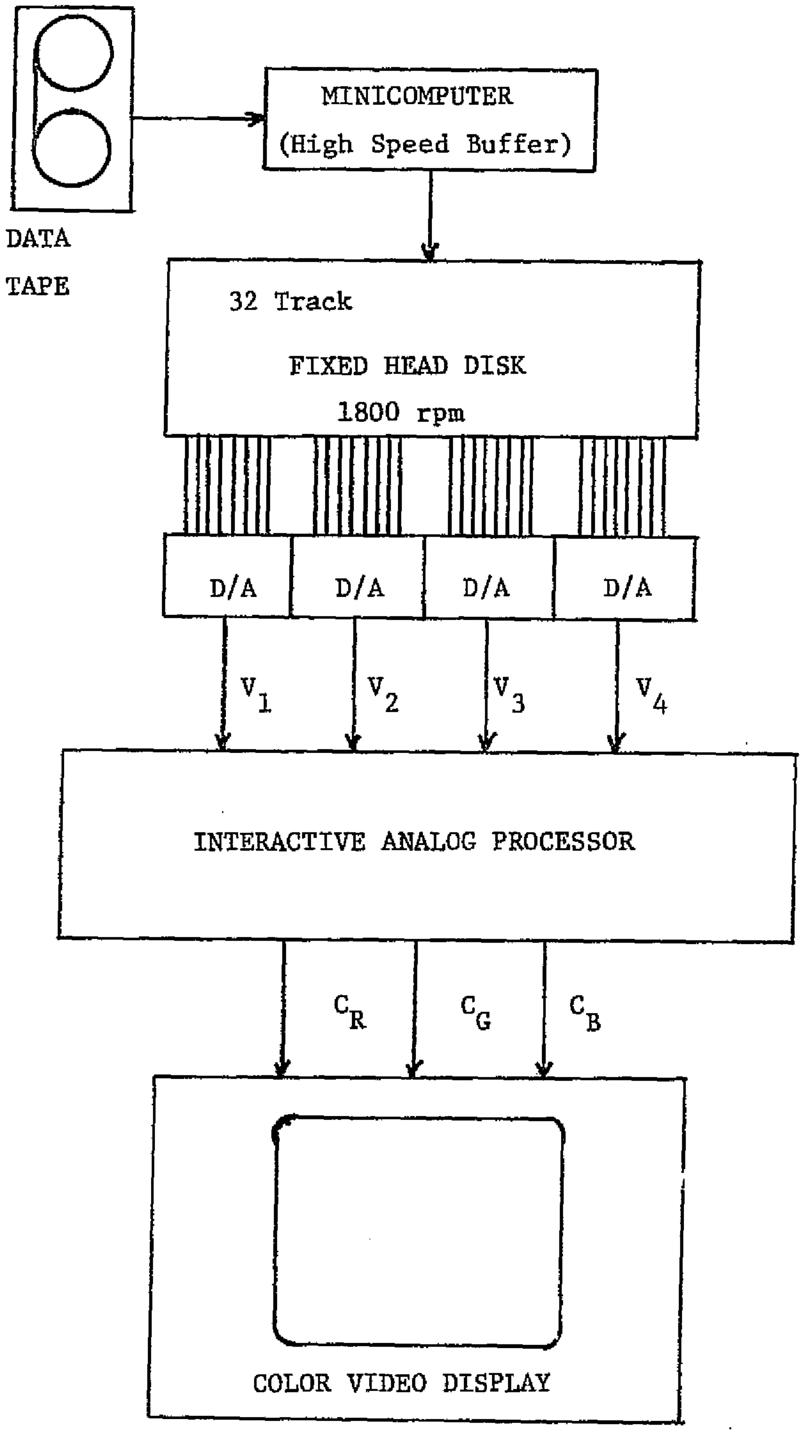


Fig. 13 Interactive Color Display System for Multispectral Imagery

The analog processor contains three similar circuits as illustrated schematically in Fig. 14. Each of these three circuits is associated with one of the three colors (red, R, green, G, and blue, B). Each of these color circuits contains two control knobs per spectral channel. Thus, there are eight variable controls for each of the three color circuits, or a total of 24 control knobs for the entire analog processor.

Each of the three color circuits making up the interactive analog processor is of the form shown in Fig. 15. The variable voltages V_a, V_b, V_c, and V_d represent a reference spectral signature that is to be correlated with the spectral signature of a given pixel which is coming from the digital-to-analog converters. The gains α_1, α_2, α_3, and α_4 of the four differential amplifiers are also under the interactive control of the operator. The values of the voltage at different points in the circuit are indicated in Fig. 15. An example of the four voltages entering the output summing amplifier in Fig. 15 as a function of V_1, V_2, V_3, and V_4 for particular settings of V_a,V_b,V_c,V_d,α_1,α_2,α_3, and α_4 is shown in Fig. 16. It is clear that the output of the summing amplifier is the correlation C_i given by Eq. (7-2). Three such outputs from the three color circuits in Fig. 14 are then combined in a color TV tube to produce a particular color as indicated by Eq. (7-1).

The entire 500 x 500 pixel TV frame is refreshed every 1/30 sec and thus the whole picture is changed in real time as the controls of the interactive analog processor are varied by the operator. These controls allow the operator to move the locations of the three color centers in feature space and to vary the size of the "region of interaction" for each color (See Fig. 12). The system described above for 4-channel data can be extended in a straightforward way to accommodate large numbers of spectral channels. Fixed head magnetic disks exist that could handle up to 24 channels of multispectral scanner data.

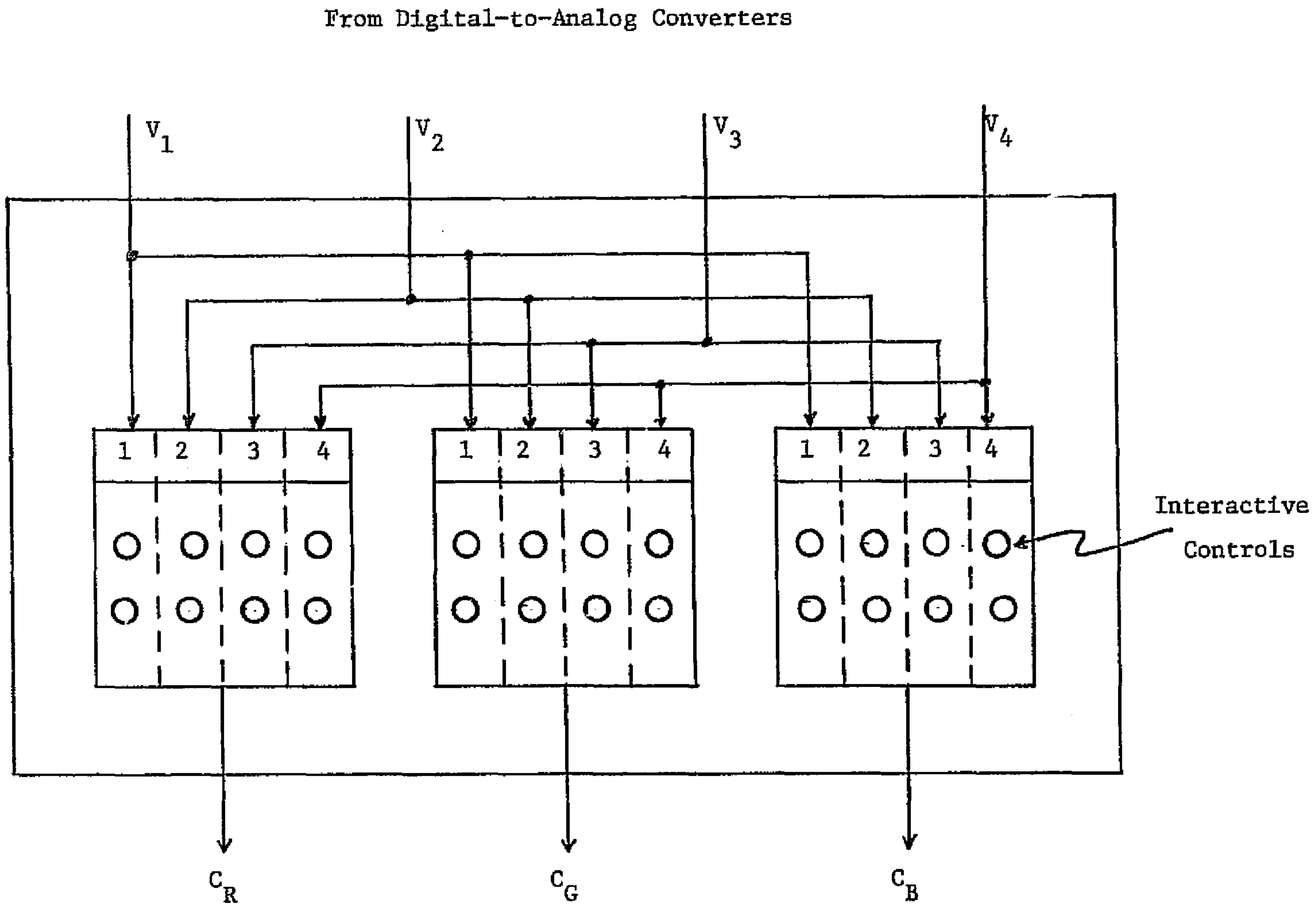
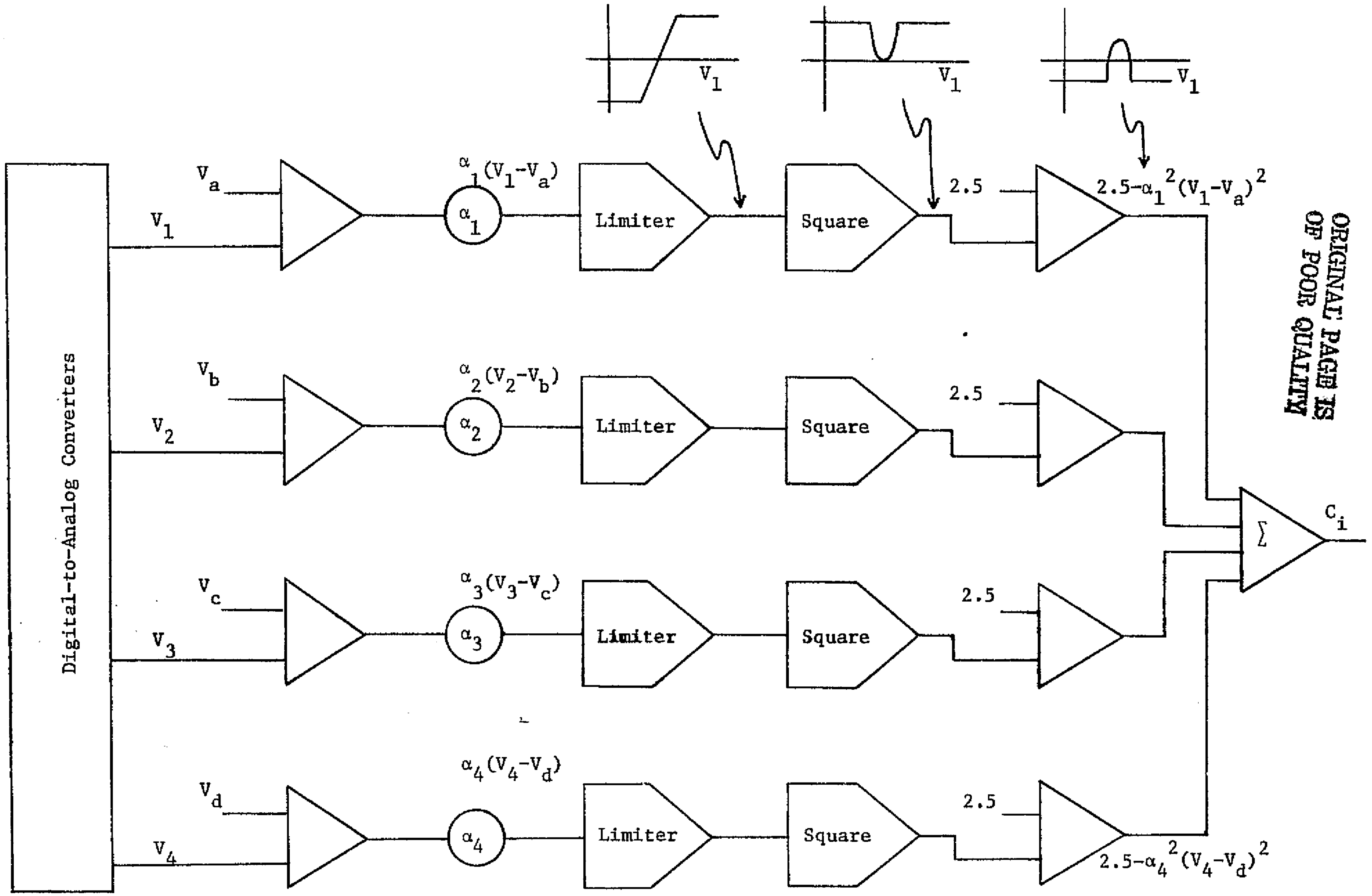


Fig. 14 Interactive Analog Processor



ORIGINAL PAGE IS OF POOR QUALITY

Fig. 15 Schematic for Each of the Three Color Circuits in the Interactive Analog Processor

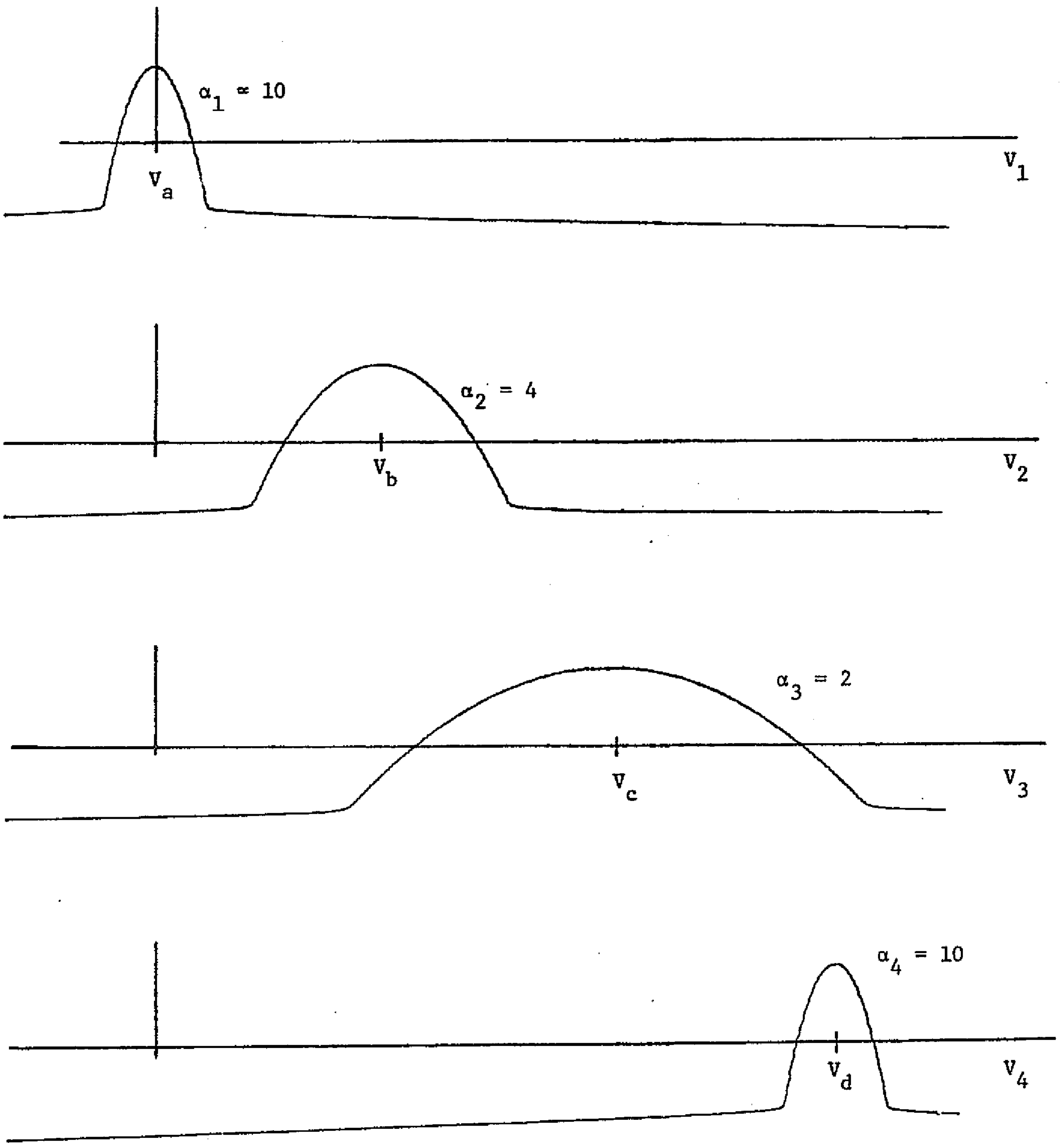


Fig. 16 Example of Signals Entering Output Summing Amplifier in Fig. 6

7.4 Conclusions

This section has described a new method of processing multispectral scanner data in a real time interactive environment. The result of the processing is a color video display of up to 500 by 500 pixels in which a given color represents a particular localized region of feature space. The size and location of these localized regions of feature space are under the interactive control of the operator. Thus, the user can elect to look at as broad or as narrow a region of feature space as he wishes.

