Energy, Mines and Resources Canada
Canada Centre for Remote Sensing

Énergie, Mines et Ressources Canada
Centre Canadien de Télédétection

Computer Implementation Of A Four-Dimensional Clustering Algorithm

M. Goldberg
S. Shlien
Data Applications Division

Research Report 76-2
March 1976

Computer Implementation Of A Four-Dimensional Clustering Algorithm

M. Goldberg
S. Shlien
Data Applications Division

Research Report 76-2
March 1976

Canada Centre for Remote Sensing
Department of Energy, Mines and Resources, Ottawa

ABSTRACT

Unsupervised classification of LANDSAT imagery can be accomplished very efficiently by using a four-dimensional histogram in table form. The clustering scheme isolates the peaks in the distribution which are used as cluster centers to classify the remaining vectors in the histogram. The general outlines of this scheme, detailed description and programming implementation are given together with flow diagrams. The program has been implemented on both DEC PDP-10 and DEC PDP-11/40 computers. A discussion of the extension of the algorithm to other than LANDSAT imagery is also given.

RESUME

Il est possible d'assurer tr`es efficacement le classement sans surveillance des images Landsat à l'aide d'un histogramme à quatre dimensions sous forme de tableau. La méthode de regroupement prévoit l'isolation, dans la distribution, des pointes qui sont utilisées comme centres de regroupement pour classer les autres vecteurs dans l'histogramme. Les grandes lignes de cet arrangement, la description détaillée et l'application de la programmation sont fournis et accompagnés d'organigrammes. Le programme a été utilisé avec les ordinateurs DEC PDP - 1- 11/40. On traite également des possibilités d'étendre l'algorithme à d'autres images qu Landsat.

ACKNOWLEDGEMENTS

We wish to thank Dr. Murray Strome and Dr. David Goodenough for their comments and suggestions. We also thank David Belyea and Kevin O'Neill for their help in drawing the flow charts.

TABLE OF CONTENTS

1. INTRODUCTION AND MOTIVATION

The requirement for classifying LANDSAT imagery data into a number of distinct spectral classes occurs in many applications. For example, in agricultural applications, it may be necessary to distinguish the different types of crop. While this function can be performed by manual photointerpretation, automatic classification methods are often required for large areas.

There are two basic approaches to automatic classification, supervised and unsupervised. In the supervised mode, the user specifies certain groups of picture elements (pixels) as training samples which are representative of the classes of interest. The computer then estimates the statistical parameters of the training samples and classifies the remaining pixels into the corresponding classes. Although this approach is conceptually pleasing, there are several practical problems. The most important is that the samples chosen must be truly representative of the classes of interest. A discussion of the supervised method and its restrictions is given by Shlien and Goodenough (1973).

The second approach is based upon the use of a clustering algorithm to identify the separable clusters in the data. It then remains for the user to correlate these clusters with his ground cover classes of interest. Unfortunately, a number of superfluous classes, which are of no interest to a particular user, may be generated. For example, four different classes of water may be distinguished, but the user only requires one water class. The four classes, therefore, must be combined into one class. This decision must be made by the user.

There have been a considerable number of studies on unsupervised classification schemes (see for example, Duda and Hart, 1973). Most schemes have been developed heuristically and generally employ a clustering algorithm to identify the classes. Many of these algorithms are recursive with unknown convergence properties and require considerable computer time and memory.

The clustering algorithm described in this report used the following criteria:

a. The algorithm is intended to be used in a timesharing system and should therefore not require more than 25,000 36-bit words of computer core memory.

b. CCRS also possesses a DEC PDP11/40 minicomputer as part of its IMAGE 100 image analysis system. By increasing the number of disc accesses, the clustering algorithm should be able to run in the 14,000 16-bit words of computer core available.

c. The results are to be displayed on a colour display device, 280,000 pixels at a time. The algorithm should therefore be capable of clustering at least 280,000 pixels at a time.

d. As the system is intended for outside users with little statistical and computer experience, there should be minimal user intervention in the operation of the program.

e. On the other hand, full advantage of the user's particular interest should be made in order to decrease the number of classes generated by the algorithm. This necessitates that the program be interactive.

We have chosen to implement a non-recursive clustering algorithm which satisfies all these requirements and which is based upon the four-dimensional histogram. The table look-up approach can then be used to rapidly generate a colour-coded display of the classification (Shlien and Smith, 1975).

In the next section the concept of a four-dimensional histogram, as it applies to LANDSAT imagery, is introduced. Rules for generating clusters are then developed. The operation of a clustering algorithm based upon these rules is then described.

In Section 3 an implementation of the algorithm is described. This is followed in Section 4, by a short discussion of some practical programming details and of the problems related to user interface. Section 5 follows with a brief description of the implementation of the algorithm on the CCRS IMAGE 100 system. Finally, in Section 6, the problems associated with using the clustering algorithm on data different than that from LANDSAT are examined.

2. CLUSTERING USING FOUR-DIMENSIONAL HISTOGRAMS

The clustering method presented is based upon an examination of the four-dimensional histogram (Goldberg and Shlien, 1975). Each pixel in an image can be represented by an intensity vector (i_1, i_2, i_3, i_4), the components of which correspond to the spectral intensities in LANDSAT bands 4, 5, 6 and 7. In theory, there are (64^4) or about 16 million different possible vectors. In practice, however, many of these vectors never occur in an individual LANDSAT multispectral scanner frame. Shlien and Goodenough (1974) have shown that in most cases 95% of the frame can be represented by 6000 vectors. A four-dimensional histogram is a table listing the frequency of each of the intensity vectors in the image. By means of hashing techniques for storing and accessing the individual vectors, the four-dimensional histogram of 280,000 pixels can be computed in about 2 minutes of computer time (Shlien and Smith, 1975).

Peaks in the histogram are assumed to be associated with specific ground cover classes. The problem of generating classes then corresponds to isolating peaks and delineating their associated clusters. The vectors in the clusters then define the classes. Intuitively, two peaks can always be isolated from one another into separate "islands". As, by definition, peaks are distinct, there is always some height, or threshold, above which the two peaks do not touch. Thus, given this threshold, the peaks can be isolated by ignoring the intensity vectors occurring less frequently than this threshold. Once the peaks are found, it remains to assign the vectors to one or other of the associated clusters.

These intuitive ideas, suggest the following procedure for finding clusters. Some threshold is chosen and the intensity vectors are divided into two sets: those occurring with at least this frequency, and those occurring less frequently. The first set of vectors is divided into clusters of vectors which form "islands", each island corresponding to a peak. The vectors in the second set are then assigned to the closest cluster. In an iterative manner, this procedure can be applied to independently break up any chosen cluster into new clusters.

In order to implement such a clustering technique, two problems must be solved. First, a computationally simple method for delineating the "islands" must be found. The second problem is that of choosing the threshold. From experiments with LANDSAT imagery, efficient heuristic rules have been developed for choosing the threshhold. These rules are detailed in the description of a computer implementation of the clustering algorithm which follows this section.

In order to delineate the "islands", some rules on connectedness between two vectors and between sets of vectors must be defined. A simple definition of connected follows. Two vectors are connected if their intensity values in the four bands do not differ from one another by more than one. Notationally, if the two vectors are (i_1, i_2, i_3, i_4) and (j_1, j_2, j_3, j_4), then the following must hold:

$$i_k - 1 \leq j_k \leq i_k + 1, \text{ for each value of}$$

$$k = 1, \ldots . 4.$$

Thus, for example, (2, 4, 6, 8) is connected to (1, 5, 6, 7), but not to (2, 6, 6, 8).