The interactive system for processing 4-channel data contains 24 control knobs that the operator can vary. In general the number of knobs will be 6xN where N is the number of spectral channels. The ultimate goal would be to have the computer control the knobs (with perhaps some fine tuning by the operator). For example, ground truth information could be used to locate "interesting" regions of feature space that could then be painted with various colors. A whole new approach to the digital processing of multispectral scanner data will be concerned with how best to have the computer "turn the knobs" in order to produce meaningful motion picture classification maps of various levels of detail.

The real time interactive system should be built in order to test the human reaction features of the system. It is expected that this system will effectively put the human brain into the data processing and pattern recognition loop. Since the operator views 250,000 pixels at a glance, he will be able to use the spatial information that is apparent to him to guide his way through the spectral feature space. After studying how the human operator reacts to this system an effort should be made to train the computer to "turn the knobs" and thus produce its own motion picture classification maps based on ground truth or other adaptive learning information.

References

1. R. E. Haskell, "CLUSTD-A New Program for the Nonsupervised Classification of Multispectral Data," NASA Tech. Rept. JSC-09010, April 1973.

2. R. E. Haskell, "A New Optical Process for Producing Classification Maps from Multispectral Data," Oakland University, School of Engineering, Tech. Rept. No. 73-1, September 1973.

3. R. E. Haskell, "Processing Multispectral Scanner Data Using Correlation Clustering and Nonparametric Classification Techniques," Technical Report, NASA Contract NAS-9-14194. School of Engineering, Oakland University, Rochester, Michigan.

4. R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis, John Wiley and Sons, New York 1973.

5. J. T. Tou and R. C. Gonzalez, Pattern Recognition Principles, Addison-Wesley, Reading, Mass., 1975.

6. W. S. Meisel, Computer-Oriented Approaches to Pattern Recognition, Academic Press, New York 1972.

7. E. Parzen, "On Estimation of a Probability Density Function and Mode," Ann. Math. Statist., 33, 1065-1076, 1962.

8. T. M. Cover and P. E. Hart, "Nearest Neighbor Pattern Classification," IEEE Trans. Info. Th. IT-13, 21-27, 1967.

9. W. H. Highleyman, "Linear Decision Functions, with Application to Pattern Recognition," Proc. IRE, 50, 1501-1514, 1962.

10. J. R. Ullmann, Pattern Recognition Techniques, Crane, Russak & Co., Inc. New York 1973.

11. A. G. Arkader and E. M. Braverman, Computers and Pattern Recognition, Chap. 4, Thompson Book Co., Inc., Washington, D. C., 1967.

12. M. A. Aizerman, E. M. Braverman, L. I. Rozonoer, "Theoretical Foundation of the Potential Function Method in Pattern Recognition Learning," Automation and Remote Control 25, 821-839, 1964.

13. M. A. Aizerman, E. M. Braverman, and L. I. Rozonoer, "The Probability Problem of Pattern Recognition Learning and the Method of Potential Functions," Automation and Remote Control 25, 1175-1190, 1964.

14. M. A. Aizerman, E. M. Braverman, and L. I. Rozonoer, "The Method of Potential Functions for the Problem of Restoring the Characteristic of a Function Converter from Randomly Observed Points," Automation and Remote Control 25, 1705-1714, 1964.

15. O. A. Bashkirov, E. M. Braverman, I. B. Muchnik, "Potential Function Algorithms for Pattern Recognition Learning Machines," Automation and Remote Control 25, 629-631, 1964.

16. R. R. Lemke and K. S. Fu, "Application of the Potential Function Method to Pattern Classification," Proc. of the NEC, XXVI, 107-112, 1968.

17. D. E. Boddy and R. E. Haskell, "A Nonparametric Hierarchical Classifier Based on an Iterative Method of Potentials," Technical Report, NASA Contract NAS-9-14195, School of Engineering, Oakland University, Rochester, Michigan, June 1975.

18. R. E. Haskell, "An Interactive Color Display for Multispectral Imagery Using Correlation Clustering," Technical Report, NASA Contract NAS-9-14195, School of Engineering, Oakland University, Rochester, Michigan, May 1975.

ORIGINAL PAGE IS OF POOR QUALITY

APPENDIX A

ALGOL Listing of CLUSTH Including Procedures

- HEAD2
- INPUTH
- CALCULATE
    - DATA3
    - ASSIGNH
    - CTAPEHEAD

```
%                           CLUSTH     [REVISED VERSION]
%
            BEGIN
FILE IN          ERTS 2 (2,500);
FILE IN          CARD DISK(2,10,30);
FILE OUT         LINE PRINT (2,17);
FILE OUT         CTAPE 2(2,900,SAVE 99);
%                IN DISKFILE  CTAPET:
%                THERE ARE 150 WRDS/RECD AND 300 WRDS/BLK
%                MAXIMUM LOGICAL RECORDS = 4 TIMES 200 = 800
FILE OUT         CTAPET DISK [4:200] (2,150,300);

%           ****************************************
%              GLOSSARY OF GLOBAL VARIABLES
%           ****************************************

REAL                CMIN;         %    THE CORRELATION THRESHOLD.
%                                      IF A CORRELATION C IS COMPUTED FOR A
%                                      GIVEN CLUSTER, AND IF C GEQ CMIN, THEN
%                                      THE PIXEL IS ASSIGNED TO THAT CLUSTER.
INTEGER ARRAY       CT1[0:50];    %    AN ARRAY CONTAINING 51 WORDS FOR STORING
%                                      HEADER INFORMATION AS DEFINED IN FIG. 5.
INTEGER             ENDREC;       %    THE FINAL SCAN LINE OR RECORD NUMBER
%                                      TO BE READ FROM THE INPUT TAPE.
INTEGER             ENDSAMP;      %    THE FINAL SAMPLE NUMBER OR PIXEL
%                                      NUMBER TO BE PROCESSED IN EACH SCAN LINE.
INTEGER             INCR;         %    THE INCREMENT USED IN READING SCAN LINES
%                                      FROM THE INPUT TAPE.  IF INCR=1, ALL
%                                      SCAN LINES ARE READ.  IF INCR=2, EVERY
%                                      OTHER SCAN LINE IS READ, ETC.
INTEGER             INCS;         %    THE INCREMENT USED IN PROCESSING SAMPLE
%                                      NUMBERS IN EACH SCAN LINE.  IF INCS=1,
%                                      EVERY PIXEL IN THE SCAN LINE IS PROCESSED.
%                                      IF INCS=2, EVERY OTHER PIXEL IS
%                                      PROCESSED, ETC.
INTEGER             MAXCLUST;     %    THE MAXIMUM NUMBER OF CLUSTERS ALLOWED.
%                                      IF THE PROGRAM TRIES TO CREATE MORE
%                                      THAN MAXCLUST CLUSTERS THE PIXEL IS
%                                      ASSIGNED TO AN "OTHER" CATEGORY BY
%                                      SETTING NS=0.
INTEGER             MR;           %    THE NUMBER OF RECORDS TO BE SKIPPED IN
%                                      ORDER TO READ SCAN LINE NUMBER NRECD.
INTEGER             NBACK;        %    THE NUMBER OF CLUSTERS FOR WHICH A
%                                      CORRELATION IS COMPUTED BEFORE A NEW
%                                      CLUSTER IS CREATED.  IF NBACK EXCEEDS
%                                      THE CURRENT NUMBER OF CLUSTERS, THEN
%                                      ALL CLUSTERS ARE CHECKED.
INTEGER             NCHAN;        %    NUMBER OF SPECTRAL CHANNELS ON TAPE.
INTEGER             NCLUST;       %    THE NUMBER OF CLUSTERS THAT HAVE
%                                      BEEN CREATED.
INTEGER             NROLD;        %    THE LAST SCAN LINE TO HAVE BEEN READ.
INTEGER             NS;           %    CLUSTER NUMBER.
INTEGER             NSAMP;        %    NUMBER OF SAMPLES (PIXELS) IN EACH
%                                      SCAN LINE ON THE TAPE.
INTEGER             NWRD48;       %    THE NUMBER OF 48-BIT WORDS NEEDED TO
%                                      STORE THE DATA IN ONE SCAN LINE.
%                                      THIS VALUE IS COMPUTED IN HEAD2 AND
%                                      USED TO ESTABLISH THE DIMENSION OF THE
%                                      ARRAY IDAT IN CALCULATE.
REAL ARRAY          SIG[0:12,0:200];
%                                           SIG[J,NS] IS A TWO-DIMENSIONAL ARRAY
%                                      CONTAINING THE AVERAGE SPECTRAL SIGNATURES
%                                      ASSOCIATED WITH EACH CLUSTER.
%                                      THE ROWS OF SIG CORRESPOND TO THE
%                                      SPECTRAL CHANNELS AND THE COLUMNS OF SIG
%                                      CORRESPOND TO THE CLUSTER NUMBER.
INTEGER             STARTREC;     %    THE INITIAL SCAN LINE OR RECORD NUMBER
%                                      TO BE READ FROM THE INPUT DATA TAPE.
INTEGER             STARTSAMP;    %    THE INITIAL SAMPLE NUMBER OR PIXEL
%                                      NUMBER TO BE PROCESSED IN EACH SCAN LINE.
REAL ARRAY          WIDTH[0:24];%      THE PARAMETERS THAT CONTROL THE WEIGHTING
%                                      FUNCTIONS IN EACH CHANNEL AS DEFINED
%                                      IN FIG. 3.

LABEL               ENDING;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
%==============================================================

%     PROCEDURES HEAD2, INPUTH, AND CALCULATE ARE INSERTED HERE
% *** *** ***   MAIN PROGRAM   *** *** ***

%          THE PROGRAM CLUSTH READS A DATA TAPE CONTAINING
%          MULTISPECTRAL SCANNER DATA AND PRODUCES A CLUSTER TAPE
%          CALLED CTAPE ON WHICH EACH PIXEL HAS BEEN ASSIGNED TO
%          ONE OF MAXCLUST CLUSTERS.  EACH CLUSTER CONTAINS PIXELS
%          WITH SIMILAR SPECTRAL SIGNATURES.
          +-------------------------------------+
          |     HEAD2(ERTS,NCHAN,NSAMP);        |
          +-------------------------------------+
          |     INPUTH;                         |
          +-------------------------------------+
          |     CALCULATE;                      |
          +-------------------------------------+
ENDING:   | END OF PROGRAM.                     |
          +-------------------------------------+
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
PROCEDURE       HEAD2(FILENAME,NCHAN,NSAMP);

%           THE PROCEDURE HEAD2 READS THE HEADER OF A DATA TAPE THAT
%           HAS BEEN WRITTEN IN LARSYS-II FORMAT.  FROM THIS HEADER
%           IT DETERMINES THE NUMBER OF SPECTRAL CHANNELS NCHAN, AND
%           THE NUMBER OF PIXELS, NSAMP, IN A SINGLE SCAN LINE.
%           IT ALSO COMPUTES NWRD48.

FILE            FILENAME;   %  THE FILE IDENTIFIER FOR THE DATA
%                              TAPE WHOSE HEADER IS BEING READ.
INTEGER         NCHAN;      %  NUMBER OF SPECTRAL CHANNELS ON TAPE.
INTEGER         NSAMP;      %  NUMBER OF SAMPLES (PIXELS) IN EACH
          BEGIN

INTEGER ARRAY   ID[0:200]; %  AN ARRAY IN WHICH THE UNPACKED 32-BIT
%                              WORDS OF THE LARSYS-II HEADER ARE STORED.
INTEGER ARRAY   IHED[0:134];% AN ARRAY INTO WHICH THE DATA FROM
%                              THE HEADER IS READ AS UNPROCESSED
%                              FULL 48-BIT WORDS.
INTEGER         K;          %  INDEXING VARIABLE

FORMAT          FMT0(X20,"HEADER RECORD OF LARSYS-II TAPE"/),
                FMT4(X10,"TAPE NUMBER",I6,X10,"DATA IS CONTINUATION"
                  " OF FLIGHT LINE STARTED ON TAPE",I6),
                FMT5(X10,"TAPE NUMBER",I6),
                FMT6(X10,"FILE NUMBER=",I6,X10,"RUN NUMBER=",I8/X10,
                  "NUMBER OF CHANNELS=",I3,X6,"NUMBER OF SAMPLES",
                  " PER LINE=",I4),
                FMT7(X10,"DATE",X3,3I6/X10,"ALTITUDE=",I10,X9,
                  "GROUND HEADING=",I6);
%
%
        ┌──────────────────────────────────────────────────────────┐
        │                 MAIN BODY OF HEAD2                       │
        ├──────────────────────────────────────────────────────────┤
        │    READ (FILENAME,134,IHED[*]);                          │
        ├──────────────────────────────────────────────────────────┤
        │    FOR K:=0 STEP 1 UNTIL 20 DO                           │
        │  ┌───────────────────────────────────────────────────────┤
        │  │   REPLACE POINTER(ID[*],8)+8+6×K                      │
        │  │        BY POINTER(IHED[*],8)+K×4 FOR 4;               │
        ├──┴───────────────────────────────────────────────────────┤
        │    NCHAN:=ID[5];                                         │
        │    NSAMP:=ID[6];                                         │
        │    NWRD48:=(NCHAN×NSAMP+4) DIV 6 + 1;                    │
        ├──────────────────────────────────────────────────────────┤
        │    WRITE(LINE,FMT0);                                     │
        ├──────────────────────────────────────────────────────────┤
        │    IF ID[4] EQL 0                                        │
        ├ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┬ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─ ┤
        │                           │ THEN                         │
        │                           ├──────────────────────────────┤
        │                           │  WRITE(LINE,FMT5,ID[1],ID[4])│
        │    ELSE                   │                              │
        ├───────────────────────────┴─────┐                        │
        │   WRITE(LINE,FMT4,ID[1],ID[4]); │                        │
        ├─────────────────────────────────┴────────────────────────┤
        │    WRITE (LINE,FMT6,ID[2],ID[3],ID[5],ID[6]);            │
        │    WRITE (LINE,FMT7,ID[11],ID[12],ID[13],ID[15],ID[16]); │
        │    WRITE (LINE[PAGE]);                                   │
        ├──────────────────────────────────────────────────────────┤
        │END OF HEAD2;                                             │
        │                                                          │
        └──────────────────────────────────────────────────────────┘
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
%=================================================

PROCEDURE       INPUTH;

%          THE PROCEDURE INPUTH READS VARIOUS INPUT PARAMETERS
%          FROM SIX DATA CARDS.  ALL VARIABLES ARE GLOBAL.

           BEGIN
INTEGER        I;           %  INDEXING VARIABLE.
INTEGER        J1;          %  INDEXING VARIABLE.

FORMAT IN      FIN1 (10I5);
FORMAT IN      FIN2 (10F6.2);
FORMAT OUT     FOUT1 ("FOR THIS RUN"//"NBACK = ",I5/
                 "MAXCLUST = ",I5," NCHAN = ",I5/
                 "STARTREC = ",I5,"  ENDREC = ",I5,"  INCR = ",
                 I5/"STARTSAMP = ",I5,"  ENDSAMP = ",I5,
                 "   INCS = ",I5//);
FORMAT OUT     FOUT2 ("THE ALLOWED DELTA FOR EACH CHANNEL IS "
                   "AS FOLLOWS"/"CHAN",X5,"DELTA"/);
FORMAT OUT     FOUT3 (I3,X5,F6.2);
FORMAT OUT     FOUT4("CMIN=",F6.1);
FORMAT OUT     FOUT5("DATA WILL BE WRITTEN ON FILE NUMBER",
                    I5,"ON THE CLUSTER TAPE CTAPE");
FORMAT OUT     SAMPERR ("   *** ERROR ***",X10,"ENDSAMP IS "
                   "TOO LARGE."/"ENDSAMP WILL BE READJUSTED TO "
                   "EQUAL NSAMP");
%              *********************
%     +----------------------------------------------------------+
      |                  MAIN BODY OF INPUTH                     |
      +----------------------------------------------------------+
      |        NCLUST:=0;                                        |
      |        NROLD:=0;                                         |
      +----------------------------------------------------------+
      |        READ(CARD,<2A6>,CT1[0],CT1[1]);                   |
      |        READ (CARD,FIN1,NBACK,MAXCLUST);                  |
      |        READ (CARD,FIN2,FOR J1:=1 STEP 1 UNTIL NCHAN      |
      |               DO WIDTH[J1]);                             |
      |        READ(CARD,FIN2,CMIN);                             |
      |        READ (CARD,FIN1,STARTREC,ENDREC,INCR);            |
      |        READ (CARD,FIN1,STARTSAMP,ENDSAMP,INCS);          |
      +----------------------------------------------------------+
      |        WRITE (LINE,FOUT1,NBACK,MAXCLUST,NCHAN,           |
      |              STARTREC,ENDREC,INCR,STARTSAMP,ENDSAMP,INCS);
      |        WRITE(LINE,FOUT4,CMIN);                           |
      +----------------------------------------------------------+
      |        IF ENDSAMP GTR NSAMP                              |
      |- - - - - - - - - - - - - - - - -+- - - - - - - - - - - - |
      |                                 | THEN                   |
      |                                 +------------------------+
      |                                 |   BEGIN                |
      |                                 |     WRITE(LINE,SAMPERR);
      |                                 |     ENDSAMP:=NSAMP;    |
      |                                 |   END                  |
      |        ELSE;                    |                        |
      +----------------------------------------------------------+
      |        WRITE (LINE,FOUT2);                               |
      |        WRITE (LINE,FOUT3,FOR J1:=1 STEP 1 UNTIL NCHAN    |
      |             DO [J1,WIDTH[J1]]);                          |
      +----------------------------------------------------------+
      |   END OF INPUTH;                                         |
      +----------------------------------------------------------+
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
PROCEDURE       CALCULATE;
%               THE PROCEDURE CALCULATE READS THE INPUT DATA TAPE A SCAN
%               LINE AT A TIME USING THE PROCEDURE DATA3.  IT THEN ASSIGNS
%               EACH PIXEL IN A GIVEN SCAN LINE TO A PARTICULAR CLUSTER
%               BY USING THE PROCEDURE ASSIGNH.

                BEGIN

%               ****************************************
%                  GLOSSARY OF VARIABLES LOCAL TO CALCULATE
%                  AND GLOBAL TO DATA3, ASSIGNH, AND CTAPEHEAD.
%               ****************************************

INTEGER ARRAY   IDAT[0:NWRD48]; %  AN ARRAY INTO WHICH THE DATA FROM
%                                  A SINGLE SCAN LINE IS READ AS
%                                  UNPROCESSED FULL 48-BIT WORDS.
INTEGER ARRAY   IDUM[0:NCHAN,0:NSAMP];% A TWO DIMENSIONAL ARRAY INTO
%                              WHICH THE UNPACKED 8-BIT BYTES
%                              REPRESENTING THE SPECTRAL SIGNATURE
%                              OF EACH PIXEL IN A SCAN LINE ARE STORED.
%                              THE ROWS CORRESPOND TO THE CHANNEL
%                              NUMBERS AND THE COLUMNS CORRESPOND
%                              TO THE SAMPLE OR PIXEL NUMBERS.
INTEGER         IRECNO;     %  THE SCAN LINE OR RECORD NUMBER AS READ
%                              FROM THE INPUT DATA TAPE.
INTEGER         JJ;         %  AN INDEX VARIABLE.
INTEGER         KSAMP;      %  AN INDEX VARIABLE CORRESPONDING TO A
%                              SAMPLE OR PIXEL NUMBER.
INTEGER         KNT;        %  AN INDEX VARIABLE CORRESPONDING TO THE
%                              PIXEL NUMBER ASSOCIATED WITH THE CLUSTER
%                              NUMBERS THAT ARE WRITTEN ON THE
%                              OUTPUT CLUSTER TAPE CTAPE.
INTEGER         NPIXELS;    %  THE NUMBER OF PIXELS IN EACH SCAN LINE
%                              THAT HAVE BEEN ASSIGNED TO CLUSTERS.
INTEGER ARRAY   NSS[0:NSAMP];%  AN ARRAY FOR STORING THE CLUSTER
%                              NUMBERS OF EACH PIXEL IN A SCAN LINE.
INTEGER         NRECD;      %  AN INDEX VARIABLE CORRESPONDING TO A
%                              SCAN LINE OR RECORD NUMBER.
INTEGER         NSCANLINE;  %  THE NUMBER OF SCAN LINES THAT
%                              HAVE BEEN PROCESSED.
INTEGER ARRAY   NUM[0:MAXCLUST];%  AN ARRAY FOR STORING THE NUMBER
%                              OF PIXELS THAT HAVE BEEN ASSIGNED
%                              TO EACH CLUSTER.

REAL ARRAY      SIG[0:NCHAN,0:MAXCLUST] ;
FORMAT OUT      RECDERR ("**  ERROR  **",X5,"NRECD=",I5,X5,
                   "IRECNO=",I5);
FORMAT OUT      NEATLY (20I4);

%       ----------------------------------------

%       PROCEDURES DATA3, ASSIGNH, AND CTAPEHEAD ARE INSERTED HERE
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
%==========================================================
* ***  ***  ***   MAIN BODY OF CALCULATE   ***  ***  ***
*                 NOTE:  ON THE B5500, ALL VARIABLES
*                        ARE AUTOMATICALLY SET TO ZERO
*                        BEFORE EXECUTION.  IF THIS
*                        WAS NOT AUTOMATIC, THE ARRAYS
*                        NUM AND SIG WOULD BE SET
*                        TO ZERO HERE.

                  NPIXELS:=(ENDSAMP-STARTSAMP)/INCS+1;
                  NSCANLINE:=(ENDREC-STARTREC)/INCR+1;

                  FOR NRECD:=STARTREC STEP INCR UNTIL ENDREC
                  DO BEGIN
                     KNT:=0;
                     DATA3 (ERTS,NRECD,IRECNO);
                     IF NRECD NEQ IRECNO
                                 THEN
                                    BEGIN
                                     WRITE(LINE,RECDERR,NRECD,IRECNO);
                                     GO TO ENDING;
                                    END
                     ELSE;
                     FOR KSAMP:=STARTSAMP STEP INCS UNTIL ENDSAMP
                     DO BEGIN

                        ASSIGNH;

                        NUM[NS]:=NUM[NS]+1;
                        NSS[KNT]:=NS;
                        KNT:=KNT+1;
                        END;
                     WRITE(LINE,NEATLY,FOR JJ:=0 STEP 1
                           UNTIL NPIXELS-1 DO NSS[JJ]);
                     WRITE(CTAPET,NPIXELS,NSS[*]);
                     END;
                  REWIND(CTAPET);
                  LOCK(ERTS);
                  CTAPEHEAD;
              END OF CALCULATE;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
PROCEDURE      DATA3(FILENAME,NRECD,IRECNO);

%              THE PROCEDURE DATA3 READS SCAN LINE NUMBER NRECD FROM AN
%              INPUT DATA TAPE WITH FILE IDENTIFIER FILENAME.  THE FIRST
%              16 BITS OF EACH SCAN LINE CONTAINS THE RECORD NUMBER
%              IRECNO.  THIS VALUE IS PASSED BACK TO THE MAIN BODY OF
%              CALCULATE WHERE IT IS COMPARED (AND SHOULD AGREE) WITH
%              THE VALUE OF NRECD.  THE SECOND 16 BITS OF EACH SCAN LINE
%              CONTAINS THE ROLL PARAMETER.  THIS VALUE IS NOT USED AND
%              IS SIMPLY STORED IN THE ARRAY WORK[1].  THE REMAINING DATA
%              IN EACH SCAN LINE CONSISTS OF 8-BIT BYTES REPRESENTING THE
%              SPECTRAL SIGNATURES OF EACH PIXEL IN THE SCAN LINE.  THIS
%              DATA IS UNPACKED AND STORED IN THE 2-DIMENSIONAL ARRAY IDUM.

FILE               FILENAME;    %   THE FILE IDENTIFIER FOR THE DATA TAPE.
INTEGER            IRECNO;      %   THE SCAN LINE OR RECORD NUMBER AS
%                                   READ FROM THE INPUT DATA TAPE.
INTEGER            NRECD;       %   AN INDEX VARIABLE CORRESPONDING TO
%                                   A SCAN LINE OR RECORD NUMBER.

               BEGIN

INTEGER            K;           %   AN INDEX VARIABLE.
INTEGER            K1;          %   AN INDEX VARIABLE.
INTEGER            TEMP;        %   A TEMPORARY STORAGE VARIABLE.
INTEGER ARRAY      WORK[0:1];   %   A TEMPORARY STORAGE AREA.

%                  ***********************

          +--------------------------------------------------------------+
%         |                     MAIN BODY OF DATA3                       |
          +--------------------------------------------------------------+
          |        MR:=NRECD-NROLD-1;                                    |
          |        SPACE (FILENAME,MR);                                  |
          +--------------------------------------------------------------+
          |        READ (FILENAME,NWRD48,IDAT[*]);                       |
          +--------------------------------------------------------------+
          |        FOR K:=0 STEP 1 UNTIL 1                               |
          |    +---------------------------------------------------------+
          |    |   DO REPLACE POINTER(WORK[*],8)+4+6×K                   |
          |    |      BY POINTER(IDAT[*],8)+2×K FOR 2;                   |
          +--------------------------------------------------------------+
          |        IRECNO:=WORK[0];                                      |
%         |        IROLLP:=WORK[1];        ROLL PARAMETER NOT USED       |
          +--------------------------------------------------------------+
          |        FOR K:=0 STEP 1 UNTIL NCHAN-1                         |
          |    +---------------------------------------------------------+
          |    |   DO BEGIN                                              |
          |    |      TEMP:= (NSAMP TIMES K) + 4;                        |
          |    +---------------------------------------------------------+
          |    |      FOR K1:=0 STEP 1 UNTIL NSAMP-1                     |
          |    |   +-----------------------------------------------------+
          |    |   |  DO REPLACE POINTER(IDUM[K,*],8)+5+6×K1             |
          |    |   |     BY POINTER(IDAT[*],8)+TEMP+K1 FOR 1;            |
          |    +---------------------------------------------------------+
          |    |      END;                                               |
          +--------------------------------------------------------------+
          |        NROLD:=NRECD;                                         |
          +--------------------------------------------------------------+
          |  END OF DATA3;                                               |
          +--------------------------------------------------------------+
%     -------------------------------------------
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
PROCEDURE       ASSIGNH;
%             THE PROCEDURE ASSIGNH ASSIGNS A PIXEL TO CLUSTER NUMBER NS,
%             USING THE CORRELATION CLUSTERING ALGORITHM DESCRIBED IN
%             SECTION 3 OF THIS REPORT.  IT USES A WEIGHTING FUNCTION
%             OF THE TYPE SHOWN IN FIG. 3 AND COMPUTES THE CORRELATION
%             FUNCTION FOR, AT MOST, THE NBACK MOST RECENT CLUSTERS.

              BEGIN
REAL                C;          %  THE CORRELATION FUNCTION
INTEGER             J;          %  AN INDEX VARIABLE.

LABEL               AWAY;
%                   **********************

%                        MAIN BODY OF ASSIGNH

                    FOR NS:=NCLUST STEP -1 UNTIL
                                       IF NCLUST-NBACK LEQ 0 THEN 1
                                       ELSE NCLUST-NBACK
                    DO BEGIN
                       C:=0;
                       FOR J:=1 STEP 1 UNTIL NCHAN DO
                          C:=C+WIDTH[J]-ABS(IDUM[J-1,KSAMP-1]-SIG[J,NS]);
                       IF C GEQ CMIN
                                 THEN
                                   BEGIN
                                     FOR J:=1 STEP 1 UNTIL NCHAN DO
                                       SIG[J,NS]:=NUM[NS]/(NUM[NS]+1)×SIG[J,NS]
                                            +IDUM[J-1,KSAMP-1]/(NUM[NS]+1);
                                     GO TO AWAY;
                                   END
                       ELSE
                       END;
                       IF NCLUST LSS MAXCLUST
                                       THEN
                                         BEGIN
                                           NCLUST:=NCLUST+1;
                                           NS:=NCLUST;
                                           FOR J:=1 STEP 1 UNTIL NCHAN DO
                                             SIG[J,NS]:=IDUM[J-1,KSAMP-1];
                                         END
                       ELSE
                              NS:=0;
AWAY:         END OF ASSIGNH;
%        -----------------------------------------
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
PROCEDURE       CTAPEHEAD;
%               THE PROCEDURE CTAPEHEAD IS USED TO WRITE THE FIRST THREE
%               RECORDS ON THE OUTPUT CLUSTER TAPE CTAPE.  IT THEN COPIES
%               THE CLUSTER NUMBERS NSS[*] THAT HAVE BEEN ASSIGNED TO EACH
%               PIXEL IN CALCULATE FROM A DISK FILE ONTO THE OUTPUT TAPE.
%
                BEGIN
INTEGER            I;          %  AN INDEX VARIABLE.
%                      MAIN BODY OF CTAPEHEAD
                   CT1[2]:=NCHAN;
                   CT1[3]:=NSAMP;
                   CT1[4]:=STARTREC;
                   CT1[5]:=ENDREC;
                   CT1[6]:=INCR;
                   CT1[7]:=STARTSAMP;
                   CT1[8]:=ENDSAMP;
                   CT1[9]:=INCS;
                   CT1[10]:=NCLUST;
                   CT1[11]:=NBACK;
                   CT1[40]:=MAXCLUST;
                   CT1[41]:=CMIN;
                   FOR I:=12 STEP 1 UNTIL 11+NCHAN DO
                   CT1[I]:=WIDTH[I-11];
                   WRITE(CTAPE,51,CT1[*]);
                   FOR I:=1 STEP 1 UNTIL NCHAN DO
                       WRITE(CTAPE,NCLUST+1,SIG[I,*]);
                   WRITE(CTAPE,NCLUST+1,NUM[*]);
                   FOR NRECD:=1 STEP 1 UNTIL NSCANLINE DO
                   BEGIN
                       READ(CTAPET,NPIXELS,NSS[*]);
                       WRITE(CTAPE,NPIXELS,NSS[*]);
                   END;
                   REWIND(CTAPE);
                END OF CTAPEHEAD;
%==========================================================
```

ORIGINAL PAGE IS
OF POOR QUALITY

APPENDIX B

ALGOL Listing of GROUPL*
Including Procedures

```
HEADIN
GROUPXMAIN
   HEADOUT
   GROUNDTRUTH
   COSTMATRIX
   POTENTIAL
     PTRAIN
     PTEST
   TAPEOUTPUT
   TRUTHMAP
     SAMPNOS
     PLOT
   TEST
```

*GROUPL is a verson of GROUPX that uses the procedure POTENTIAL for classification.

```
%
%                            PRODUCES OUTPUT CLASSIFICATION TAPE
%                            FROM INPUT CLUSTER TAPE
%
%
%
           BEGIN
FILE IN             CTAPE 2(2,900);
FILE IN             CARDS DISK(2,10,30);
FILE OUT            OUTAPE 2(2,500,SAVE 99));
FILE OUT            LINE PRINT(2,17);
FILE OUT            DISC DISK[1:10] (2,15,30,SAVE=99);

%               ******************************
%          GLOSSARY OF GLOBAL VARIABLES
%               ******************************

REAL                CMIN;          %   THE CORRELATION THRESHOLD.
%                                      IF A CORRELATION C IS COMPUTED FOR A
%                                      GIVEN CLUSTER, AND IF C GEQ CMIN, THEN
%                                      THE PIXEL IS ASSIGNED TO THAT CLUSTER.
INTEGER ARRAY       CT1[0:50];     %   AN ARRAY CONTAINING 51 WORDS FOR STORING
%                                      HEADER INFORMATION AS DEFINED IN FIG. 5.
REAL ARRAY          SIG[0:12,0:200];   %   SIG[J,NS] IS A TWO-DIMENSIONAL
%                                      ARRAY CONTAINING THE AVERAGE SPECTRAL
%                                      SIGNATURES ASSOCIATED WITH EACH CLUSTER.
%                                      THE ROWS OF SIG CORRESPOND TO THE
%                                      SPECTRAL CHANNELS AND THE COLUMNS OF
%                                      SIG CORRESPOND TO THE CLUSTER NUMBER.
INTEGER             INCR;          %   THE INCREMENT USED IN READING SCAN LINES
%                                      FROM THE INPUT TAPE.  IF INCR=1, ALL
%                                      SCAN LINES ARE READ.  IF INCR=2, EVERY
%                                      OTHER SCAN LINE IS READ, ETC.
INTEGER             INCS;          %   THE INCREMENT USED IN PROCESSING SAMPLE
%                                      NUMBERS IN EACH SCAN LINE.  IF INCS=1,
%                                      EVERY PIXEL IN THE SCAN LINE IS PROCESSED.
%                                      IF INCS=2, EVERY OTHER PIXEL IS
%                                      PROCESSED, ETC.
INTEGER             J;             %   AN INDEX VARIABLE.
INTEGER             K1;            %   THE INITIAL SAMPLE NUMBER OR PIXEL
%                                      NUMBER TO BE PROCESSED IN EACH SCAN LINE.
INTEGER             K2;            %   THE FINAL SAMPLE NUMBER OR PIXEL
%                                      NUMBER TO BE PROCESSED IN EACH SCAN LINE.
INTEGER             MATOT;         %   THE NUMBER OF CLASSES FOR WHICH
%                                      GROUND TRUTH IS BEING USED.
INTEGER             MAXCLUST;      %   THE MAXIMUM NUMBER OF CLUSTERS ALLOWED.
%                                      IF THE PROGRAM TRIES TO CREATE MORE
%                                      THAN MAXCLUST CLUSTERS THE PIXEL IS
%                                      ASSIGNED TO AN "OTHER" CATEGORY BY
%                                      SETTING NS=0.
INTEGER             NBACK;         %   THE NUMBER OF CLUSTERS FOR WHICH A
%                                      CORRELATION IS COMPUTED BEFORE A NEW
%                                      CLUSTER IS CREATED.  IF NBACK EXCEEDS
%                                      THE CURRENT NUMBER OF CLUSTERS, THEN
%                                      ALL CLUSTERS ARE CHECKED.
INTEGER             NCHAN;         %   NUMBER OF SPECTRAL CHANNELS ON TAPE.
INTEGER             NCLUST;        %   THE NUMBER OF CLUSTERS THAT HAVE
%                                      BEEN CREATED.
INTEGER             NPIXEND;       %   THE NUMBER OF PIXELS PER SCAN LINE
%                                      ON CTAPE.
INTEGER             NR1;           %   THE INITIAL SCAN LINE OR RECORD NUMBER
%                                      TO BE READ FROM THE INPUT DATA TAPE.
INTEGER             NR2;           %   THE FINAL SCAN LINE OR RECORD NUMBER
%                                      TO BE READ FROM THE INPUT TAPE.
INTEGER             NRECEND;       %   THE NUMBER OF SCAN LINES ON CTAPE.
INTEGER             NROLD;         %   THE LAST RECORD NUMBER TO HAVE BEEN
%                                      READ FROM CTAPE.
INTEGER             NS;            %   CLUSTER NUMBER.
INTEGER             NSAMP;         %   NUMBER OF SAMPLES (PIXELS) IN EACH
%                                      SCAN LINE ON THE TAPE.


%              *************************
%         PROCEDURES HEADIN AND GROUPXMAIN ARE INSERTED HERE
%              *************************


%                    MAIN PROGRAM

                HEADIN;

                GROUPXMAIN;

           END OF GROUPL.
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
PROCEDURE HEADIN;

%          THE PROCEDURE HEADIN IS USED TO READ THE FIRST TWO
%          RECORDS FROM THE INPUT TAPE CTAPE.  THE INFORMATION
%          CONTAINED IN THE FIRST RECORD OF CTAPE IS USED FOR
%          DIMENSIONING ARRAYS IN GROUPXMAIN.

           BEGIN
%          ************************

%                 READ FIRST TWO RECORDS OF CLUSTER TAPE

              READ(CTAPE,51,CT1[*]);

              NCHAN:=CT1[2];
              NSAMP:=CT1[3];
              NR1:=CT1[4];
              NR2:=CT1[5];
              INCR:=CT1[6];
              K1:=CT1[7];
              K2:=CT1[8];
              INCS:=CT1[9];
              NCLUST:=CT1[10];
              NBACK:=CT1[11];
              MAXCLUST:=CT1[40];
              CMIN:=CT1[41];
              NROLD:=0;

              NRECEND:=(NR2-NR1)/INCR+1;
              NPIXEND:=(K2-K1)/INCS+1;

              FOR J:=1 STEP 1 UNTIL NCHAN DO
                 READ (CTAPE, NCLUST+1,SIG[J,*]);

              READ(CARDS,<I5>,MATOT);

           END OF HEADIN;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
PROCEDURE GROUPXMAIN;

%          THE PROCEDURE GROUPXMAIN PRODUCES THE OUTPUT
%          CLASSIFICATION TAPE OUTAPE FROM THE INPUT CLUSTER
%          TAPE CTAPE USING THE FOLLOWING STEPS:
%          1)  THE PROCEDURE GROUNDTRUTH USES GROUND TRUTH
%          INFORMATION IN CONJUNCTION WITH THE DATA ON THE
%          CLUSTER TAPE TO PRODUCE A COSTMATRIX MATN[NS,MAT] THAT
%          CONTAINS THE NUMBER OF PIXELS KNOWN TO BE OF CLASS
%          MAT THAT HAVE BEEN ASSIGNED TO CLUSTER NUMBER NS.

%          2)  THE PROCEDURE COSTMATRIX USES THE INFORMATION
%          IN THE COSTMATRIX MATN[NS,MAT] TO ESTIMATE THE A POSTERIORI
%          PROBABILITIES OF CLUSTER NS BELONGING TO CLASS MAT.
%          IF ENOUGH GROUND TRUTH DATA IS AVAILABLE AND THE
%          A POSTERIORI PROBABILITY IS SUFFICIENTLY HIGH, ALL OF
%          THE PIXELS IN CLUSTER NS ARE ASSIGNED TO CLASS MAT.

%          3)  THOSE CLUSTERS THAT WERE UNABLE TO BE CLASSIFIED
%          BY THE COSTMATRIX ARE CLASSIFIED USING THE METHOD OF
%          POTENTIALS BY THE PROCEDURE POTENTIAL.

%          4)  THE RESULTING CLASSIFICATION OF EACH PIXEL IS WRITTEN
%          ON THE OUTPUT TAPE OUTAPE USING THE PROCEDURE TAPEOUTPUT.

           BEGIN

%              ******************************

%         GLOSSARY OF VARIABLES LOCAL TO GROUPXMAIN
%         AND GLOBAL TO READOUT, GROUNDTRUTH, COSTMATRIX,
%         POTENTIAL,TAPEOUTPUT,TRUTHMAP,AND TEST.
%              ******************************

REAL               ALFA;        %  A PARAMETER USED IN THE DEFINITION OF

%                                  THE POTENTIAL FUNCTION.
BOOLEAN            BOL;         %  A BOOLEAN VARIABLE USED TO PRINT MAP
INTEGER ARRAY CLASS[0:NRECEND,0:NPIXEND-1]; % AN ARRAY CONTAINING THE
%                                  CLASS NUMBERS ASSIGNED TO EACH PIXEL
INTEGER ARRAY GRNDT[0:MATOT,0:50,0:4]; %AN ARRAY TO STORE GROUND TRUTH
%                                  INFORMATION FOR USE IN PROCEDURE TRUTHMAP
INTEGER            I;           %  AN INDEX VARIABLE.
INTEGER            J;           %  AN INDEX VARIABLE CORRESPONDING TO THE
%                                  CHANNEL NUMBER.
INTEGER ARRAY   KLASS[0:NPIXEND-1];%  AN ARRAY WRITTEN ON OUTAPE
%                                  CONTAINING THE CLASS NUMBERS THAT HAVE
%                                  BEEN ASSIGNED TO EACH PIXEL IN A
%                                  SCAN LINE.
REAL               LAMDA;       %  THE WEIGHTING FACTOR USED IN MODIFYING
%                                  THE POTENTIAL FUNCTION IN EACH ERROR
%                                  CORRECTING ITERATION.
INTEGER            LB;          %  THE BEGINNING SCAN LINE NUMBER (ON
%                                  ORIGINAL DATA TAPE) IN A GROUND TRUTH
%                                  AREA.
INTEGER            LE;          %  THE ENDING SCAN LINE NUMBER (ON
%                                  ORIGINAL DATA TAPE) IN A GROUND TRUTH
%                                  AREA.
INTEGER            MAT;         %  AN INDEX VARIABLE CORRESPONDING TO
%                                  THE CLASS NUMBER.
INTEGER ARRAY   MATN[0:NCLUST,0:MATOT];%  MATN[NS,MAT] IS AN ARRAY
%                                  CONTAINING THE NUMBER OF PIXELS KNOWN
%                                  TO BE OF CLASS MAT THAT HAVE BEEN
%                                  ASSIGNED TO CLUSTER NUMBER NS.
INTEGER            MINPER;      %  IF THE PERCENTAGE PERCLS[NS] IS LESS
%                                  THAN MINPER FOR ANY CLUSTER NS, THEN
%                                  CLASSIFY THIS CLUSTER WITH POTENTIAL.
INTEGER            MINTOT;      %  IF THE NUMBER OF PIXELS WITHIN A CLUSTER
%                                  FOR WHICH GROUND TRUTH EXISTS IS LESS
%                                  THAN MINTOT THEN CLASSIFY THIS
%                                  CLUSTER WITH POTENTIAL.
INTEGER            NR;          %  THE NUMBER OF RECORDS TO BE SKIPPED IN
%                                  ORDER TO READ SCAN LINE NUMBER NREC1.
INTEGER            NCARDS;      %  THE NUMBER OF DATA CARDS CONTAINING
%                                  GROUND TRUTH INFORMATION ABOUT A
%                                  GIVEN CLASS.
INTEGER ARRAY NCRDS[0:MATOT]; % AN ARRAY FOR STORING THE NUMBER OF
%                                  GROUND TRUTH CARDS FOR USE IN TRUTHMAP
INTEGER            NPIX1;       %  THE BEGINNING PIXEL NUMBER ON CTAPE
%                                  CORRESPONDING TO A GROUND TRUTH AREA.
INTEGER            NPIX;        %  AN INDEX VARIABLE CORRESPONDING TO
%                                  A PIXEL NUMBER.
INTEGER            NPIX2;       %  THE ENDING PIXEL NUMBER ON CTAPE
%                                  CORRESPONDING TO A GROUND TRUTH AREA.
INTEGER            NREC;        %  AN INDEX VARIABLE CORRESPONDING TO A
%                                  RECORD NUMBER.
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
INTEGER          NREC1;        %  THE BEGINNING RECORD NUMBER ON CTAPE
%                                 CORRESPONDING TO A GROUND TRUTH AREA.
INTEGER          NREC2;        %  THE ENDING RECORD NUMBER ON CTAPE
%                                 CORRESPONDING TO A GROUND TRUTH AREA.
INTEGER          NSB;          %  THE BEGINNING SAMPLE (OR PIXEL) NUMBER (ON
%                                 ORIGINAL DATA TAPE) IN A GROUND TRUTH
%                                 AREA.
INTEGER          NSE;          %  THE ENDING SAMPLE (OR PIXEL) NUMBER (ON
%                                 ORIGINAL DATA TAPE) IN A GROUND TRUTH
%                                 AREA.
INTEGER ARRAY    NUM[0:NCLUST];   %  AN ARRAY FOR STORING THE NUMBER
%                                 OF PIXELS THAT HAVE BEEN ASSIGNED
%                                 TO EACH CLUSTER.
ARRAY            PERCLS[0:NCLUST];%  PERCLS[NS] IS THE MAXIMUM
%                                 PERCENTAGE OF PIXELS IN CLUSTER NS
%                                 THAT BELONG TO ONE CLASS.
INTEGER ARRAY    PIXELS[0:NPIXEND-1];%  AN ARRAY READ FROM CTAPE
%                                 CONTAINING THE CLUSTER NUMBERS ASSOCIATED
%                                 WITH EACH PIXEL IN A SCAN LINE.
INTEGER          PIXTOT;       %  THE TOTAL NUMBER OF PIXELS PROCESSED.
INTEGER ARRAY    TRANSP[0:NCLUST];%  TRANSP[NS] IS AN ARRAY CONTAINING
%                                 THE CLASS NUMBER MAT TO WHICH CLUSTER
%                                 NUMBER NS HAS BEEN ASSIGNED.

    -----------------------------------------

      PROCEDURES HEADOUT, GROUNDTRUTH, COSTMATRIX,POTENTIAL,
      TAPEOUTPUT,TRUTHMAP,AND TEST ARE INSERTED HERE.

%           MAIN BODY OF GROUPXMAIN

               HEADOUT;

               GROUNDTRUTH;

               READ(CARDS,<I5>,MINTOT);
               READ(CARDS,<I5>,MINPER);
               READ(CARDS,<2F10.4>,LAMDA,ALFA);

               WRITE(LINE,<X5,"MINTOT=",I6>,MINTOT);
               WRITE(LINE,<X5,"MINPER=",I6>,MINPER);
               WRITE(LINE,<X5,"LAMDA= ",F7.2>,LAMDA);
               WRITE(LINE,<X5,"ALFA= ",F7.2>,ALFA);

               COSTMATRIX;

               POTENTIAL;

%              TAPEOUTPUT;

               BOL:=TRUE;
               TRUTHMAP(NR1,NR2,INCR,K1,K2,INCS,NRECEND,NPIXEND,BOL);

               TEST;
               TRUTHMAP(NR1,NR2,INCR,K1,K2,INCS,NRECEND,NPIXEND,BOL);

               BOL:=FALSE;
               TRUTHMAP(NR1,NR2,INCR,K1,K2,INCS,NRECEND,NPIXEND,BOL);
               LOCK (DISC);

            END OF GROUPXMAIN;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
PROCEDURE HEADOUT;
%               THE PROCEDURE HEADOUT READS THE THIRD RECORD OF CTAPE
%               AND WRITES THE INFORMATION FROM THE FIRST THREE
%               RECORDS OF CTAPE ON THE LINE PRINTER.
                BEGIN
LIST                L1(FOR I:=0 STEP 1 UNTIL 11 DO CT1[I],NRECEND,NPIXEND),
                    L2(FOR I:=12 STEP 1 UNTIL 11+NCHAN DO[I-11,CT1[I]]),
                    L3(FOR I:=1 STEP 1 UNTIL NCLUST DO
                      [I,NUM[I],NCHAN,FOR J:=1 STEP 1 UNTIL NCHAN DO