With this rule of connectedness, the intensity vectors can be grouped into distinct clusters, where the vectors of one cluster are never connected to any vector of another cluster. Alternatively, this implies that there is a chain of pairwise connected vectors from any one vector in a cluster to any other vector in the cluster. These concepts are illustrated

by the following example. Consider the following set of five intensity vectors:

(a) (4, 5, 6, 7)

(b) (5, 6, 7, 8)

(c) (5, 6, 7, 9)

(d) (3, 7, 8, 10)

(e) (1, 1, 1, 1)

Vector (a) is connected to (b), but not to (c). Vector (c) is, however, connected to (b), so that there is a chain or path from (a) to (c). Thus, vectors (a), (b), and (c) are in the same cluster. On the other hand, (d) and (e) are not connected to these vectors, nor are they connected to one another, so that each is in a cluster by itself. Therefore, the five intensity vectors can be grouped into three distinct clusters by using the connectedness rule.

Although conceptually very simple, the implementation of a clustering algorithm based upon connectedness can lead to a substantial amount of calculation. Since each intensity vector must be compared for connectedness with each of the other vectors, as the number of vectors increases, the computer time required to generate clusters increases very quickly, and is soon too great for an interactive system.

To overcome this practical problem some simplifications will be made. A cluster is uniquely defined by a list of the vectors. When a new vector is introduced, in order to determine if it belongs to the cluster, it must be compared for connectedness with each vector of the cluster. Thus, for example, vector (d) must be compared with the three vectors, (a), (b), and (c). We now assume that a cluster is specified by the smallest parallelepiped (for LANDSAT imagery these are four-dimensional rectangles) containing all the vectors in the cluster. The implications of this simplification are first illustrated by an example.

Reconsidering the set of vectors in the example, the parallelepiped representing the cluster with three vectors is seen to be (4-5, 5-6, 6-7, 7-9). This notation refers to the parallelepiped where the lower boundaries in the four dimensions are respectively, 4, 5, 6 and 7, and where the upper bounds are 5, 6, 7 and 9. It is important to note that this definition of a cluster is not unique, and that it defines a cluster with 24 (=2 x 2 x 2 x 3) vectors. In general, a parallelepiped will be denoted by (l_1-u_1, l_2-u_2, l_3-u_3, l_4-u_4), where l_1 refers to the lower bounds and u_1 to the upper bounds. The smallest parallelepiped containing a class of vectors can be easily computed by noting that the lower bound in any dimension is simply the minimum intensity in the corresponding band for all the vectors, and that, likewise, the upper bound is the maximum intensity. Thus, for the example, 7 is the minimum intensity in band 4 and 9 is the maximum so that the corresponding bounds are 7 and 9.

By representing the clusters as rectangular parallelpipeds, a computationally simple criterion of "connected to a cluster" can be given. An intensity vector (i_1, i_2, i_3, i_4) is said to be "connected to a cluster" if the intensities in each band lie within one value of the bounds of the parallelepiped. Using the notation introduced, the following four relations must hold simultaneously:

$$l_k - 1 < i_k < u_k + 1 \quad \text{for all } k,\ k = 1, \ldots 4.$$

Thus, for the example, vector (d), (3, 7, 8, 10), is connected to the cluster represented by (4-5, 5-6, 6-7, 7-9).

When a new vector is added to a cluster, it may be necessary to recalculate the parallelepiped specifying the cluster. This is accomplished by extending the boundaries, if necessary, using the following equations:

$$l_k = \min (l_k, i_k), \text{ for all } k,\ k = 1, \ldots 4$$

$$u_k = \max (u_k, i_k), \text{ for all } k,\ k = 1, \ldots 4$$

Thus, for example, when the vector (3, 7, 8, 10) is added to the cluster, (4-5, 5-6, 6-7, 7-9), the new parallelepiped is (3-5, 5-7, 6-8, 7-10).

As illustrated by this example, a cluster represented by a rectangular parallelepiped results in the formation of much broader clusters. However, the amount of computation required to generate the clusters is greatly reduced since the intensity vectors are now tested for connectedness to the parallelepipeds, and not to each individual vector.

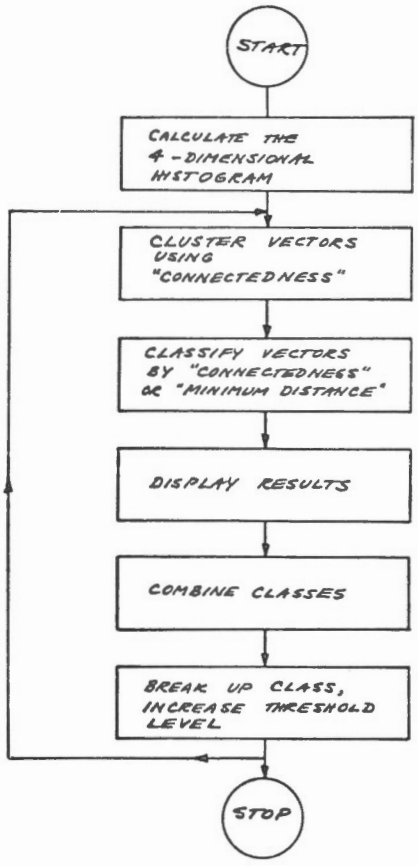


FIGURE 1. CLUSTERING ALGORITHM USING 4-DIMENSIONAL HISTOGRAMS.

An algorithm incorporating these ideas is now described and is sketched in FIGURE 1. The first step is to calculate the four-dimensional histogram. In order to isolate the peaks, some initial threshold value must be chosen. This divides the vectors into two sets: those occurring with a frequency at least as great as the threshold value and those occurring less frequently than the threshold value. The vectors occurring with at least the threshold frequency are then grouped into separate clusters by using the "connected to a cluster" rule. The remaining vectors, that is, those occurring less frequently than the threshold, are then assigned to some cluster by the same criterion. If this fails to classify a vector, then the vector is assigned to the "closest" cluster. The measure of closeness used is the Euclidean distance of a vector to the mean vector of the cluster. Thus, the final shape of the clusters are not necessarily rectangular parallelepipeds.

The results are now displayed in a colour-coded fashion. By making use of the ground truth, the user must now decide whether to combine clusters or to further break them up. In the latter case, the vectors of the chosen cluster are tagged and the clustering algorithm is repeated for these vectors. The vectors of the chosen cluster were identified as one cluster at a previous iteration by using some threshold value. In order to isolate distinct peaks in the chosen cluster, a higher threshold must be chosen. New clusters corresponding to these peaks can then be identified and displayed. This process can be repeated at the user's discretion.

It is clear that the choice of the threshold is important for an efficient computer implementation. Heuristic rules have been developed for choosing the thresholds. These rules and a more detailed exposition of the algorithm are described in the following section.

3. COMPUTER IMPLEMENTATION OF THE CLUSTERING ALGORITHM

The algorithm described in Section 2 was first implemented on a DEC-10 System and used in conjunction with the Bendix Corporation Multispectral Analyzer Display (MAD). After considerable testing with LANDSAT imagery, certain heuristic rules were developed which considerably increased the speed of the program. These rules and our implementation of the clustering algorithm are described in this section. Detailed flow charts of the individual subroutines and information about the practical computer implementation are presented in the

Appendix.

It is clear that a single threshold value will not be sufficient to isolate all of the peaks and the corresponding clusters. The capability of making several passes on the vectors at different threshold values has, therefore, been incorporated into the program. This ability is used in two distinct ways. First of all, if the threshold is set at too low a value, then a cluster chosen by the user will not be broken up. The program detects this situation, raises the threshold level, and attempts to isolate at least two peaks and the corresponding clusters. The threshold is raised in stages until the cluster is broken, or until the presence of only one peak, and thus, only one cluster is detected.

In the second and opposite situation, the threshold may be set too high. Now, some of the peaks, and their corresponding clusters may be missed. These peaks occur among those vectors that are not assigned to any cluster by using the "connected to a cluster" rule. These vectors are now recycled at some lower threshold than that previously set and new peaks and clusters may be differentiated. The vectors which are still not "connected to any cluster", are now classified by the minimum distance rule. Some of the clusters isolated by recycling are often of particular interest to the user. A penalty is paid, for recycling as many clusters which are of little or no interest to the user may be generated. The user must then decide whether they are to be retained or ignored.

Aside from these important modifications, the computer implementation follows closely the description of the clustering algorithm detailed in the previous section. The step-by-step operation of the computer implementation of the algorithm is now given, as well as the rules for choosing the thresholds. These steps are summarized in FIGURE 2.

Step 1: The first step is to calculate the four-dimensional histogram for the area of interest. The procedure used, based upon hashing, is described by Shlien and Smith (1975).

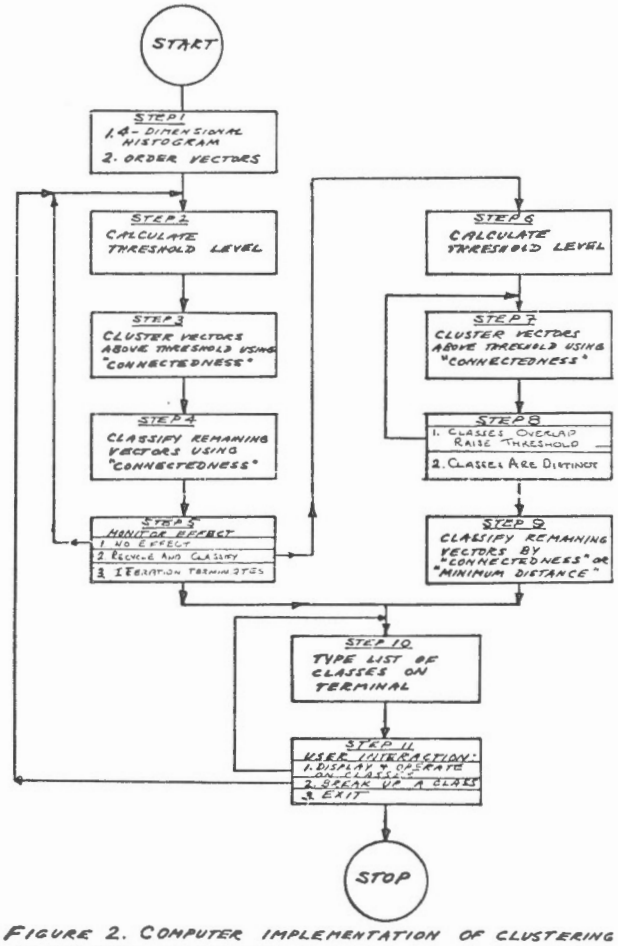


FIGURE 2. COMPUTER IMPLEMENTATION OF CLUSTERING ALGORITHM. FLOW DIAGRAM OF CONTROLLING PROGRAMME.

Step 2: A threshold level must now be chosen. Initially, the threshold is set to the mean frequency of the vectors. From experience, this usually separates the broad types of classes, such as water and land. When a class chosen by the user is being split, the new threshold level (LVLNEW) is heuristically chosen as a function of the old threshold (LVLOLD) and the maximum frequency (LVLMAX) of the class; namely,

$$LVLNEW = LVLOLD + \tfrac{1}{4}\,(LVLMAX - LVLOLD).$$

Step 3: By means of the "connected to a cluster" rule, the vectors occurring with the threshold frequency are now clustered. Overlapping

clusters are merged, so that the resultant clusters are distinct.

Step 4: The remaining vectors are now classified by using the "connected to a cluster" rule, but the clusters are not expanded. Because the boundaries of the classes are not expanded, a vector may lie on the boundary between two clusters, and in this case the vector is arbitrarily assigned to the first cluster. If a vector occurring less frequently than the threshold is not connected to any cluster, then it is left as unclassified.

Step 5: (i) If the chosen cluster has not been split up, the threshold is raised and a new attempt to break up the cluster is made. This is realized by setting LVLOLD to the current value of LVLNEW and then returning to Step 2.

(ii) If some of the vectors have been left unclassified, the program proceeds to recycle these vectors in Step 6.

(iii) If either the threshold level is equal to the maximum frequency, or the chosen class has been successfully broken into two or more classes, the present iteration of the algorithm must terminate, and the user is informed of the results.

Step 6: A threshold level for the unclassified vectors is chosen. After some experimentation the following heuristic function was chosen: LVLNEW = MIN (LVLOLD, 3/4 (LVLMAX)), where LVLMAX is with respect to the unclassified vectors.

Step 7: Step 3 is repeated.

Step 8: The clusters are checked for overlap, in which case the threshold level is raised and Step 7 is repeated.

Step 9: The remaining unclassified vectors are classified according to the "connected to a cluster" rule; those remaining are assigned to clusters by the minimum distance rule.

Step 10: The complete list of clusters is typed out on the tele-type. The clusters are identified by a number, the number of different vectors in the cluster, the total number of pixels, and by the mean vector of the cluster.

Step 11: This step is under complete user control. He may choose to stop the program; display the results in colour coded fashion; combine two or more clusters and display the combination; re-assign the vectors of a cluster amongst the other clusters by using the minimum distance rule; or finally choose any cluster to be further broken up.

These are the basic steps in the computer implementation of the clustering algorithm. A more detailed exposition, together with flow charts, is given in the Appendix.

4. PROGRAM STRUCTURE AND USER INTERFACE

A short discussion of some of the practical details and problems associated with the computer implementation of the clustering algorithm, and its interface with the user follows. Except for the bit manipulation routines, the program is written in standard FORTRAN. Approximately 25,000 36-bit words of computer core are required to run the program, 12,000 of these to store the vectors. Two words are allocated for each vector: one containing intensity information, the other containing auxiliary information, cluster number and frequency value. Since for each interation the vectors are processed several times, but only one at a time, the space required can be reduced considerably by using

disc storage. Adaptation to other computers, therefore, poses few problems. The number of distinct clusters present at any one time is limited to 30 in the present implementation. Since clusters can be combined or re-assigned at each iteration, this limitation has not caused user difficulties.

User interface is of primary importance as the program is designed for users with little computer and statistical experience. In FIGURE 3 we reproduce a sample of the computer dialogue with the user for the PDP-10 implementation. The portion of the scene from which the histogram is to be calculated is first specified. The vectors of the histogram are clustered and the results are presented to the user for decisions on subsequent actions. The colour number corresponds to the colour assigned the class on the Multispectral Analyzer display. This permits the user to relate classes found by the clustering program with the colour coded display. In the example, Class 1 corresponds to water and Class 2 to land features.

At this stage the user has chosen to break up Class 2. As the program searches for clusters, the different threshold levels are typed out. The higher the values, the greater is the relative degree of overlap between the different clusters. At the end of the iteration the results are again presented and the user has a number of choices as illustrated in FIGURE 3.

BREAK-Attempt to break up specified class.

COMBINE-Combine up to 15 classes into one new class.

DISPLAY-Display results in colour coded form on the MAD.

EXIT-Stop the program.

INFOR-Type out statistical information for the chosen class.

REASSIGN-The vectors of the chosen classes are reassigned on an individual basis to the closest class as measured by the Euclidean distance.

START OVER-The program will start over from the beginning.

AUXILIARY-Reserved for research programs.

The dialogue has purposely been made very simple and does not require the user to supply any parameters other than the class numbers.