                        SIG[J,I]]);
FORMAT              F101(X5,"TAPE NO.=",2A6,X5,"NCHAN=",I5,X5,"NSAMP=",I5/
                         X5,"NR1=",I5,X5,"NR2=",I5,X5,"INCR=",I5/
                         X5,"K1=",I5,X5,"K2=",I5,X5,"INCS=",I5/
                         X5,"NCLUST=",I5,X5,"NBACK=",I5,X5,"NRECEND=",I5,
                         X5,"NPIXEND=",I5),
                    F102(I10,F10.1),
                    F103(I7,I10,X6,*I6);
%                        READ THIRD RECORD OF CLUSTER TAPE
                    READ(CTAPE,NCLUST+1,NUM[*]);
%                        WRITE OUTPUT FROM CLUSTER TAPE
                    WRITE(LINE,F101,L1);
                    WRITE(DISC,F101,L1);
                    WRITE(LINE,<X5,"MAXCLUST=",I5>,MAXCLUST);
                    WRITE(LINE,<X5,"CMIN= ",F7.2>,CMIN);
                    WRITE(LINE,</X5,"MATOT=",I5>,MATOT);
                    WRITE(LINE,<//X9,"I",X4,"DELTA[I]">);
                    WRITE(LINE,F102,L2);
                    WRITE(LINE,<X5,"NS",X5,"NUM[NS]",X8,
                          "VALUES OF SIG[NS,J]">);
                    WRITE(LINE,F103,L3);
                    FOR I:=0 STEP 1 UNTIL NCLUST DO
                          PIXTOT:=PIXTOT+NUM[I];
                    WRITE(LINE,<X5,"THE TOTAL NUMBER OF PIXELS=",I8>,PIXTOT);
                    WRITE(LINE,</X5,"THE NUMBER OF UNCLASSIFIED PIXELS=",
                                 I5/>,NUM[0]);
                END OF HEADOUT;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
PROCEDURE GROUNDTRUTH;

%           STORE INFORMATION IN COSTMATRIX MATN[NS,MAT],
%                READ GROUND TRUTH AND COUNT PIXEL ASSIGNMENTS

            BEGIN
LIST            L4(LB,LE,NSP,NSE);
FORMAT          F104(4I5);

                FOR MAT:=1 STEP 1 UNTIL MATOT DO
                BEGIN
                    READ(CARDS,<I5>,NCARDS);

                    WRITE(LINE,</X5,"THE FOLLOWING",I4,
                      " GROUND TRUTH SITES CONTAIN CLASS NUMBER",I3/>,

                      NCARDS,MAT);
                    WRITE(LINE,<X4,"BEGINNING",X2,"ENDING",X3,
                       "BEGINNING",X2,"ENDING"/X4,"SCAN LINE",
                       X1,"SCAN LINE",X2,"ELEMENT",X3,"ELEMENT">);

                    FOR I:=1 STEP 1 UNTIL NCARDS DO
                    BEGIN
                       READ(CARDS,F104,L4);
                       WRITE(LINE,<4I10>,L4);

                       LB:=IF LB LSS NR1 THEN NR1 ELSE LB;
                       LE:=IF LE GTR NR2 THEN NR2 ELSE LE;
                       NSB:=IF NSB LSS K1 THEN K1 ELSE NSB;
                       NSE:=IF NSE GTR K2 THEN K2 ELSE NSE;
                       NREC1:=INTEGER((LB-NR1)/INCR+1);
                       NREC2:=ENTIER((LE-NR1)/INCR+1);
                       NPIX1:=INTEGER((NSB-K1)/INCS+1);
                       NPIX2:=ENTIER((NSE-K1)/INCS+1);
                       GRND[MAT,I,1]:=LB;
                       GRND[MAT,I,2]:=LE;
                       GRND[MAT,I,3]:=NSB;
                       GRND[MAT,I,4]:=NSE;
                       NCRDS[MAT]:=NCARDS;

                       MR:=NREC1-NROLD-1;
                       SPACE(CTAPE,MR);

                       FOR NREC:=NREC1 STEP 1 UNTIL NREC2 DO
                       BEGIN
                          READ(CTAPE,NPIXEND,PIXELS[*]);

                          FOR NPIX:=NPIX1 STEP 1 UNTIL NPIX2 DO
                          MATN[PIXELS[NPIX-1],MAT]
                                :=MATN[PIXELS[NPIX-1],MAT]+1;
                       END;

                       NROLD:=NREC2;
                    END;
                END;
            END OF GROUNDTRUTH;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
%      -----------------------------------------

PROCEDURE COSTMATRIX;
%
%              PRINTS OUT THE COSTMATRIX MATN[NS,MAT].
%              MATN[NS,MAT] IS EQUAL TO THE NUMBER OF PIXELS IN
%              CLUSTER NUMBER NS THAT ARE KNOWN FROM GROUND TRUTH
%              TO BELONG TO MATERIAL NUMBER MAT.
%              FOR EACH ROW NS, TRANSP[NS] IS EQUAL TO THE COLUMN
%              NUMBER MAT CONTAINING THE LARGEST VALUE OF MATN[NS,MAT].
%              THIS MEANS THAT CLUSTER NUMBER NS HAS BEEN ASSIGNED
%              TO MATERIAL NUMBER MAT.

               BEGIN

%                 ********************************
%            GLOSSARY OF VARIABLES LOCAL TO COSTMATRIX.
%                 ********************************

%                                    HEADING OF THE COSTMATRIX.
INTEGER ARRAY     KOUNT[0:MATOT];%   AN ARRAY CONTAINING THE INTEGERS

%                                    1 TO MATOT FOR PRINTING ON THE
INTEGER           MATMAX;      %     THE MAXIMUM ENTRY IN A GIVEN ROW OF THE
%                                    COSTMATRIX MATN[NS,MAT].
ARRAY             PERHIT[0:MATOT];%  THE PERCENTAGE OF PIXELS THAT ARE
%                                    ACTUALLY OF CLASS MAT THAT HAVE BEEN
%                                    CLASSIFIED AS BELONGING TO MAT BY
%                                    THE COSTMATRIX.
REAL              PERTOT;      %     THE AVERAGE PERCENT CORRECT CLASSIFICATION
%                                    OVER ALL CLASSES FOR CLUSTERS CLASSIFIED
%                                    BY COSTMATRIX.
INTEGER           SKIP;        %     THE NUMBER OF SPACES TO SKIP IN PRINTING
%                                    OUT THE COSTMATRIX FOR A VARIABLE
%                                    NUMBER OF CLASSES.
ARRAY             SUM[0:MATOT];%     THE NUMBER OF PIXELS OF A GIVEN CLASS
%                                    THAT HAVE BEEN ASSIGNED TO THAT
%                                    CLASS BY THE COSTMATRIX.
REAL              TOTAL;       %     THE SUM OF TOTNS[MAT] OVER ALL CLASSES.
ARRAY             TOTM[0:NCLUST];%   AN ARRAY CONTAINING THE SUM OF EACH
%                                    ROW IN THE COSTMATRIX MATN[NS,MAT].
ARRAY             TOTNS[0:MATOT];%   AN ARRAY CONTAINING THE SUM OF
%                                    EACH COLUMN IN THE COSTMATRIX
%                                    MATN[NS,MAT], EXCLUDING THOSE CLUSTERS
%                                    TO BE CLASSIFIED BY POTENTIAL.
REAL              TOTSUM;      %     THE SUM OF SUM[MAT] OVER ALL CLASSES.

FORMAT            F1020(X3,"CLUSTER/CLASS",*I6,X*,"TOTAL",X5,"CLASS",X3,

                     "PERCENT"),
                  F1021(X4,I4,X9,*I6,X*,F10.0,I10,F10.2),
                  F1022(/X5,"TOTAL",X7,*F6.0,X*,F10.0),
                  F1023(/X4,"CORRECT",X6,*F6.0,X*,F10.0),
                  F1024(/X4,"PERCENT",X6,*F6.1,X*,F10.2);

FORMAT            F1025(/X4,"PERCENT",X6,*F6.1,F10.2);


%             ************************
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
%         MAIN BODY OF COSTMATRIX

%   FIND THE MAXIMUM ENTRY IN EACH ROW OF MATN[NS,MAT]
%   AND SET THE CORRESPONDING VALUE OF MAT EQUAL TO
%   TRANSP[NS].  THE RATIO OF THIS MAXIMUM NUMBER TO THE
%   TOTAL OF EACH ROW, STORED AS A PERCENTAGE IN PERCLS[NS],
%   IS A MEASURE OF HOW GOOD THE CLUSTER IS FROM THE POINT
%   OF VIEW OF CONTAINING PIXELS OF ONLY ONE CLASS.
%   ONLY CLASSIFY THOSE CLUSTERS THAT HAVE AT LEAST MINTOT
%   GROUND TRUTH PIXELS IN THEM.  THE REMAINING CLUSTERS
%   WILL BE CLASSIFIED BY POTENTIAL.

FOR NS:=1 STEP 1 UNTIL NCLUST DO
BEGIN
    MATMAX:=0;

    FOR MAT:=1 STEP 1 UNTIL MATOT DO
    BEGIN
        IF MATN [NS,MAT] GTR MATMAX
        THEN
            BEGIN
              MATMAX:=MATN[NS,MAT];
              TRANSP[NS]:=MAT;
            END
        ELSE;
        TOTM[NS]:=TOTM[NS]+MATN[NS,MAT];
    END;
    IF TOTM[NS] LSS MINTOT
    THEN
        TRANSP[NS]:=
        PERCLS[NS]:=0
    ELSE
        PERCLS[NS]:=
        MATMAX/TOTM[NS]×100;
END;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
%                   COSTMATRIX (CONT.)
%        FOR EACH CLASS MAT, THE PERCENTAGE CORRECT
%        CLASSIFICATION AS MEASURED BY THE NUMBER OF GROUND
%        TRUTH PIXELS IN THOSE CLUSTERS ASSIGNED TO MAT
%        DIVIDED BY THE TOTAL NUMBER OF PIXELS IN ALL CLUSTERS
%        THAT ARE KNOWN TO BELONG TO CLASS MAT, IS STORED
%        IN THE ARRAY PERHIT[MAT].
%        THE OVERALL CORRECT CLASSIFICATION FOR ALL
%        CLASSES IS GIVEN BY PERTOT.

FOR MAT:=1 STEP 1 UNTIL MATOT DO
  BEGIN
    FOR NS:=1 STEP 1 UNTIL NCLUST DO
      BEGIN
        IF PERCLS[NS] LSS MINPER
          THEN
            TRANSP[NS]:=0
          ELSE;
        IF TRANSP[NS] EQL MAT AND PERCLS[NS] GEQ MINPER
          THEN
            SUM[MAT]:=
              SUM[MAT]+MATN[NS,MAT]
          ELSE;
        IF TRANSP[NS] NEQ 0 AND PERCLS[NS] GEQ MINPER
          THEN
            TOTNS[MAT]:=
              TOTNS[MAT]+MATN[NS,MAT]
          ELSE;
      END;
    IF TOTNS[MAT] LSS 0.5
      THEN
        PERHIT[MAT]:=0
      ELSE
        PERHIT[MAT]:=
          SUM[MAT]/TOTNS[MAT]×100;
  END;
FOR MAT:=1 STEP 1 UNTIL MATOT DO
  BEGIN
    TOTSUM:=TOTSUM+SUM[MAT];
    TOTAL:=TOTAL+TOTNS[MAT];
  END;
IF TOTAL LSS 0.5
  THEN
    PERTOT:=0
  ELSE
    PERTOT:=TOTSUM/TOTAL×100;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
%               COSTMATRIX (CONT.)
%       WRITE OUT THE COSTMATRIX AND ALL TOTALS.
            WRITE(LINE[PAGE]);
            FOR MAT:= 1 STEP 1 UNTIL MATOT DO
              KOUNT[MAT]:=MAT;
            SKIP:=(10-MATOT)×6+1;
            WRITE(LINE,F1020,MATOT,FOR MAT:=1 STEP 1 UNTIL MATOT
                DO KOUNT[MAT] ,SKIP+5);
            FOR NS:=1 STEP 1 UNTIL NCLUST DO
              WRITE(LINE,F1021,NS,MATOT,FOR MAT:=1 STEP 1 UNTIL
                MATOT DO MATN[NS,MAT],SKIP,TOTM[NS],TRANSP[NS],
                PERCLS[NS]);
            WRITE(LINE,F1022,MATOT,FOR MAT:=1 STEP 1 UNTIL MATOT
                DO TOTNS[MAT],SKIP,TOTAL);
            WRITE(LINE,F1023,MATOT,FOR MAT:=1 STEP 1 UNTIL MATOT
                DO SUM[MAT],SKIP,TOTSUM);
            WRITE(LINE,F1024,MATOT,FOR MAT:=1 STEP 1 UNTIL MATOT
                DO PERHIT[MAT],SKIP,PERTOT);
            WRITE(DISC,F1025,MATOT,FOR MAT:=1 STEP 1 UNTIL MATOT
                DO PERHIT[MAT],PERTOT);
        END OF COSTMATRIX;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
PROCEDURE POTENTIAL;

%           THE PROCEDURE POTENTIAL SORTS THE SPECTRAL SIGNATURES
%           ASSOCIATED WITH EACH CLUSTER INTO TWO GROUPS.  THE
%           SIGNATURES OF CLUSTERS THAT WERE CLASSIFIED BY COSTMATRIX
%           ARE STORED IN THE ARRAY X[I,J,K] AND ARE USED FOR
%           TRAINING THE POTENTIAL CLASSIFIER IN THE PROCEDURE
%           PTRAIN.  THE SIGNATURES OF CLUSTERS THAT WERE NOT
%           CLASSIFIED BY COSTMATRIX ARE STORED IN THE ARRAY
%           Y[J,K] AND ARE THEN CLASSIFIED IN THE PROCEDURE PTEST.

            BEGIN

%              ******************************
%         GLOSSARY OF VARIABLES LOCAL TO POTENTIAL.
%              ******************************

INTEGER ARRAY   COUNT[0:MATOT,0:NCLUST];%  COUNT[I,J] IS THE NUMBER
%                                  OF TIMES THAT THE POTENTIAL FUNCTION
%                                  AT THE SAMPLE LABELED CLASS I IS
%                                  AUGMENTED BY LAMDA IN ORDER TO CORRECTLY
%                                  CLASSIFY ALL LABELED SAMPLES.
ARRAY           G[0:MATOT];%       AN ARRAY CONTAINING THE DISCRIMINANT
%                                  FUNCTION FOR EACH CLASS.
REAL            GMAX;         %    THE MAXIMUM VALUE OF THE DISCRIMINANT
%                                  FUNCTION.
INTEGER         IFLAG;        %    A FLAG TO DETERMINE WHEN ALL TRAINING
%                                  SAMPLES ARE CORRECTLY CLASSIFIED BY
%                                  POTENTIAL.
INTEGER         I;            %    AN INDEX VARIABLE CORRESPONDING TO
%                                  A CLASS NUMBER.
INTEGER         IGMAX;        %    THE NUMBER OF THE CLASS WITH THE LARGEST
%                                  DISCRIMINANT FUNCTION.
INTEGER         J;            %    AN INDEX VARIABLE CORRESPONDING TO THE
%                                  CHANNEL NUMBER.
INTEGER         K;            %    AN INDEX VARIABLE CORRESPONDING TO
%                                  A SAMPLE NUMBER

INTEGER ARRAY   KEEP[0:NCLUST];%   AN ARRAY CONTAINING THE CLUSTER
%                                  NUMBERS FOR EACH CLUSTER THAT IS TO BE
%                                  CLASSIFIED USING THE METHOD OF POTENTIALS.
INTEGER         KSW;          %    A COUNTER TO LIMIT THE TOTAL NUMBER
%                                  OF ITERATIONS IN PTRAIN TO 100.
INTEGER         KT;           %    AN INDEX VARIABLE CORRESPONDING TO
%                                  A SAMPLE NUMBER
INTEGER         L;            %    AN INDEX VARIABLE CORRESPONDING TO
%                                  A CLASS NUMBER.
INTEGER ARRAY   N[0:MATOT];%       N[I] IS AN ARRAY CONTAINING THE NUMBER
%                                  OF SAMPLES LABELED CLASS I THAT ARE
%                                  USED FOR TRAINING THE POTENTIAL CLASSIFIER.
INTEGER         NSAMPK;       %    THE NUMBER OF CLUSTERS TO BE CLASSIFIED
%                                  BY POTENTIAL.
REAL            SUM;          %    THE SQUARE OF THE DISTANCE IN FEATURE
%                                  SPACE BETWEEN A CLUSTER TRAINING SAMPLE
%                                  AND A CLUSTER SAMPLE TO BE CLASSIFIED
%                                  BY POTENTIAL.
ARRAY           X[0:MATOT,0:NCLUST,0:NCHAN];%  X[I,J,K] IS THE
%                                  SPECTRAL SIGNATURE OF THE K TH SAMPLE
%                                  LABELED CLASS I.
ARRAY           Y[0:NCLUST,0:NCHAN];%  AN ARRAY CONTAINING THE SPECTRAL
%                                  SIGNATURE OF EACH CLUSTER TO BE
%                                  CLASSIFIED BY POTENTIAL.
INTEGER ARRAY   WT[0:MATOT,0:NCLUST];%  WEIGHTING FACTOT ACCOUNTING
%                                       FOR THE VARYING NUMBER OF
%                                       PIXELS IN EACH CLUSTER.

%      -------------------------------------

%          PROCEDURES PTRAIN AND PTEST ARE INSERTED HERE

%      -------------------------------------
```

ORIGINAL PAGE IS OF POOR QUALITY

```
%        **************************
%                    MAIN BODY OF POTENTIAL
             FOR I:=1 STEP 1 UNTIL MATOT DO
                  N[I]:=0;
             K:=0;
             FOR NS :=1 STEP 1 UNTIL NCLUST DO
                 IF TRANSP[NS] EQL 0 OR PERCLS[NS] LSS MINPER
                                          THEN
                                           BEGIN
                                             K:=K+1;
                                             KEEP[K]:=NS;
                                             FOR J:=1 STEP 1 UNTIL NCHAN DO
                                                  Y[K,J]:=SIG[J,NS];
                                           END
                 ELSE
                  BEGIN
                    I:=TRANSP[NS];
                    N[I]:=N[I]+1;
                    WT[I,N[I]]:= NUM[NS];
                    FOR J:=1 STEP 1 UNTIL
                                NCHAN DO
                        X[I,N[I],J]:=SIG[J,NS];
                  END;
             NSAMPK:=K;
             WRITE(LINE,</X5,"I",X5,"N[I]"/>);
             FOR I:=1 STEP 1 UNTIL MATOT DO
                  WRITE(LINE,<I6,I10>,I,N[I]);
             PTRAIN;
             PTEST;
          END OF POTENTIAL;
%     ---------------------------------------------
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
PROCEDURE PTRAIN;
%          THE PROCEDURE PTRAIN ITERATIVELY MODIFIES THE DISCRIMINANT
%          FUNCTIONS G[I] UNTIL ALL OF THE LABELED SAMPLES WHOSE
%          SPECTRAL SIGNATURES ARE STORED IN THE ARRAY X[L,J,KT]
%          ARE CORRECTLY CLASSIFIED.
      BEGIN
         IFLAG:=1;
         KSW:=0;
         FOR I:=1 STEP 1 UNTIL MATOT DO
            FOR J:=1 STEP 1 UNTIL N[I] DO
               COUNT[I,J]:=0;
         WHILE IFLAG EQL 1 AND KSW LSS 100 DO
         BEGIN
           IFLAG:=0;
           FOR L:=1 STEP 1 UNTIL MATOT DO
              FOR KT:=1 STEP 1 UNTIL N[L] DO
              BEGIN
                 GMAX:=0;
                 FOR I:=1 STEP 1 UNTIL MATOT DO
                       G[I]:=0;
                 FOR I:=1 STEP 1 UNTIL MATOT DO
                    IF N[I] NEQ 0
                          THEN
                          BEGIN
                            FOR K:=1 STEP 1 UNTIL N[I] DO
                            BEGIN
                              SUM:=0;
                              FOR J:=1 STEP 1 UNTIL NCHAN DO
                                 SUM:=SUM+(X[L,KT,J]-X[I,K,J])*2;
                              G[I]:=G[I]+((1+LAMDA×COUNT[I,K])/
                                          (1+ALFA×SUM))×
                                           W[I,K];
                            END;
                            G[I]:=G[I]/N[I];
                            IF G[I] GTR GMAX
                                        THEN
                                          BEGIN
                                            GMAX:=G[I];
                                            IGMAX:=I
                                          END
                            ELSE;
                          END
                    ELSE;
                 IF IGMAX NEQ L
                          THEN
                            BEGIN
                              IFLAG:=1;
                              COUNT[L,KT]:=COUNT[L,KT]+1;
                            END
                 ELSE;
              END;
              KSW:=KSW+1;
         END;
         WRITE(LINE,<" KSW= ",I5>,KSW);
      END OF PTRAIN;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
PROCEDURE PTEST;
%           THE PROCEDURE PTEST USES THE DISCRIMINANT FUNCTIONS
%           CALCULATED IN PTRAIN TO CLASSIFY THE CLUSTERS, WITH
%           SPECTRAL SIGNATURES STORED IN THE ARRAY Y[J,KT], THAT WERE
%           NOT CLASSIFIED BY THE PROCEDURE COSTMATRIX.
    BEGIN
        FOR KT:=1 STEP 1 UNTIL NSAMPK DO
        BEGIN
            GMAX:=0;
            FOR I:=1 STEP 1 UNTIL MATOT DO
                G[I]:=0;
            FOR I:=1 STEP 1 UNTIL MATOT DO
                IF N[I] EQL 0
                                                            THEN
                                                              IGMAX:=0
                ELSE
                BEGIN
                  FOR K:=1 STEP 1 UNTIL N[I] DO
                  BEGIN
                     SUM:=0;
                     FOR J:=1 STEP 1 UNTIL NCHAN DO
                        SUM:=SUM+(Y[KT,J]-X[I,K,J])*2;
                     G[I]:=G[I]+((1+LAMDA×COUNT[I,K])/(1+ALFA×SUM))×
                                                  WT[I,K];
                  END;
                  G[I]:=G[I]/N[I];
                  IF G[I] GTR GMAX
                               THEN
                                 BEGIN
                                    GMAX:=G[I];
                                    IGMAX:=I;
                                 END
                  ELSE;
                END;
            TRANSP[KFEP[KT]]:=IGMAX;
        END;
    END OF PTEST;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
%       ------------------------------------------------
PROCEDURE TAPEOUTPUT;

%            THE PROCEDURE TAPEOUTPUT PRODUCES THE OUTPUT TAPE
%            OUTAPE.  THE FIRST RECORD OF OUTAPE CONTAINS THE CONTENTS
%            OF CT1[*].  EACH SUCCEEDING RECORD CONTAINS THE
%            CLASSIFICATION OF EACH PIXEL IN A GIVEN SCAN LINE.
%            THE ARRAY PIXELS[*] CONTAINS THE CLUSTER NUMBERS
%            FOR EACH PIXEL IN A SCAN LINE AND THE ARRAY KLASS[*]
%            CONTAINS THE CORRESPONDING CLASS TO WHICH EACH PIXEL
%            HAS BEEN ASSIGNED BY EITHER COSTMATRIX OR POTENTIAL.
            BEGIN
                WRITE(OUTAPE,51,CT1[*]);

                SPACE(CTAPE,-NROLD);
                FOR NREC:=1 STEP 1 UNTIL NRECEND DO
                BEGIN
                   READ(CTAPE,NPIXEND,PIXELS[*]);
                   FOR NPIX:=0 STEP 1 UNTIL NPIXEND-1 DO
                   KLASS[NPIX]:=TRANSP[PIXELS[NPIX]];
                   WRITE(OUTAPE,NPIXEND,KLASS[*]);
                END;
                WRITE(LINE,<X5,I5,"SCAN LINES,EACH CONTAINING",I5,
                     "PIXELS HAVE BEEN WRITTEN ON OUTPUT TAPE">,
                     NRECEND,NPIXEND);
                   REWIND(OUTAPE);
            END OF TAPEOUTPUT;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
 %          *************************

 PROCEDURE        TRUTHMAP    (NR1,NR2,INCR,K1,K2,INCS,NRCD,KSMP,BOL);
 INTEGER          NR1;                          % BEGINNING RECORD TO BE
 %                                                PROCESSED
 INTEGER          NR2;                          % FINAL RECORD TO BE PROCESSED
 INTEGER          INCR;                         % THE SCAN-LINE INCREMENT
 INTEGER          K1;                           % BEGINNING SAMPLE NUMBER
 INTEGER          K2;                           % ENDING SAMPLE NUMBER
 INTEGER          INCS;                         % SAMPLE NUMBER INCREMENT
 INTEGER          NRCD;                         % NUMBER OF RECORDS TO BE
 %                                                PROCESSED
 INTEGER          KSMP;                         % NUMBER OF SAMPLES TO BE
 %                                                PROCESSED
 BOOLEAN          BOL;            %  A BOOLEAN VARIABLE USED TO PRINT MAP
              BEGIN
 %=========================================================================
 INTEGER ARRAY   SCALE[0:3,0:KSMP DIV 5];% OUTPUT SCALE FOR THE SAMPLE
 %                                                NUMBER AXIS

 %                   PROCEDURE SAMPNOS INSERTED HERE

 %                   PROCEDURE PLOT INSERTED HERE

 REAL ARRAY      COMBMAP[0:NRCD,0:KSMP]; % USED TO PRINTOUT THE SPECIFIED
 %                                         AREA. CLUSTER NUMBERS ARE
 %                                         ARE PLACED AT GROUNDTRUTH
 %                                         LOCATIONS
 REAL ARRAY      CHAR[1:9];              % USED TO STORE 1 THRU 9 AS
 %                                         CHARACTER DATA
 INTEGER ARRAY BRC[0:50];                % SCALING ARRAY
 INTEGER ARRAY ERC[0:50];                % SCALING ARRAY
 INTEGER ARRAY BSMP[0:50];               % SCALING ARRAY
 INTEGER ARRAY ESMP[0:50];               % SCALING ARRAY
 INTEGER         I,J,K,L;                % COUNTERS
 FORMAT IN       FMT3(X20,2I5,X5,2I5);

 FORMAT OUT      FMT8("   THIS IS THE COMBINED GROUNDTRUTH MAP.",///);
 %                 *** MAIN BODY OF GROUNDTRUTH ***
                   IF BOL
                   THEN
                   BEGIN
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
%           *** INITIALIZES THE COMBMAP ***
            FOR J:=1 STEP 1 UNTIL NRCD
            DO FOR K:=1 STEP 1 UNTIL KSMP
               DO COMBMAP[J,K]:=" ";
%           *** SETS UP THE CHARACTER VECTOR ***
            FILL CHAR[*] WITH "1","2","3","4","5","6","7","8","9";

%           *** GENERATES THE SCALE ALONG THE SAMPLE NUMBER AXIS ***
                SAMPNOS(K1,KSMP,INCS);
%           *** PROCESSES THE CLASSES ***
            FOR L:=1 STEP 1 UNTIL MATOT
            DO BEGIN
                 FOR I:=1 STEP 1 UNTIL NCRDS[L] DO
              BEGIN
             BRC[I]:=IF GRND[L,I,1] LSS NR1 THEN NR1
                                            ELSE GRND[L,I,1];
              BRC[I]:=INTEGER((BRC[I]-NR1)/INCR+1);
              ERC[I]:=IF GRND[L,I,2] GTR NR2 THEN NR2
                                            ELSE GRND[L,I,2];
              ERC[I]:=(ERC[I]-NR1) DIV INCR+1;
              BSMP[I]:=IF GRND[L,I,3] LSS K1 THEN K1
                                            ELSE GRND[L,I,3];
              BSMP[I]:=INTEGER((BSMP[I]-K1)/INCS+1);
              ESMP[I]:= IF GRND[L,I,4] GTR K2 THEN K2
                                            ELSE GRND[L,I,4];
              ESMP[I]:=(ESMP[I]-K1) DIV INCS+1;
              END;
%          *** ASSIGNS THE CLASS NUMBER TO COMBMAP AT GROUND DATA
%               LOCATIONS ***
           FOR I:=1 STEP 1 UNTIL NCRDS[L]
              DO FOR J:=BRC[I] STEP 1 UNTIL ERC[I]
                 DO FOR K:=BSMP[I] STEP 1 UNTIL ESMP[I]
                    DO BEGIN
                        COMBMAP[J,K]:=CHAR[L];
                        END;
              END;
%          *** PRINTS OUT THE COMBINED GROUNDTRUTH MAP ***
           WRITE(LINE[PAGE]);
           WRITE(LINE,FMT8);
           PLOT(COMBMAP);
           END
           ELSE
             BEGIN
               WRITE(LINE[PAGE]);
               SAMPNOS(K1,KSMP,INCS);
               PLOT(CLASS);
             END;
     END OF TRUTHMAP;
```