In a system with a large amount of user iteration, fast computer response is important. Typical computer time (CPU) and waiting time required on the CCRS timesharing system are as follows. With vector storage by hashing we find that approximately two minutes of CPU time and three minutes waiting time are required to form the histogram of 280,000 pixels. By sampling the pixels, this time could be reduced. The clustering part of the program (Step 2 through to Step 10 of FIGURE 2) requires at most 10 seconds of CPU time and 4 seconds waiting time for the last iteration. In a typical case, about 10 iterations are required for a total time of about 50 seconds CPU time a 4 minutes waiting time. Displaying the 280,000 pixels corresponding to one image of the Multispectral Analyzer Display (MAD) requires 150 seconds of CPU time. The display can be speeded up by presenting only a magnified portion of the image. This is accomplished by simply repeating the pixels in both dimensions.

In practice, the intermediate classification is usually displayed in a two-fold or three-fold magnification, which decreases the corresponding times by 4 and 9 respectively. From start to finish, a typical user can generate classification of the 28,000 pixels in about an hour using 10 minutes of CPU time. Most of this time is spent in Step 11, in which the user experiments by breaking up classes, reassigning classes, or combining classes and displaying the results. Examples of classifications, and of accuracies achieved with this program are described in Goldberg, Goodenough and Shlien (1975).

5. IMPLEMENTATION OF CLUSTERING ALGORITHM ON IMAGE 100

The clustering algorithm as implemented on the DEC PDP-10 timesharing system has been transferred to the CCRS Image-100 system.

A number of changes were required to interface the program with the Image-100 system. The clustering portion of the program requires essentially the same time as on the PDP-10. By making use of the special purpose hardware of the Image-100, the results can be displayed much more rapidly than on the MAD.

The CCRS Image-100 classification system includes a PDP-11/40 minicomputer with 72,000 16-bit words of computer core and a number of special hardware features which are used to speed up both the classification and the displays of the results in a colour coded fashion on a television monitor. A more detailed description is given by Goodenough (1975).

Under the Disc Operating System (DOS) of the PDP-11/40 only 14,000 words of core are available for programs. Because of this restriction a number of changes to the clustering program are required. Only the 2000 most frequently occurring vectors are now used to generate the clusters. The remaining vectors, of which there may be any number, are then classified one at a time using the "minimum distance to class mean" rule. This restriction is not too serious, since the clustering algorithm would rarely use the less frequent vectors to generate the clusters. In order to further reduce the core requirements the program itself is divided into three overlays, each one of which is called once per iteration. One overlay reads in the vectors; the second overlay generates the clusters; and the third overlay communicates with the user.

Because of the availability of 8-bit byte instructions on the PDP-11/40, the numerous bit manipulations required on the PDP-10 are eliminated. As a result, the algorithm requires essentially the same amount of time on the PDP-11/40 as on the PDP-10. Display of the results is done using the hardware capabilities of the Image-100, reducing the display generation time to a range of 3 to 50 seconds per class.

Except for these changes, the implementations on the two computers are essentially identical. Any modifications made to one program are easily transferred to the other.

6. EXTENSION OF THE CLUSTERING ALGORITHM TO IMAGERY OF DIFFERENT FORMAT

The clustering algorithm has been devised using the format of LANDSAT multispectral imagery originally developed for computer compatible tape (CCT). In this format, the CCT contains four channels of uncorrected 6-bit (0-63) data. The standard format (NEW FORMAT) now offered by CCRS provides the data in linearized, radiometrically corrected 8-bit (0-255) form. Other data may consist of a greater number of channels. There are certain problems associated with extending the clustering algorithm to these different data. These are related to the calculation of the multidimensional histogram, to the choice of an appropriate connectedness rule, and to the selection of threshold levels.

For LANDSAT imagery it has been shown that for scenes devoid of snow, a histogram of 6000 vectors will cover at least 95% of the image (Shlien and Goodenough, 1970). In other types of data, the number of vectors required to achieve a similar coverage may be considerably higher. For LANDSAT imagery in 8-bit format the corresponding number of independent vectors in the histogram is of the order of 1.5×10^6 ($4^4 \times 6000$), a number which is clearly unmanageable. One method of treating these vectors, which can be also applied to other sets of data, is to simply drop the least two significant bits, thus reducing the number of intensity levels to 64. This technique has been tested for one agricultural area and yields essentially the same results.

As the number of channels is increased, the number of vectors required to achieve a 95% coverage will increase very rapidly. Shlien (1975) has shown that for 5-channel data about 15,000 vectors are required, while for 6-channel data, the number is about 130,000. Given the core size limitation, it is, therefore, still possible to treat 5-channel data but not 6-channel data.

Another problem is the selection of a suitable rule of connectedness to be applied to different data sets. For example, for six-bit LANDSAT imagery, experiments with different rules, such as using a distance of 2 rather than 1 for connectedness were conducted.

This distance function results in the generation of only two classes, water and land. By defining a cluster as a string of vectors which are pairwise connected, about five times as much computer time is required, although smaller clusters results. This approach is especially useful for water classification, where the number of vectors per cluster is very small, and where the clusters are grouped together very closely. Other computationally more demanding distance functions, such as the Euclidean distance, can also be used and may yield better results for some applications.

Different rules for choosing the threshold levels in Steps 2 and 6 of FIGURE 2 were tried on LANDSAT imagery. The final classification results were essentially equivalent, but differed in the number of iterations required. The threshold rules chosen will, in most cases, minimize the number of iterations. If other data sets are used, it may be necessary to experiment in order to find the most appropriate threshold rule.

As can be seen from this discussion, the extension of the clustering algorithm to different sets of data is not straightforward, but depends closely upon the particular structure of the data.

It is probably possible to apply the algorithm to data with resolution greater than 64, but storage problems arise in the calculation of the histogram for data with six or more channels.

7. CONCLUSIONS

A clustering technique for the unsupervised classification of LANDSAT imagery has been described. The computer implementations of this technique on a DEC PDP-10 computer and the Image-100, using a DEC PDP-11/40 computer, have been given. The program operates in an iterative manner, and because of its speed, is well suited for use in an interactive environment. Because all the vectors of the histogram do not have to be in core, the program can be easily adapted to run in computers with small core size, without a serious loss in speed. Extension to different data types may be possible, but this requires further investigation.

REFERENCES

Duda, O. and P. E. Hart (1973) Pattern Classification and Scene Analysis, John Wiley and Sons.

Goldberg, M., and S. Shlien (1976) A Four Dimensional Histogram Approach to the Clustering of LANDSAT Data, Canadian Journal of Remote Sensing 2 (1) 1-11.

Goldberg, M., D. Goodenough, and S. Shlien (1975) Classification Methods and Error Estimation for Multispectral Scanner Data, Proc. 3rd. Canadian Symposium on Remote Sensing, Edmonton, Sept., 125-143.

Goodenough, D. (1975) IMAGE 100 Classification Methods for ERTS Scanner Data, Canadian Journal of Remote Sensing 2 (1) 18-29.

Goodenough, D., S. Shlien, A. Smith, N. Davis, H. Edel, R. Fawcett, G. Wayne. (1973) The Multispectral Analyzer Display (MAD) User's Manual, CCRS Technical Note 73-8, Ottawa, 54 p.

Shlien, S. (1975) Practical Aspects Related to Automated Classification of ERTS-1 Imagery Using Look-up Tables, CCRS Research Report 75-2, Ottawa, 15 p.

Shlien, S. and D. Goodenough (1973) Automatic Interpretation of ERTS-A Imagery Using the Maximum Likelihood Decision Rule, CCRS Research Report 73-2, Ottawa, 24 p.

Shlien, S., and D. Goodenough, (1974), Quantitative Methods of Processsing the Information Content of ERTS Imagery for Terrain Classification. Proc. 2nd Canadian Symposium on Remote Sensing, Guelph, Ontario, April, 237-265.