ORIGINAL PAGE IS OF POOR QUALITY

```
PROCEDURE       SAMPNOS(K1,KSMP,INCS);
INTEGER         K1;                          % BEGINNING SAMPLE NUMBER
INTEGER         KSMP;                        % NUMBER OF SAMPLES TO BE
%                                              PROCESSED
INTEGER         INCS;                        % SAMPLE NUMBER INCREMENT
                BEGIN
%=========================================================================
INTEGER         TEMP;                        % INITIALIZED TO THE BEGINNING
%                                              SAMPLE NUMBER, THEN INCREMENTED
%                                              TO CREATE THE SAMPLE NUMBER
%                                              SCALE
INTEGER         I,J;                         % COUNTERS
%                   ******************************
%               *** MAIN BODY OF SAMPNOS ***
                TEMP:=K1;
                FOR I:=1 STEP 1 UNTIL KSMP DIV 5
                DO BEGIN
%                   *** CONVERTS A SAMPLE NUMBER INTO A COLUMN VECTOR ***
                    SCALE[1,I]:=TEMP DIV 100;
                    SCALE[2,I]:=(TEMP MOD 100) DIV 10;
                    SCALE[3,I]:=TEMP MOD 10;
                    TEMP:=TEMP+5×INCS;
                   END;
            END OF SAMPNOS;
```

```
PROCEDURE        PLOT(MAP);
INTEGER ARRAY    MAP[0,0];                      % SPECIFIED AREA TO BE PRINTED
         BEGIN
%=========================================================================
INTEGER          SCL;                           % INITIALIZED TO THE INITIAL
%                                                 RECORD NUMBER, THEN IS
%                                                 INCREMENTED TO GENERATE THE
%                                                 THE SCALE ALONG THE RECORD
%                                                 NUMBER AXIS
INTEGER          I,J,K;                         % COUNTERS
FORMAT OUT       FMT3(X5,120A1);


FORMAT OUT       FMT4(X5,120T1);

FORMAT OUT       FMT6(X1,I3);


FORMAT OUT       FMT7(X5,25(I1,X4));

%                       ********************************
%                *** MAIN BODY OF PLOT ***
%                *** PRINTS OUT THE SCALE ALONG THE SAMPLE NUMBER AXIS ***
                 FOR J:=1 STEP 1 UNTIL 3
                 DO WRITE(LINE,FMT7,FOR I:=1 STEP 1 UNTIL KSMP DIV 5
                                    DO SCALE[J,I]);
                 SCL:=NR1;
                 FOR J:=1 STEP 1 UNTIL NRCD
                 DO BEGIN
%                     *** PRINTS OUT THE RECORD NUMBER EVERY FIFTH PASS ***
                      IF J MOD 5 EQL 1
                      THEN
                          BEGIN
                              WRITE(LINE[NO],FMT6,SCL);
                              SCL:=SCL+5×INCR;
                          END
                                                          ELSE;
%                     *** PRINTS OUT A SCAN-LINE ***
                 IF BOL
                 THEN
                      WRITE(LINE,FMT3,FOR K:=1 STEP 1 UNTIL KSMP
                                DO MAP[J,K])
                 ELSE
                      WRITE(LINE,FMT4,FOR K:=0 STEP 1 UNTIL KSMP-1
                                    DO MAP[J,K]);
                   END;
            END OF PLOT;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
PROCEDURE TEST;
          BEGIN
INTEGER         COL;          %  AN INDEX VARIABLE

INTEGER ARRAY   ERRMAT[0:10,0:10];% THE CLASSIFICATION ERROR MATRIX
ARRAY           PERCOR[0:10];%  PERCENT OF TRUE CLASS THAT ARE
%                             CLASSIFIED CORRECTLY
ARRAY           PERCORCOL[0:10]; % PERCENTAGE OF PIXELS WE CALL MAT
%                             THAT ARE REALLY MAT
INTEGER ARRAY   SUM[0:20];  %  THE SUM OF EACH ROW IN ERRMAT
INTEGER         SUMROW;     %  THE SUM OF ALL ELEMENTS IN ERRMAT
INTEGER ARRAY   TOT[0:20];  %  THE SUM OF EACH COLUMN IN ERRMAT
INTEGER         TOTDIAG;    %  THE SUM OF THE DIAGONAL ELEMENTS IN ERRMAT
REAL            TOTSUM;     %  THE OVERALL PERCENT CORRECT CLASSIFICATION

FORMAT    F1(/,X13,5I7,X5,I5,X5,F6.2),

          F2(//,X5,"PERCENT ",5F7.2,X15,F6.2),
          F3(/,X5,"SUM",X5,5I7,X5,I5);

LIST      L1(FOR I:=1 STEP 1 UNTIL MATOT DO ERRMAT[MAT,I],SUM[MAT],
             PERCOR[MAT]),
          L2(FOR I:=1 STEP 1 UNTIL MATOT DO TOT[I],SUMROW),
          L3(FOR I:=1 STEP 1 UNTIL MATOT DO PERCORCOL[I],TOTSUM);

          WRITE(LINE,<///,X23,"TEST  ERROR  MATRIX ">);

          WRITE(LINE,<////,X25,"CLASSIFIED",X20,"SUM",X5,"PERCENT",
                   //,X5,"ACTUAL">);

                SPACE(OUTAPE,1);
                FOR NS:=1 STEP 1 UNTIL NRECEND DO

                   READ(OUTAPE,NPIXEND,CLASS[NS,*]);

                FOR MAT:=1 STEP 1 UNTIL MATOT DO

                BEGIN
                   READ(CARDS,<I5>,NCARDS);

                   FOR I:=1 STEP 1 UNTIL NCARDS DO

                   BEGIN

                     READ(CARDS,<4I5>,LB,LE,NSB,NSE);

                      LB:=IF LB LSS NR1 THEN NR1 ELSE LB;
                      LE:=IF LE GTR NR2 THEN NR2 ELSE LE;
                      NSB:=IF NSB LSS K1 THEN K1 ELSE NSB;
                      NSE:=IF NSE GTR K2 THEN K2 ELSE NSE;
                     NREC1:=INTEGER((LB-NR1)/INCR+1);
                     NREC2:=ENTIER((LE-NR1)/INCR+1);
                     NPIX1:=INTEGER((NSB-K1)/INCS+1);
                     NPIX2:=ENTIER((NSE-K1)/INCS+1);
                      GRND[MAT,I,1]:=LB;
                      GRND[MAT,I,2]:=LE;
                      GRND[MAT,I,3]:=NSB;
                      GRND[MAT,I,4]:=NSE;
                      NCRDS[MAT]:=NCARDS;

                     FOR NREC:=NREC1 STEP 1 UNTIL NREC2 DO

                       FOR NPIX:=NPIX1 STEP 1 UNTIL NPIX2 DO

                         ERRMAT[MAT,CLASS[NREC,NPIX-1]]:=
                            ERRMAT[MAT,CLASS[NREC,NPIX-1]]+1;

                   END;

                END;
          FOR MAT:=1 STEP 1 UNTIL MATOT DO
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
BEGIN
     FOR COL:=1 STEP 1 UNTIL MATOT DO
     SUM[MAT]:=SUM[MAT]+ERRMAT[MAT,COL];
     IF SUM[MAT] GTR 0
     THEN
          PERCOR[MAT]:=(ERRMAT[MAT,MAT]/SUM[MAT])×100
     ELSE;
     SUMROW:=SUMROW+SUM[MAT];
     WRITE(LINE,F1,L1);
END;
FOR COL:=1 STEP 1 UNTIL MATOT DO
BEGIN
     FOR MAT:=1 STEP 1 UNTIL MATOT DO
     TOT[COL]:=TOT[COL]+ERRMAT[MAT,COL];
     IF TOT[COL] GTR 0
     THEN
          PERCORCOL[COL]:=(ERRMAT[COL,COL]/TOT[COL])×100
     ELSE;
END;
WRITE(LINE,<//>);
WRITE(LINE,F3,L2);
FOR MAT:=1 STEP 1 UNTIL MATOT DO
TOTDIAG:=TOTDIAG+ERRMAT[MAT,MAT];
TOTSUM:=(TOTDIAG/SUMROW)×100;
WRITE(LINE,F2,L3);
WRITE(DISC,F2,L3);
  END OF TEST;
```

ORIGINAL PAGE IS
OF POOR QUALITY

APPENDIX C

ALGOL Listing for Procedure GAUSS Including Procedures

CLASS1
CLASS2
CHOLDET1
CHOLSOL1
CLASS3
CLASS4

If the procedure GAUSS is substitued for the procedure POTENTIAL in GROUPL then a Gaussian Maximum Likelihood classification of the clusters is effected.

C. 2

```
%
PROCEDURE GAUSS(KLASS,NCHANU,NCLUST,SIG,TDFILE,CLASSINDEX,VALPOINT,
                TRANSP);
INTEGER          NCLUST;      %  THE NUMBER OF CLUSTERS THAT HAVE BEEN
%                                CREAEED.
REAL ARRAY       SIG[0,0];    %  AN ARRAY CONTAINING THE AVERAGE SPECTRAL
%                                SIGNATURES ASSOCIATED WITH EACH CLUSTER.
FILE             TDFILE;
INTEGER          KLASS;       %  THE NUMBER OF CLASSES FOR WHICH GROUND
%                                TRUTH IS BEING USED.
INTEGER          NCHANU;      %  NUMBER OF SPECTRAL CHANNELS ON TAPE.
INTEGER          VALPOINT;    %  A POINTER USED TO POINT THE BEGINING
%                                OF VAL[J] IN MATLINE.
INTEGER          CLASSINDEX;% A POINTER THAT POINTS TO THE CLASS
%                                NUMBER IN MATLINE.
INTEGER ARRAY    TRANSP[0];   %  AN ARRAY CONTAINING THE CLASS NUMBER
%                                MAT TO WHICH CLUSTER NUMBER NS
%                                HAS BEEN ASSIGNED.


          BEGIN


FILE IN          TRAIN DISK"TRAIN"/"AX311"(2,15,30);
LABEL            FAIL;
LABEL            DONE;
LABEL            NEXT;
FORMAT           FMT1(/I6,I10);
INTEGER ARRAY    NUM[0:14]; %  AN ARRAY CONTAINING THE NUMBER OF SAMPLES
%                                TRAINING THE POTENTIAL CLASSIFIER.
INTEGER          J;           %  AN INDEX VARIABLE CORRESPONDING TO THE
%                                CHANNEL NUMBER.
INTEGER          N5;          %  AN INDEX VARIABLE CORRESPONDING TO THE
%                              THE  CLUSTER NUMBER.
INTEGER          KL;          %  AN IN INDEX VARIABLE CORRESPUNDING TO THE
%                                CLASS NUMBER.
INTEGER          CLASS;       %  AN INDEX VARIABLE CORRESPONDING TO THE
%                                CLASS NUMBER.
INTEGER          D2;          %  A PARAMETER USED TO COMPUTE THE
%                                DETERMINANT FUNCTION.
INTEGER          NC;          %  AN INDEX VARIABLE CORRESPONDING TO THE
%                                CHANNEL NUMBER.
INTEGER          NC1;         %  AN INDEX VARIABLE CORRESPONDING TO THE
%                                CHANNEL NUMBER.
REAL             GMAX;        %  THE MAXIMUM VALUE OF THE DESCRIMINANT
%                                FUNCTION.
REAL             D1;          %  A NUMBER USED TO CALCULATE THE
%                                DETERMINANT FUNCTION.
REAL             G;           %  THE VALUE OF THE DESCRIMINANT FUNCTION.
REAL             DF1;         %  A NUMBER USED TO CALCULATE THE
%                                WEIGHTING FUNCTION.
REAL             QZ;          %  A NUMBER USED TO CALCULATE THE
%                                VALUE OF THE DESCRIMINANT FUNCTION.
REAL             Z7;          %  A NUMBER USED TO CALCULATE THE
%                                VALUE OF THE DESCRIMINANT FUNCTION.
REAL ARRAY       L[0:14,0:12,0:12];    %  AN ARRAY FOR STORING THE
%                                COVARIANCE MATRIX AND ITS LOWER
%                                TRIANGLER TRANSFORMATION.
REAL ARRAY       SUMSQ[0:14,0:12,0:12]; %  AN ARRAY FOR STORING THE
%                                SUM OF THE PRODECTS VAL[I]×VAL[J].
REAL ARRAY       MU[0:14,0:12];      %   AN ARRAY FOR STORING THE
%                               MEAN VECTOR EACH CLASS.
REAL ARRAY       MU1[0:12]; %   AN ARRAY FOR STORING THE MEAN
%                               VECTOR FOR THE ROWS OF MU.
REAL ARRAY       SUM[0:14,0:12]; %  AN ARRAY FOR STORING THE
%                               SUM OF VAL[I].
REAL ARRAY       VAL[0:12]; %   AN ARRAY  FOR STORING THE
%                               SPECTRAL SIGNATURESOF EACH PIXELS.
REAL ARRAY       P[0:14,0:12]; %  AN ARRAY FOR STORING THE MAIN
%                               DIAGONAL ELEMENTS OF THE LOWER
%                               TRIANGULAR MATRIX.
REAL ARRAY       W[0:14];     %   AN ARRAY CONTAINING THE  WEIGHTING
%                             FUNCTION OF EACH CLASS.
REAL ARRAY       Q[0:14,0:12];
REAL ARRAY       Z[0:14,0:12];
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
%                     MAIN BODY OF GAUSS
                      FOR KL:=1 STEP 1 UNTIL KLASS DO
                      BEGIN
                        NUM[KL]:=0;
                       FOR NC:=1 STEP 1 UNTIL NCHANU DO
                        BEGIN
                           SUM[KL,NC]:=0.0;
                          FOR NC1:=1 STEP 1 UNTIL NCHANU DO
                           BEGIN
                                SUMSQ[KL,NC,NC1]:=0;
                                L[KL,NC,NC1]:=0;
                           END;
                        END;
                      END;
                      WHILE TRUE DO
                      BEGIN
                           READ(TRAIN,15,MATLINE[*]) [NEXT];
                           KL:=MATLINE[CLASSINDEX];
                           FOR J:=1 STEP 1 UNTIL NCHANU DO
                             VAL[J]:=MATLINE[VALPOINT+J-1];
                           CLASS1(KL,NCHANU,VAL,NUM,SUM,SUMSQ);
                      END;
    NEXT:             CLASS2(NCHANU,NUM,SUM,SUMSQ,MU,L);
                          FOR KL:=1,2,3,4,6 DO
                         BEGIN
                           FOR J:=1 STEP 1 UNTIL NCHANU DO
                             MU1[J]:=MU[KL,J];
                             CHOLDET1(NCHANU,L,P,D1,D2,FAIL);
                             CHOLSOL1(NCHANU,L,P,MU1,Q);
                             CLASS3;
                           GO TO DONE;
     FAIL:             WRITE(LINE,<"MATRIX NOT POSITIVE DEFINITE FOR

                                CLASS",I4>,KL);
     DONE:             END;
                         WRITE(LINE[PAGE]);
                      WRITE(LINE,<X5,"NS",X10,"CLASS">);
                      FOR NS:=1 STEP 1 UNTIL NCLUST DO
                      BEGIN
                         FOR J:=1 STEP 1 UNTIL NCHANU DO
                           VAL[J]:= SIG[J,NS];
                         CLASS:=1;
                          FOR KL:=1,2,3,4,6 DO
                           BEGIN
                              CHOLSOL1(NCHANU,L,P,VAL,Z);
                              CLASS4;
                           END;
                         TRANSP[NS]:=CLASS;
                         WRITE(LINE,FMT1,NS,CLASS);
                      END;

                       END OF GAUSS;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
PROCEDURE CLASS1(KL,NCHANU,VAL,NUM,SUM,SUMSQ);
INTEGER          KL;          %  AN INDEX VARIABLE CORRESPONDING TO THE
%                                CLASS NUMBER.
INTEGER          NCHANU;      %  AN INDEX VARIABLE CORRESPONDING TO THE
%                                CHANNEL NUMBER.
REAL ARRAY       VAL[0];      %  AN ARRAY CONTAINING THE SPECTRAL
%                                SIGNATURES
INTEGER ARRAY NUM[0];         %  AN ARRAY CONTAINING THE NUMBER OF SAMPLES
%                                FOR EACH CLASS THAT ARE USED FOR
%                                TRAINING THE POTENTIAL CLASSIFIER.
REAL ARRAY       SUM[0,0];
REAL ARRAY       SUMSQ[0,0,0];
      BEGIN

            FOR NC:=1 STEP 1 UNTIL NCHANU DO

            BEGIN
                SUM[KL,NC]:=SUM[KL,NC]+VAL[NC];
                FOR NC1:=NC STEP 1 UNTIL NCHANU DO

                SUMSQ[KL,NC,NC1]:=SUMSQ[KL,NC,NC1]+VAL[NC]
                                  ×VAL[NC1];
            END;

               NUM[KL]:=NUM[KL]+1;

      END OF CLASS1;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
PROCEDURE CLASS2(NCHANU,NUM,SUM,SUMSQ,MU,L);
INTEGER          NCHANU;      %  AN INDEX VARIABLE CORRESPONDING
%                                TO THE CHANNEL NUMBER.
INTEGER ARRAY NUM[0];         %  AN ARRAY CONTAINING THE NUMBER OF
%                                SAMPLES FOR EACH CLASS THAT ARE USED
%                                FOR TRAINING THE POTENTIAL CLASSIFIER.
REAL ARRAY       SUM[0,0];
REAL ARRAY       SUMSQ[0,0,0];
REAL ARRAY       MU[0,0];
REAL ARRAY       L[0,0,0];
         BEGIN
LIST             L2(FOR NC:=1 STEP 1 UNTIL NCHANU DO MU[KL,NC]);
LIST             L3( FOR NC:=1 STEP 1 UNTIL NCHANU DO
                      FOR NC1:=1  STEP 1 UNTIL NCHANU DO L[KL,NC,NC1]);
FORMAT           FMT2(//4E20.4/);
FORMAT           FMT3(4E20.4);
```

```
FOR KL:=1 STEP 1 UNTIL KLASS DO
    BEGIN
     FOR NC:=1 STEP 1 UNTIL NCHANU DO
        MU[KL,NC]:= SUM[KL,NC]/NUM[KL];
     FOR NC:=1 STEP 1 UNTIL NCHANU DO
        FOR NC1:=NC STEP 1 UNTIL NCHANU DO
           L[KL,NC,NC1]:=(SUMSQ[KL,NC,NC1]/NUM[KL])-
                          MU[KL,NC]×MU[KL,NC1];
    WRITE(LINE,FMT2,L2);
    WRITE(LINE,FMT3,L3);
    END;
END OF CLASS2;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
PROCEDURE CHOLDET1(N,L,P,D1,D2,FAIL);

INTEGER          N;           %  AN INDEX VARIABLE CORRESPONDING TO
%                                THE CHANNEL NUMBER.
REAL ARRAY       L[0,0,0];
REAL ARRAY       P[0,0];
REAL             D1;          %  A NUMBER USED TO CLACULATE THE
%                               DETERMINANT FUNCTION.
INTEGER          D2;          %  A NUMBER USED TO CLACULATE THE
%                               DETERMINANT FUNCTION.
LABEL            FAIL;
%          CHOLESKY DEFACTORIZATION TO PRODUCE L
           BEGIN
INTEGER          I;
INTEGER          J;
INTEGER          K;
REAL             V;

              D1:=1;   D2:=0;
              FOR I:=1 STEP 1 UNTIL N DO
                 FOR J:=I STEP 1 UNTIL N DO
              BEGIN
                      V:=L[KL,I,J];
                      FOR K:=I-1 STEP -1 UNTIL 1 DO
                         V:=V-L[KL,J,K]×L[KL,I,K];
                         IF J EQL I THEN
                         BEGIN
                            D1:=D1×V;
                            IF V EQL 0 THEN
                            BEGIN
                               D2:=0;
                               GO TO FAIL;
                            END
                         ELSE;
                         WHILE ABS(D1) GEQ 1 DO
                         BEGIN
                           D1:=D1×0.0625;
                           D2:=D2+4;
                         END;
                    WHILE ABS(D1) LSS 0.0625 DO
                    BEGIN
                        D1:=D1×16;
                        D2:=D2-4;
                    END;
                IF V LSS 0 THEN GO TO FAIL;
                P[KL,I]:=1.0/SQRT(V);
                END
                ELSE L[KL,J,I]:=V×P[KL,I];
             END;
       END OF CHOLDET1;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
PROCEDURE CHOLSOL1(N,L,P,B,X);
INTEGER          N;         %   AN INDEX VARIABLE CORRESPONDING
%                               TO THE CHANNEL NUMBER,
REAL ARRAY       L[0,0,0];
REAL ARRAY       P[0,0];
REAL ARRAY       B[0];
REAL ARRAY       X[0,0];

%                SOLUTION OF AX=B

           BEGIN
INTEGER          I;
INTEGER          J;
INTEGER          K;
REAL             V;

% SOLUTION OF LY=B;

          FOR I:=1 STEP 1 UNTIL N DO
          BEGIN V:=B[I];
                FOR K:=I-1 STEP -1 UNTIL 1 DO
                    V:=V-L[KL,I,K]×X[KL,K];
                X[KL,I]:=V×P[KL,I];
          END;
       END OF CHOLSOL1;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
PROCEDURE CLASS3;
%               DETERMINE W
        BEGIN
              W[KL]:=0.0;
              FOR NC:=1 STEP 1 UNTIL NCHANU DO
                    W[KL]:=W[KL]+Q[KL,NC]×Q[KL,NC];
              DET:=D1×2*D2;
              W[KL]:=-.5×W[KL]-.5×LN(DET);
               W[KL]:=W[KL]+LN(APPROB[KL]);
        END OF CLASS3;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
PROCEDURE CLASS4;
%          CLASSIFY BY G
           BEGIN

           QZ:=0,0;   ZZ:=0,0;
           FOR NC:=1 STEP 1 UNTIL NCHANU DO
           BEGIN
                QZ:=QZ+Q[KL,NC]×Z[KL,NC];
                ZZ:=ZZ+Z[KL,NC]×Z[KL,NC];
           END;
           G:=.5×ZZ+QZ+W[KL];
           IF KL EQL 1 THEN GMAX:=G;
           IF G GTR GMAX THEN
           BEGIN

                GMAX:=G;
                CLASS:=KL;
           END;
       END OF CLASS4;
```

ORIGINAL PAGE IS
OF POOR QUALITY

APPENDIX D

ALGOL Listing of Procedure CHIMP

Including Procedures

DUMPDATAROW
DUMPDATA
OUTPUTSUBLIST
NEWCBOX
NEWNODE
INITIALIZATION
SETTREE
INWINDOW
INPUT
DUMPTREE
TREECLIMBER
DISTSQ
POTENTIAL
DISCRIMINANT
CLASSIFIEDCORRECTLY
CHECKSUBLIST
TREECHECKER
CLASSIFYTRAIN

If the procedure CHIMP is substituted for the procedure POTENTIAL in GROUPL then a hierarchical classification using the method of potentials is effected.