Shlien, S. and A. Smith (1975) A Rapid Method to Generate Spectral Theme Classification of LANDSAT Imagery, Remote Sensing of Environment 4 (1) 67-77.

```
NUMBER OF PIXELS ACROSS,NUMBER OF LINES DOWN = 250 250
STARTING LINE,STARTING PIXEL  = 1  1
LINE DECIMATION M OUT OF N    M,N= 1  2
PIXEL DECIMATION I OUT OF J    I,J= 1  3


                  UNSUPR VERSION 7.1
THRESHOLD=  9 MAXIMUM FREQUENCY=  393 FOR CLASS  0
1063 VECTORS OCCUR AT LEAST   9 TIMES IN CLASS 0
   PIXELS VECTORS COLOUR           PIXELS VECTORS COLOUR
 1    390     158     -1         2  39810    4444     -1

(B)REAK, (C)OMBINE, (D)ISPLAY, (E)XIT,(I)NFOR,
(R)EASSIGN, (S)TART OVER ,OR A(U)XILIARY? =B
WHICH CLASS =2
THRESHOLD= 107 MAXIMUM FREQUENCY= 393 FOR CLASS 2
 23 VECTORS OCCUR AT LEAST 107 TIMES   IN CLASS 2
THRESHOLD=   9 MAXIMUM FREQUENCY=  85 FOR CLASS 2
 797 VECTORS OCCUR AT LEAST  9 TIMES   IN CLASS 2
  52 VECTORS OCCUR AT LEAST 47 TIMES   IN CLASS 2
  14 VECTORS OCCUR AT LEAST 66 TIMES   IN CLASS 2
    PIXELS VECTORS COLOUR         PIXELS VECTORS COLOUR
1      390     158     -1       2   8444     435     -1
3     4752     208     -1       4    572     173     -1
5     6895    1110     -1       6   2832     393     -1
7     3051     401     -1       8   5130     537     -1
9     3419     329     -1      10   4715     858     -1

(B)REAK, (C)OMBINE, (D)ISPLAY (E)XIT,(I)NFOR,
(R)EASSIGN, (S)TART OVER ,OR A(U)XILIARY? = I
WHICH CLASS? 1
CLASS= 1  MEAN= 12.4  9.1  9.2  2.0  DETERMINANT=  0.78
PIXELS=  390  VECTORS= 158  LEVEL=   9
COVARIANCE =   0.66
               0.40     0.82
               0.04     0.21    4.06
              -0.06     0.06    2.14    1.66
CLASSES IN SAME BRANCH ARE= 1  2  3  4  5  6  7  8  9  10
(B)REAK, (C)OMBINE, (D)ISPLAY, (E)XIT,(I)NFOR,
(R)EASSIGN, (S)TART OVER ,OR A(U)XILIARY? =R
ENTER UP TO 15 CLASSES TO BE REASSIGNED 3
    PIXELS VECTORS COLOUR           PIXELS VECTORS COLOUR
1      390     158     -1         2    8444     435     -1
3      572     173     -1         4    6895    1110     -1
5     4074     446     -1         6    4759     465     -1
7     5513     548     -1         8    4838     409     -1
9     4715     858     -1

(B)REAK, (C)OMBINE, (D)ISPLAY, (E)XIT,(I)NFOR,
(R)EASSIGN, (S)TART OVER ,OR A(U)XILIARY? =E
```

FIGURE 3: Example of the computer dialogue with the user of the clustering program. User responses are underlined. After specifying area to be clustered, the first pass of algorithm finds two classes. The user has chosen to break up class 2 and the class has been broken into 9 classes. By typing "I" and "1" the complete statistics of class 1 is presented. The user then has that class 3 re-assigned on an individual vector basis.

APPENDIX

In this appendix, the flow diagrams of the implementation of the clustering algorithm on the CCRS DEC-10 KI computer are presented. This program, except for bit manipulations, is written in FORTRAN. As actually written, the program requires 20,000 words of computer core (36-bit words). Of this, 12,000 words are for the 6000 vectors, two words per vector. As the vectors are only processed one at a time, and only several times per iteration, the size can be substantially reduced by reading in the vectors from a disc storage. Because development is continuing, the program has been constructed in a very modular and flexible form. Changes and improvements to the algorithm are easily implemented. As a result, the program code is probably not as efficient as possible.

The entire program is built around the main routine, called PIPED. This routine controls the flow of information between the different subroutnes. A list of the different arrays, parameters, and flags, used in the program are first presented in Table A. 1.

FIGURES A. 1 - A. 3 illustrate the flow of the program between the different subroutines. The action taken by the subroutine is typed out. The initial iteration is described in FIGURE A. 1. The procedure for breaking up a class is given in FIGURE A. 2. Combining and re-assigning classes are illustrated in FIGURE A. 3.

Table A. 1 Non-local Variables Used in Program

NAME (DIMENSION)	DESCRIPTION
BOUND (8, 30)	Class or parallelepiped boundaries.
IICUR	Flag used in subroutine TREE.
I1234 (4)	Intensity values of current vectors
ICHECK	Flag set by subroutine CHECK
IEFCT	Flag set by subroutine EFFECT
IPROC	Flag set in subroutine OUT
IRESTA	Flag set in subroutine IN
ISW1	Flag read by subroutine RECT and set by subroutine KLASS, TREE
KLASS	Class of current vector
LEVEL (31)	Stores significance levels at which classes are created.
LMAX	Maximum frequency of occurrence of the vectors of class MXCLAS.
LVL	Current threshold
LVLMIN	Minimum threshold value permitted
MXCLAS	The class currently being broken up, initially set to ∅
NCLASS	The current number of classes isolated
NOLD	The intermediate number of classes found before the current iteration.
NOLDT	The intermediate number of classes found before recycling the unclassified vectors
NOHIST (31)	Number of pixels in each class
NOVEC	The total number of vectors
NOVECT (31)	The number of vectors in each class
RMEAN (4, 31)	The mean vector in each class
VECT1 (6000)	The intensity values of the vectors
VECT2 (6000)	The corresponding class and frequency of each vector

```
*MAIN*          UNSUPR VERSION 7.1      ITRACE= 4
*LEVEL*     THRESHOLD=   9 MAXIMUM FREQUENCY= 393 FOR CLASS  0
*RECT*      CLASS    1 FORMED, STARTING AT  12  8  8  1
*RECT*      CLASS    2 FORMED, STARTING AT  14 10 28 22
*RECT*      CLASS    3 FORMED, STARTING AT  14 13 14  6
*RECT*      CLASS    4 FORMED, STARTING AT  15 13 24 16
*OVRLAP*    CLASSES  2 AND  4 OVERLAP
*OVRLAP*    CLASSES  2 AND  3 OVERLAP
*TREE*      1063 VECTORS OCCUR AT LEAST    9 TIMES IN CLASS 0
*TREE*      CLASS          BOUNDARIES
*TREE*        1     12 13    8 10    8  9    1  2
*TREE*        2     14 22   10 26   14 37    5 30
*KLASS*
*EFFECT*    NO EFFECT
*UNCLA*
*MOVE*      ELIMINATED CLASS  0
*OUT*       PIXELS VECTORS COLOUR          PIXELS VECTORS COLOUR
*OUT*    1     390     158     -1        2   39810    4444     -1
*STA*       CLASS=  1 MEAN=  12.4   9.1   9.2   2.0  DET=   0.78
*STA*       COVARIANCE=   0.658   0.401   0.816   0.039   0.213   4.060  -0.060   0.059   2.142   1.658
*STA*       CLASS=  2 MEAN=  17.6  17.4  25.7  16.6  DET= 193.81
*STA*       COVARIANCE=   3.454   4.837   9.606   5.255   3.670  29.458   4.157   1.157  30.632  33.830
*STA*       CLASS=  0 MEAN=  17.5  17.4  25.5  16.5  DET= 218.27
*STA*       COVARIANCE=   3.689   5.212  10.186   6.033   4.954  31.820   4.849   2.312  32.664  35.562
```

FIGURE A. 1: Sub-routines called in the initial iteration of the clustering algorithm.

```
*COMBIN*    COMBINING CLASSES   2  5

*REASS*     REASSIGNING CLASSES   6
*UNCLA*
*MOVE*      ELIMINATED CLASS  6
*OUT*       PIXELS VECTORS COLOUR          PIXELS VECTORS COLOUR
*OUT*    1    16336    1713     -1        2     390     158     -1
*OUT*    3     6107     331     -1        4     572     173     -1
*OUT*    5     2847     405     -1        6    5130     537     -1
*OUT*    7     4103     427     -1        8    4715     858     -1
*STA*       CLASS=  1 MEAN=  17.2  16.1  24.6  15.8  DET=  95.98
*STA*       COVARIANCE=   2.092   1.448   3.060   7.179  -0.126  55.462   7.158  -1.042  59.598  65.570
*STA*       CLASS=  2 MEAN=  12.4   9.1   9.2   2.0  DET=   0.78
*STA*       COVARIANCE=   0.658   0.401   0.816   0.039   0.213   4.060  -0.060   0.059   2.142   1.658
*STA*       CLASS=  3 MEAN=  17.0  17.3  24.5  15.1  DET=   0.64
*STA*       COVARIANCE=   0.537   0.275   0.750  -0.002  -0.372   2.371  -0.130  -0.609   2.441   3.504
*STA*       CLASS=  4 MEAN=  14.4  12.6  14.7   6.1  DET=   0.93
*STA*       COVARIANCE=   0.783   1.117   2.846  -0.407  -0.956   1.639   0.060   0.080   0.658   1.081
*STA*       CLASS=  5 MEAN=  15.4  13.9  25.6  17.9  DET=   1.96
*STA*       COVARIANCE=   0.762   0.869   2.169  -0.119  -0.787   2.348  -0.400  -1.323   2.558   4.021
*STA*       CLASS=  6 MEAN=  17.4  16.7  28.7  20.3  DET=   2.80
*STA*       COVARIANCE=   1.695   2.141   4.205   0.336   0.159   1.167  -0.600  -1.234   0.576   1.760
*STA*       CLASS=  7 MEAN=  18.6  20.1  25.3  15.2  DET=   1.50

*STA*       COVARIANCE=   0.859   0.431   1.036   0.066   0.082   2.506  -0.074  -0.150   2.510   3.424
*STA*       CLASS=  8 MEAN=  20.8  23.4  29.4  19.0  DET=   7.95
*STA*       COVARIANCE=   1.747   2.105   4.136   1.357   2.134   4.033   1.032   1.608   3.986   5.039
```

FIGURE A. 2: Sub-routines called for combining and reassigning classes.

```
*MAIN*      BREAKING UP CLASS  2
*LEVEL*     THRESHOLD= 107 MAXIMUM FREQUENCY= 393 FOR CLASS  2
*RECT*      CLASS   3 FORMED, STARTING AT  16 15 16  7
*RECT*      CLASS   4 FORMED, STARTING AT  17 17 24 15
*TREE*        23 VECTORS OCCUR AT LEAST  107 TIMES IN CLASS 2
*TREE*      CLASS          BOUNDARIES
*TREE*        2     14 22   10 26   14 37    5 30
*TREE*        3     16 18   15 18   16 20    7 10
*TREE*        4     17 17   17 17   24 26   15 17
*KLASS*
*EFFECT*    RECYCLING CLASS  2
*LEVEL*     THRESHOLD=   9 MAXIMUM FREQUENCY=  85 FOR CLASS  2
*RECT*      CLASS   5 FORMED, STARTING AT  14 10 28 22
*RECT*      CLASS   6 FORMED, STARTING AT  14 13 14  6
*RECT*      CLASS   7 FORMED, STARTING AT  15 13 24 16
*OVRLAP*    CLASSES  5 AND  7 OVERLAP
*RECT*      CLASS   7 FORMED, STARTING AT  16 15 20 12
*OVRLAP*    CLASSES  5 AND  7 OVERLAP
*TREE*       797 VECTORS OCCUR AT LEAST    9 TIMES IN CLASS 2
*TREE*      CLASS          BOUNDARIES
*TREE*        4     17 17   17 17   24 26   15 17
*TREE*        5     14 22   10 26   20 37   10 30
*TREE*        6     14 16   13 15   14 15    5  7
*CHECK*    CLASSES  3 AND  5 OVERLAP
*CHECK*    CLASSES  3 AND  6 OVERLAP
*RECT*      CLASS   5 FORMED, STARTING AT  15 14 14  6
*RECT*      CLASS   6 FORMED, STARTING AT  16 15 24 16
*RECT*      CLASS   7 FORMED, STARTING AT  17 16 28 20
*RECT*      CLASS   8 FORMED, STARTING AT  17 17 22 12
*OVRLAP*    CLASSES  6 AND  7 OVERLAP
*RECT*      CLASS   8 FORMED, STARTING AT  18 15 33 25
*OVRLAP*    CLASSES  6 AND  7 OVERLAP
*TREE*        52 VECTORS OCCUR AT LEAST   47 TIMES IN CLASS 2
*TREE*      CLASS          BOUNDARIES
*TREE*        4     17 17   17 17   24 26   15 17
*TREE*        5     15 15   14 14   14 14    6  6
*TREE*        6     16 21   15 24   22 29   12 20
*TREE*        7     18 19   15 17   32 35   23 27
*CHECK*    CLASSES  4 AND  6 OVERLAP
*RECT*      CLASS   7 FORMED, STARTING AT  16 15 24 16
*RECT*      CLASS   8 FORMED, STARTING AT  17 17 23 13
*RECT*      CLASS   9 FORMED, STARTING AT  17 17 27 19
*RECT*      CLASS  10 FORMED, STARTING AT  18 20 26 16
*RECT*      CLASS  11 FORMED, STARTING AT  20 23 28 17
*TREE*        14 VECTORS OCCUR AT LEAST   66 TIMES IN CLASS 2
*TREE*      CLASS          BOUNDARIES
*TREE*        4     17 17   17 17   24 26   15 17
*TREE*        5     15 15   14 14   14 14    6  6
*TREE*        6     18 19   15 17   32 35   23 27
*TREE*        7     16 16   15 15   24 26   16 18
*TREE*        8     17 17   17 17   23 23   13 13
*TREE*        9     17 17   17 17   27 27   19 19
*TREE*       10     18 18   20 20   26 26   16 16
*TREE*       11     20 21   23 24   28 29   17 18
*KLASS*
*UNCLA*
*MOVE*      ELIMINATED CLASS  2
*OUT*       PIXELS VECTORS COLOUR          PIXELS VECTORS COLOUR
*OUT*    1     390     158     -1        2    8444     435     -1
*OUT*    3    4752     208     -1        4     572     173     -1
*OUT*    5    6895    1110     -1        6    2832     393     -1
*OUT*    7    3051     401     -1        8    5130     537     -1
*OUT*    9    3419     329     -1       10    4715     858     -1
*STA*       CLASS=  2 MEAN=  16.4  16.3  18.0   8.7  DET=   0.53
*STA*       COVARIANCE=   0.777   0.818   1.578   0.768   1.115   2.753   0.452   0.682   1.722   1.587
*STA*       CLASS=  3 MEAN=  16.9  17.1  25.0  15.9  DET=   0.25
*STA*       COVARIANCE=   0.509   0.229   0.606   0.162   0.006   1.457   0.058  -0.132   1.190   1.713
*STA*       CLASS=  4 MEAN=  14.4  12.6  14.7   6.1  DET=   0.93
*STA*       COVARIANCE=   0.783   1.117   2.846  -0.407  -0.956   1.639   0.060   0.080   0.658   1.681
*STA*       CLASS=  5 MEAN=  18.2  16.0  33.0  25.0  DET=  12.34
*STA*       COVARIANCE=   2.062   2.534   5.042   1.160   0.472   3.316   0.098  -0.992   2.748   4.159
*STA*       CLASS=  6 MEAN=  15.4  13.9  25.6  17.9  DET=   1.73
*STA*       COVARIANCE=   0.752   0.847   2.127  -0.152  -0.857   2.272  -0.451  -1.427   2.451   3.861
*STA*       CLASS=  7 MEAN=  17.2  17.6  22.4  12.5  DET=   1.30
*STA*       COVARIANCE=   1.298   1.627   3.387   0.157   0.461   1.090  -0.284  -0.312   0.378   0.950
*STA*       CLASS=  8 MEAN=  17.4  16.7  28.7  20.3  DET=   2.80
*STA*       COVARIANCE=   1.695   2.141   4.205   0.336   0.159   1.167  -0.600  -1.234   0.576   1.760
*STA*       CLASS=  9 MEAN=  18.6  20.2  25.8  15.8  DET=   0.77
*STA*       COVARIANCE=   0.918   0.495   1.137   0.043  -0.096   1.267  -0.136  -0.405   1.076   1.785
*STA*       CLASS= 10 MEAN=  20.8  23.4  29.4  19.0  DET=   7.95
*STA*       COVARIANCE=   1.747   2.105   4.136   1.357   2.134   4.033   1.032   1.608   3.986   5.039
```