ORIGINAL PAGE IS
OF POOR QUALITY

```
    PROCEDURE CHIMP (NCHAN, NLEV, NSONS, NODES, WINDOWSIZE, THRESH,NROWS,
       ALFA, LAMBDA, MAXLEVEL, TDFILE, FIRST,SECOND,DEBUG, NCLUST,
        NCENTERS,SIG,OUTPUTARRAY);

%   PARAMETER SPECIFICATIONS:
       BOOLEAN  DEBUG;
       INTEGER
           NCHAN,                   % DIMENSION AF A FEATURE VECTOR.
           NLEV,                    % NO. OF LEVELS OF CLASSIFICATION.
           NSONS,                   % MAX. NO. OF CLASSES AT ANY LEVEL.
           NODES,                   % NO. OF TREE NODES TO BE AVAILABLE.
           NROWS;                   % NO OF TRAINING SAMPLES
       REAL
           THRESH,                  % SPECIFIES THRESHOLD AS FRACTION OF MAXPOT
           WINDOWSIZE,              % FOR CLUSTERING TRAINING DATA.
           ALFA,                    % PARAMETERS  OF
           LAMBDA;                  % THE POTENTIAL FUNCTION.
       FILE  TDFILE;                % THE FILE FROM WHICH TRAINING DATA IS READ.
       INTEGER
           FIRST,                   % STARTING POSITION OF CLASS DATA IN FILE.
           SECOND,                  % STARTING POS. OF FEATURE DATA IN FILE
           NCLUST,                  % NO. OF DATA TO BE CLASSIFIED.
           NCENTERS,                % NO. OF POTENTIAL CENTERS USED
           MAXLEVEL;                % DESIRED LEVEL OF CLASSIFICATION.
       REAL ARRAY
           SIG [0,0],               % ARRAY OF FEATURES OF DATA TO BE CLASSIFIED
           OUTPUTARRAY [0,0];       % FOR CLASSIFICATION  RESULTS.

    BEGIN  % ALLOCATE GLOBAL DATA
       FILE OUT LINE PRINT (2,15);
       INTEGER
           K,M,
           CLISTHEAD,
           CAVAIL,
           TROOT,
           TAVAIL;

       REAL
           MAXPOT;
       INTEGER ARRAY
           NEWCLASS,
           TRAIL  [1:NLEV],
           COUNT,
           CLASS,
           CLINK  [1:NROWS,1:NLEV],
           WEIGHT,
           CLISTLINK [1:NROWS],
           TNODE [1:NODES,1:NSONS+1];

       REAL ARRAY
              NEWFEATURE,
              WINDOW [1:NCHAN],
              FEATURE [1:NROWS,1:NCHAN],
              DUMMY [0:SECOND-1+NLEV];
       DEFINE THRESHOLD=MAXPOTxTHRESH#;

       LABEL    IPT, DONE;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      PROCEDURE DUMPDATAROW (ROW,TAB);

         INTEGER ROW,TAB;
      BEGIN     INTEGER K, NNCHAN,NNLEV;
            NNCHAN:=NCHAN;  NNLEV := NLEV;
            WRITE (LINE,
            <X*,I3,":",*F8.4,";",*I3,";",I3,";",*I3>,
            5×TAB, ROW,
            NNCHAN,FOR K:=1 STEP 1 UNTIL NCHAN DO FEATURE[ROW,K],
            NNLEV  ,FOR K := 1 STEP 1 UNTIL NLEV    DO CLASS[ROW,K],
            WEIGHT[ROW],
            NNLEV  ,FOR K:=1 STEP 1 UNTIL NLEV    DO CLINK[ROW,K]);
      END OF DUMPDATAROW;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      PROCEDURE DUMPDATA;

      BEGIN  INTEGER K;
         WRITE (LINE,</"DUMPDATA:">);
         FOR K :=1 STEP 1 UNTIL NROWS DO DUMPDATAROW(K,1);
      END OF DUMPDATA;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      PROCEDURE OUTPUTSUBLIST (P,LEVEL,KLASS);  VALUE P;

         INTEGER
            P,
            LEVEL,
            KLASS;
      BEGIN
         IF P=0 THEN %RETURN
         ELSE
         BEGIN
            WRITE (LINE);
            P := TNODE[P,1];
            WHILE P>0 DO
            BEGIN
               DUMPDATAROW (P,LEVEL);
               P:= CLINK [P,LEVEL]
            END
         END
      END OF OUTPUTSUBLIST;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      INTEGER PROCEDURE NEWCBOX;

%     GET A NEW DATA STORAGE BOX.  RETURN THE LOCATION (ROW NUMBER) OF
%     THE BOX AND CLEAR THE BOX.. ALSO ADJUST THE AVAIL STACK.

      BEGIN  INTEGER NB,K;

         IF CAVAIL=0    THEN BEGIN
                                WRITE (LINE,
                                <"***OVERFLOW DATA MEMORY***">);
                                % GENERATE DIVIDE EXCEPTION TO HALT EXEC
                                CAVAIL:=1/CAVAIL;
                             END
         ELSE BEGIN
                 NB := CAVAIL;  CAVAIL := CLISTLINK[CAVAIL];
                    FOR K←1 STEP 1 UNTIL NLEV DO  CLASS[NB,K]←0;
                    NEWCBOX := NB
              END
      END OF NEWCBOX;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      INTEGER PROCEDURE NEWNODE;

%     THIS PROCEDURE RETURNS THE LOCATION OF A NEW NODE BOX WHICH HAS
%     BEEN CLEARED AND READIED FOR USE

      BEGIN   INTEGER K,NN;
         IF TAVAIL=0 THEN BEGIN
                             WRITE(LINE,<"***OVERFLOW TREE MEMORY**">);
                             TAVAIL:=1/TAVAIL % GENERATE DIV EXCEPTION
                          END  ;              % TO HALT EXECUTION
         NN := TAVAIL;
         NEWNODE := NN;
         TAVAIL := TNODE[TAVAIL,1];
         FOR K:=1 STEP 1 UNTIL NLEV    DO TNODE[NN,K]:=0;
      END OF NEWNODE;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     PROCEDURE INITIALIZATION;

%    INITIALIZE LINKED STORAGE.

     BEGIN INTEGER K;
       FOR K:=1 STEP 1 UNTIL NROWS-1 DO CLISTLINK[K]:=K+1;
       FOR K:=1 STEP 1 UNTIL NODES-1 DO TNODE[K,1]:=K+1;
       CLISTLINK [NROWS]:=0 ;  TNODE[NODES,1] := 0;
       CAVAIL:= TAVAIL:= 1;  TROOT := 0;
       FOR K:= 1 STEP 1 UNTIL NCHAN DO WINDOW[K]:=WINDOWSIZE;
     END OF INITIALIZATION;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     PROCEDURE SETTREE (NEWCLASS);

%    PUT NEW NODES INTO THE TREE AS REQUIRED TO ACCOMODATE "NEWCLASS".
%    ASSIGN TO "TRAIL" THE LOCATION OF TREE NODES THAT POINT TO THE
%    SUBLISTS OF "NEWCLASS"
       INTEGER ARRAY NEWCLASS[1];

     BEGIN  INTEGER P,Q,NC,L;
       IF TROOT = 0 THEN TROOT := NEWNODE;
       P := TROOT ;
       FOR L := 1 STEP 1 UNTIL NLEV   DO
       IF 0<NC := NEWCLASS[L] THEN
       BEGIN
          Q := TNODE[P,1+NC];
          IF Q = 0 THEN  BEGIN
                            Q := NEWNODE;
                            TNODE[P,1+NC] := Q;
                         END;
          TRAIL[L] := Q;
          P := Q;
       END
       ELSE TRAIL[L]:=0;
     END OF  SETTREE;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     BOOLEAN PROCEDURE INWINDOW (NEWFEATURE,P);

%    DETERMINES WHETHER OR NOT "NEWFEATURE" IS IN THE WINDOW
%    OF THE FEATURE IN ROW "P".


       REAL    ARRAY NEWFEATURE[1];
       INTEGER P;
     BEGIN  BOOLEAN B;   INTEGER K;
       B := TRUE;
       FOR K:=1 STEP 1
       WHILE K LEQ NCHAN AND B DO
          B := ABS(FEATURE[P,K] - NEWFEATURE[K]) < WINDOW[K]      ;
       INWINDOW := B;
     END OF INWINDOW;

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
        PROCEDURE INPUT (NEWFEATURE,NEWCLASS);

%       INPUTS "NEWFEATURE" AND "NEWCLASS" TO LINKED STORAGE.  IF
%       "NEWFEATURE" IS IN THIS WIND
%       "NEWFEATURE" IS IN THE WINDOW OF AN EXISTING SAMPLE THEN IT IS
%       CLUSTERED WITH THAT SAMPLE; ELSE IT IS PLACED IN A NEW STORAGE B

        INTEGER ARRAY NEWCLASS[1];
        REAL ARRAY NEWFEATURE [1];
        BEGIN  INTEGER CMARK,T,K,L,P;   LABEL XIT;
        SETTREE (NEWCLASS);
            CMARK := 0;
            FOR L:=NLEV    STEP -1 UNTIL 1 DO
            IF 0 NEQ T:=TRAIL[L] THEN
            BEGIN
                IF CMARK > 0 THEN BEGIN
                                  CLINK[CMARK,L]:=TNODE[T,1];
                                  TNODE[T,1] := CMARK
                              END
            ELSE BEGIN
                     P := TNODE[T,1];
                     WHILE P>0 DO
                     BEGIN
                         IF NOT INWINDOW (NEWFEATURE,P)
                         THEN P:=CLINK[P,L]
                         ELSE
                            BEGIN
                             FOR K:=1 STEP 1 UNTIL NCHAN DO
                                FEATURE[P,K]:=(WEIGHT[P] × FEATURE [P,K]
                                   +NEWFEATURE[K])/ (1+WEIGHT[P]);
                             WEIGHT[P] := WEIGHT[P]+1;
                             IF WEIGHT[P]>MAXPOT THEN MAXPOT←WEIGHT[P];
                             GO TO XIT
                         END
                     END;
                     CMARK := NEWCBOX;
                     CLINK[CMARK,L] := TNODE[T,1];
                     TNODE[T,1]:= CMARK;
                     FOR K:=1 STEP 1 UNTIL NCHAN DO
                        FEATURE[CMARK,K] := NEWFEATURE[K];

                     FOR K:=1 STEP 1 UNTIL NLEV    DO
                        CLASS[CMARK,K] := NEWCLASS[K];
                        WEIGHT[CMARK] := 1;
                  END
            END;
 XIT:  END OF INPUT;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       PROCEDURE  DUMPTREE;

       BEGIN INTEGER M,N;
           WRITE (LINE, <"DUMPTREE">);
           FOR M:=1 STEP 1 UNTIL NODES DO
               WRITE (LINE,<20I3>,M      ,
                   FOR N:=1 STEP 1 UNTIL NSONS +1 DO
                        TNODE [M,N]) ;
       END OF DUMPTREE;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    PROCEDURE TREECLIMBER (LOC,LEVEL,KLASS); VALUE LOC,LEVEL,KLASS;

%   THIS PROC. TRAVERSES THE CLASSIFICATION TREE IN END ORDER.  WHEN A
%   NODE IS VISITED, THE CORRESPONDING SUBLIST IS OUTPUT (VIA "OUTPUTSUB-
%   LIST") AND THEN EACH OF THE CHILDREN ARE VISITED.
% GLOBAL DATA:
%     INTEGER ARRAY TNODE [1:NODES,1:NSONS ];

    INTEGER LOC,       % LOCATION OF THE NODE VISITED
            KLASS,     % CLASSIFICATION  OF THE SUBLIST
            LEVEL;     % LEVEL OF THE NODE VISITED

    BEGIN
        INTEGER K,NLOC;
        IF LOC = 0 THEN          % RETURN
        ELSE BEGIN
                IF LEVEL NEQ 0 THEN OUTPUTSUBLIST(LOC,LEVEL,KLASS);
                FOR K := 1 STEP 1  UNTIL NSONS  DO
                IF (NLOC:=TNODE[LOC,1+K]) > 0 THEN
                          TREECLIMBER(NLOC,LEVEL+1,K);
             END
    END OF THE PROCEDURE TREECLIMBER ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
        REAL PROCEDURE
           DISTSQ (XFEATURE,ROW) ;

%       COMPUTES THE SQUARE OF THE EUCLIDEAN DISTANCE BETWEEN
%       XFEATURE AND THE VECTOR FEATURE[ROW,*].

           REAL ARRAY XFEATURE [1];
           INTEGER ROW;

        BEGIN
           REAL    SUM;                 INTEGER K;
           SUM ← 0;
           FOR K←1 STEP 1 UNTIL NCHAN DO
              SUM←SUM+ (XFEATURE[K] - FEATURE[ROW,K])*2;
           DISTSQ ← SUM;
        END OF DISTSQ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        REAL PROCEDURE
           POTENTIAL (XFEATURE,ROW,LEVEL);

%       EVALUATES THE FOLLOWING POTENTIAL FUNCTION
%                   1 + LAMBDA × COUNT[ROW,LEVEL]
%       WEIGHT[ROW]×--------------------------------------------------
%                   1 + ALFA × (FEATURE[ROW,*] - XFEATURE[*] ) * 2

           REAL ARRAY XFEATURE [1 ] ;
           INTEGER     ROW,   LEVEL;

        BEGIN
           POTENTIAL ← WEIGHT[ROW]
                     ×(1+LAMBDA×COUNT[ROW,LEVEL])
                     /(1+ALFA×DISTSQ (XFEATURE,ROW));
        END OF POTENTIAL;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        REAL PROCEDURE  DISCRIMINANT  (XFEATURE,LISTHEAD,LEVEL);

%       EVALUATES THE DISCRIMINANT FUNCTION FOR A SUBCLASS
%        AT THE POINT XFEATURE.

           REAL ARRAY XFEATURE[1];
           INTEGER LISTHEAD,
                   LEVEL;

        BEGIN
           INTEGER P;      REAL SUM;
           SUM←0;    P←LISTHEAD;
           WHILE P>0 DO   % MOVE THROUGH THE LIST ADDING THE POTENTIAL
           BEGIN          % FUNCTION VALUES AT THE POINT XFEATURE
              SUM ← SUM + POTENTIAL (XFEATURE,P,LEVEL) ;
              P ← CLINK[P,LEVEL];
           END;
           DISCRIMINANT ← SUM ;
        END OF DISCRIMINANT;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
        BOOLEAN PROCEDURE
           CLASSIFIEDCORRECTLY (CLOC,LEVEL,PARENT);

   %    DETERMINES WHETHER OR NOT A TRAINING SAMPLE IS CLASSIFIED
   %    CORRECTLY BY THE PRESENT DISCRIMINANT FUNCTIONS.

           INTEGER
              CLOC,          % A LOCATION IN CLIST
              LEVEL,         % THE LEVEL OF CLASSIFICATION
              PARENT;        % THE LOCATION OF THE TREE NODE THAT IS THE
                             % PARENT OF THIS SUBLIST

        BEGIN
           REAL ARRAY  XFEATURE [1:NCHAN];  % THE FEATURE VECTOR TO BE
           INTEGER                          % CLASSIFIED
              K,  S,  BIGCLASS;
           REAL
              BIGVALUE,  D;
           FOR K+1 STEP 1 UNTIL NCHAN  DO
              XFEATURE[K] + FEATURE[CLOC,K];
           BIGVALUE + 0 ;  BIGCLASS + 0 ;
           FOR S+1 STEP 1 UNTIL NSONS DO
             IF TNODE [PARENT,1+S] > 0 THEN
              IF BIGVALUE < D+DISCRIMINANT (XFEATURE,
                            TNODE[TNODE[PARENT,1+S],1],LEVEL)
                 THEN BEGIN
                            BIGVALUE+D;   BIGCLASS+S ;
                      END;
           CLASSIFIEDCORRECTLY + CLASS[CLOC,LEVEL]=BIGCLASS;
        END OF CLASSIFIEDCORRECTLY;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        BOOLEAN PROCEDURE CHECKSUBLIST (LISTHEAD,LEVEL,PARENT);

   %    TRAVERSES A SUBLIST TO DETERMINE WHETHER OR NOT ALL TRAINING
   %    SAMPLES IN THE SUBLIST ARE CORRECTLY CLASSIFIED BY THE
   %    DISCRIMINANT FUNCTIONS.  FOR EACH SAMPLE INCORRECTLY
   %     CLASSIFIED, COUNT IS INCREMENTED IMMEDIATELY.  THIS HAS AN
   %    IMMEDIATE EFFECT ON THE DISCRIMINANT FUNCTION.

           INTEGER
              LISTHEAD, % THE LOCATION OF THE SUBLIST
              LEVEL,    % THE LEVEL OF CLASS TO BE CHECKED
              PARENT;   % THE LOCATION OF THE PARENT TREE NODE
        BEGIN
           BOOLEAN B ;   REAL POT ;
           INTEGER  P;
           B+TRUE;  P+LISTHEAD;
           WHILE P>0 DO  % MOVE THROUGH THE LIST CHECKING EACH ELEMENT
           BEGIN
              IF NOT CLASSIFIEDCORRECTLY (P,LEVEL,PARENT)
              THEN BEGIN    % INCREMENT COUNT
                      B+FALSE;
                      COUNT[P,LEVEL]+COUNT[P,LEVEL]+1;
                      IF POT+WEIGHT[P]x(1+LAMBDAxCOUNT[P,LEVEL])
                            > MAXPOT THEN MAXPOT+POT; % UPDATE MAXPOT

                   END;
             P + CLINK[P,LEVEL]; % MOVE P DOWN LIST
           END;
           CHECKSUBLIST + B;
        END OF CHECKSUBLIST;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        BOOLEAN PROCEDURE TREECHECKER (LOC,LEVEL,PARENT);

   %    TRAVERSES ALL SUBLISTS TO DETERMINE WHETHER OR NOT ALL SUBLISTS
   %    ARE CORRECTLY CLASSIFIED BY THE DISCRIMINAT FUNCTIONS.

           INTEGER
              LOC,     % A TREE NODE LOCATION
              LEVEL,   % A TREE LEVEL
              PARENT;  % THE LOC OF THE PARENT OF NODE AT LOC

        BEGIN
           BOOLEAN B;                INTEGER  S,SON;
                 IF LEVEL =0 THEN B + TRUE
                 ELSE B+CHECKSUBLIST(TNODE[LOC,1],LEVEL,PARENT);
                 FOR S+1 STEP 1 UNTIL NSONS DO
                    IF (0<SON+TNODE[LOC,1+S])
                          THEN B+B AND TREECHECKER (SON,LEVEL+1,LOC);
                 TREECHECKER + B;
        END OF TREECHECKER;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
         PROCEDURE CLASSIFY (MAXLEVEL);

%        CLASSIFIES "NEWFEATURE" LEVEL-BY-LEVEL (UP TO MAXLEVEL),
%        PLACING THE RESULTS IN "NEWCLASS", THE LARGEST DESCRIMINANT
%        DISCRIMINANT FUNCTION VALUE MUST BE GREATER THAN "THRESHOLD"
%        (WHICH IS 1% OF "MAXPOT", THE LARGEST VALUE OF ANY POTENTIAL
%        FUNCTION) ELSE -1 IS ENTERED IN "NEWCLASS" AT THE APPROPRIATE LEV

            INTEGER MAXLEVEL;
%        GLOBAL DATA: NEWFEATURE,NEWCLASS   % NEWFEATURE IS CLASSIFIED
         BEGIN                              % RESULT IN NEWCLASS
            INTEGER  LEVEL,P,BIGCLASS,K,J;
            REAL  BIGVALUE,D ;

            MAXLEVEL ← MIN(MAXLEVEL,NLEV);
            P ← TROOT;
            LEVEL := 0;
            WHILE P>0 AND LEVEL<MAXLEVEL DO
            BEGIN
               LEVEL ← LEVEL + 1 ;
                  BIGVALUE := 0;  BIGCLASS := 0;
               FOR K←1 STEP 1 UNTIL NSONS DO
               IF TNODE[P,1+K]>0 THEN
               BEGIN
                  IF BIGVALUE<D←DISCRIMINANT(NEWFEATURE,
                                 TNODE[TNODE[P,1+K],1],LEVEL)
                  THEN BEGIN
                          BIGVALUE← D;  BIGCLASS← K;
                       END;
               END;
               IF BIGVALUE GEQ THRESHOLD
                  THEN BEGIN
                          NEWCLASS[LEVEL]← BIGCLASS ;
                          P← TNODE[P,1+BIGCLASS];
                       END
               ELSE BEGIN
                       NEWCLASS[LEVEL]← -1 ;          P← 0 ;
                        FOR J:=LEVEL+1 STEP 1 UNTIL NLEV DO
                              NEWCLASS[J]:=0;
                    END;
            END
         END OF CLASSIFY;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%;
         PROCEDURE TRAIN ;

%        EXECUTES "TREECHECKER" AT MOST 20 TIMES OR UNTIL ALL TRAINING
%        DATA ARE CLASSIFIED CORRECTLY BY THE DISCRIMINANT FUNCTIONS.

         BEGIN
            INTEGER I;                  BOOLEAN  OK;
            OK← FALSE ;
            FOR I←0 STEP 1 WHILE I<20 AND NOT OK DO
             BEGIN

               OK← TREECHECKER(TROOT,0,0);
               WRITE(LINE,<"TRAINING WAS ",L5,I10," PASSES USED">,OK,I);
               WRITE(LINE,<"ELAPSED TIME: PR,IO",2R15.4>,
                      TIME(2)/60,TIME(3)/60);
             END;
         END OF TRAIN;

%   CHIMP EXECUTION
           INITIALIZATION;
IPT:     READ (TDFILE,FIRST  +NCHAN+NLEV,DUMMY[*]) [DONE];
         FOR K←1 STEP 1 UNTIL NLEV DO
            NEWCLASS[K] ← DUMMY [FIRST+K-1];
         FOR K←1 STEP 1 UNTIL NCHAN DO
            NEWFEATURE [K] ← DUMMY [SECOND+K-1];
         INPUT (NEWFEATURE,NEWCLASS);
         GO TO IPT;
DONE:     NCENTERS := CAVAIL - 1;
          WRITE(LINE,<"NCENTERS=",I6>,NCENTERS);
          IF DEBUG THEN DUMPDATA;
          TRAIN;
         FOR K←0 STEP 1 UNTIL NCLUST -1  DO
         BEGIN
            FOR M←1 STEP 1 UNTIL NCHAN DO
                NEWFEATURE [M] ← SIG [M,K];
            CLASSIFY (MAXLEVEL);
            FOR M←1 STEP 1 UNTIL NLEV DO
               OUTPUTARRAY [K,M] ← NEWCLASS [M];
         END;
      END OF CHIMP;
```

ORIGINAL PAGE IS
OF POOR QUALITY