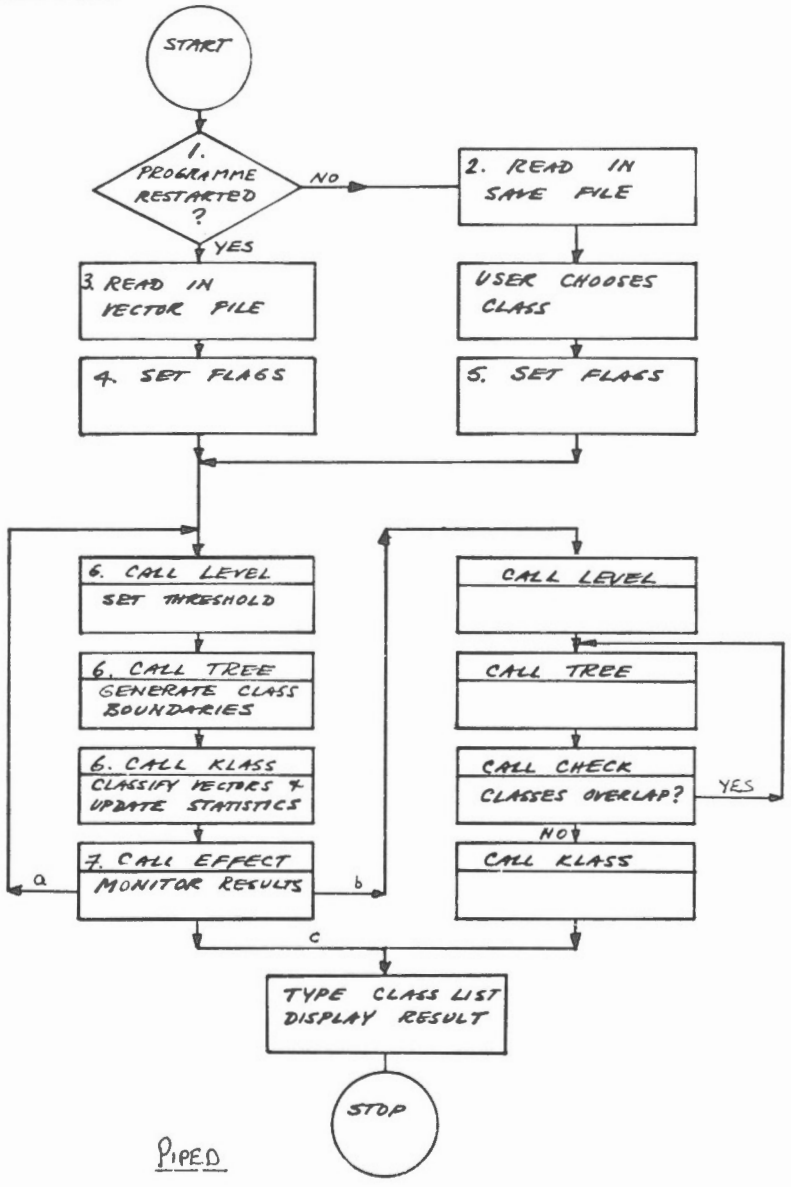
FIGURE A. 3: Action of the program in breaking up a class.

PIPED: This is the main routine, the role of which is to direct the flow of information between the various subroutines. A list of the non-local variables is presented in Table A. 1. We note that the convention "Implicit Integer (A-Q, S-Z)" is being used. The arrays VECT1 and VECT2 contain the information about the vectors. The first word, VECT1 contains the four intensity values for the four LANDSAT bands; the second, VECT2, contains the auxiliary information - class number, frequency value, and a block reserved for future use.

1. Subroutine IN determines if the program is being started for the first time or being restarted from an earlier session and returns this information in IRESTA.

2. The program is being continued from the point before the intermediate results are displayed. The saved arrays are then read back in from disc, where they have been saved as the file CLSSAV.

3. The program is being started for the first time and the histogram list formed by the program 'VECKOU' is read in.



4. Two flags are set: IEFCT = ∅ and MXCLAS = ∅.

5. Two flags are set: IEFCT = ∅ and MXCLAS is set to the class chosen by the user.

6. These are the three subroutines where the actual clustering and classification is performed.

7. EFFECT returns one of the three values in IEFCT, depending upon the results of the clustering.

   (a) IEFCT = ∅: If all the vectors of MXCLAS have been reclassified into a single new class, then there has been no effect. The threshold is raised and another attempt is made.

   (b) IEFCT = 1: The class has been only partially split. The unclassified vectors are recycled through the classifier a second time. The unclassified vectors or residue are those vectors which cannot be classified by using the "connected to a cluster" rule.

   (c) IEFCT = -1: The class has either been successfully split or it cannot be broken up. In both cases, the user is informed.

   A class cannot be broken up if there has been no effect (IEFCT = ∅) and LVL = LMAX, so that the threshold cannot be raised. A class is successfully split if at least two new classes are found and all the vectors are reclassified.

8. The unclassified vectors are sent through the classifier a second time. A new LMAX is calculated for these vectors and some new threshold is chosen as a function of this value. We note that the new LMAX must be lower than the current threshold LVL.

9. This subroutine checks for overlap between the classes found at the current iteration. If there is no overlap then the program continues. If there is overlap then there are two possibilities. If the current LVL is less than the current LMAX, the threshold is raised and another attempt is made. If LVL equals LMAX, then no attempt is made to find new clusters.

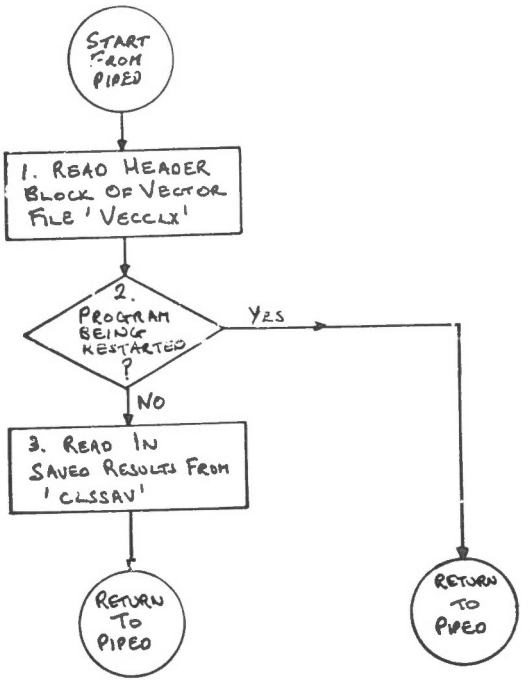
10. The results in the form of class statistics are shown on the user's terminal. The user can then either display the results, break a class, combine two or more classes into one, or re-assign the vectors of any of the classes on an individual basis, to the closest class using the Euclidean distance to the class mean vector.

IN: The function of this subroutine is to determine whether the user is starting the program for the first time. This information is stored in the header block of the vector file stored in the file 'VECCLX'. If the program is being started for the first time, IRESTA is set to zero and control returns to PIPED. Otherwise, the results from the previous classification are read in from the file 'CLSSAV'.

1. The header block indicates whether the program is being started for the first time.

2. The answer is returned in IRESTA.

3. 'CLSSAV' is simply a file containing the results from a previous classification.



IN

LEVEL: This subroutine, called by the main program determines the threshold (LVL) for the chosen class, MXCLAS. The maximum frequency, LMAX, is calculated. By assuming that LVL $\leq$ LMAX, there will always be at least one vector with the threshold value. The flag IEFCT indicates to LEVEL from which point in the main program it is being called.

1. LMAX is the maximum frequency of MXCLAS. If the program is recycling, then LMAX is the maximum frequency of the residue of MXCLAS.

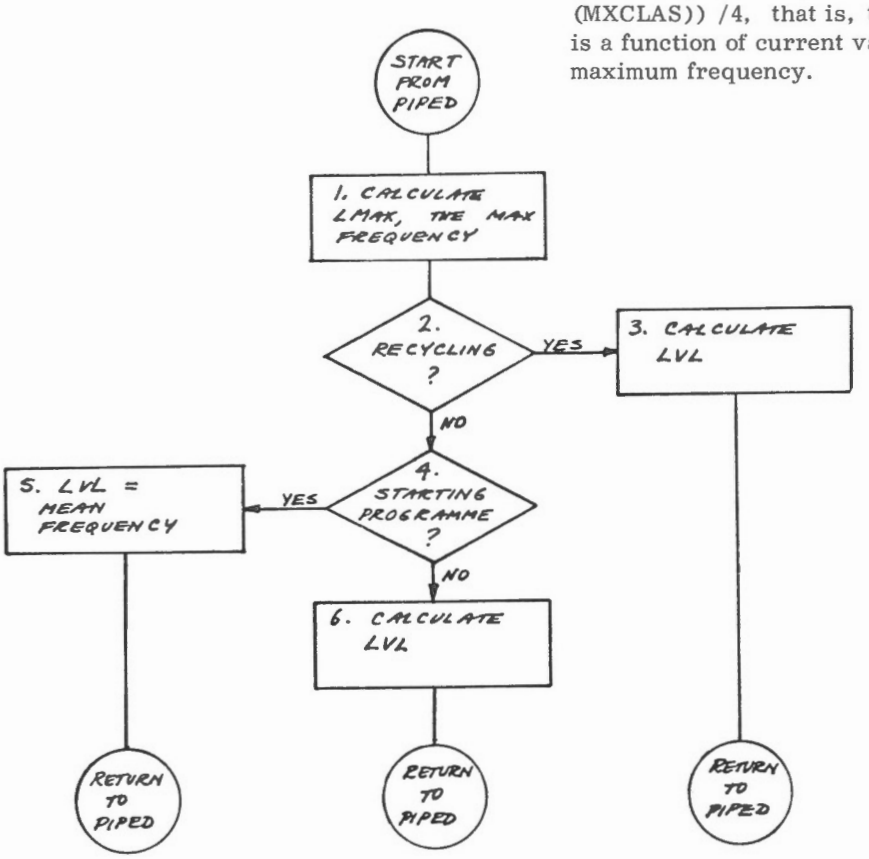
2. If the program is recycling, then IEFCT = 1.

3. LVL = MIN(LEVEL (MXCLAS), 3/4 IMAX), where Level (MXCLAS) is the value of LVL at which MXCLAS was generated.

4. This condition is indicated by ISW2 = Ø.

5. The initial threshold level is the mean frequency of the vectors.

6. User has chosen to break up MXCLAS. The threshold level is: LVL = LEVEL (MXCLAS) + 2 + (LMAX-LEVEL (MXCLAS)) /4, that is, the threshold is a function of current value and the maximum frequency.



LEVEL

TREE: Subroutine TREE generates the parallelepipeds which define the classes.

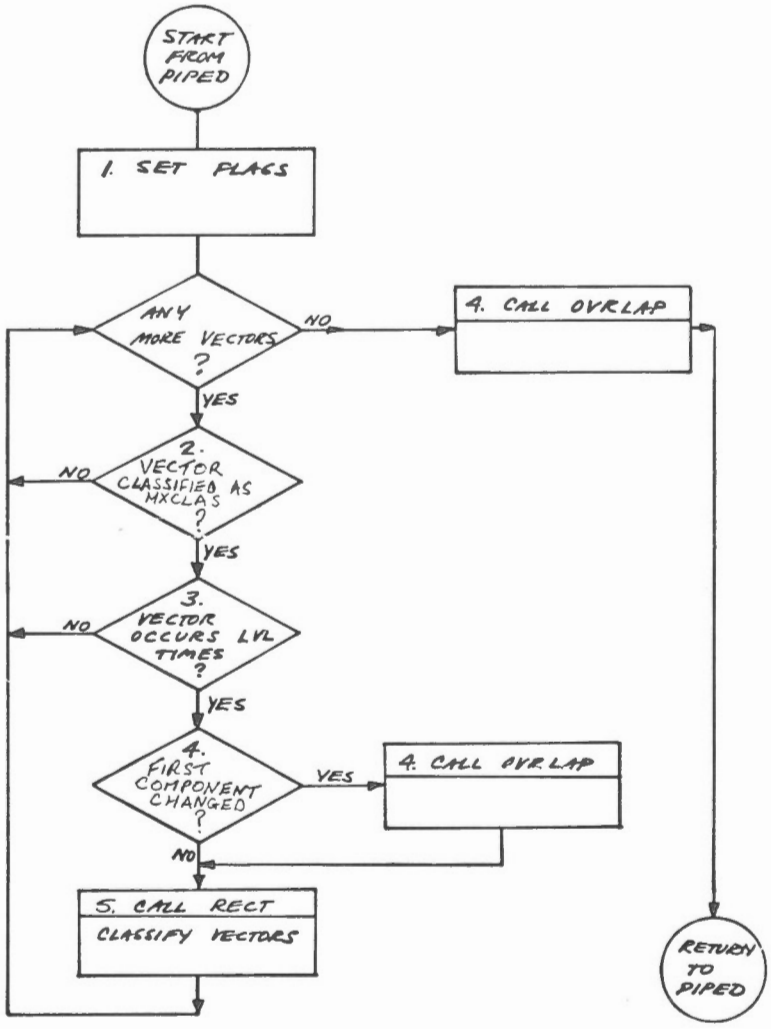
1. The flags ISW1 and I1CUR are set to zero. ISW1 indicates to subroutine RECT to expand existing classes and generate new classes. I1CUR is used to indicate when the value in the first component of the vector has changed.

2. MXCLAS is the class chosen by the user to be broken up; initially it is set to zero so that all vectors are inspected. This loop picks up the vectors currently classified as MXCLAS.

3. LVL is the threshold value calculated in subroutine LEVEL.

4. Every time the value in the first component changes, subroutine OVRLAP is called and overlapping classes are merged. This step reduces the intermediate number of classes, as classes may grow into each other.

5. This subroutine is called to classify the vectors and to set up the parallelepiped bounds.



TREE

KLASS: This subroutine, called from the main program, classifies the vectors belonging to class MXCLAS, by using the connectedness rule.

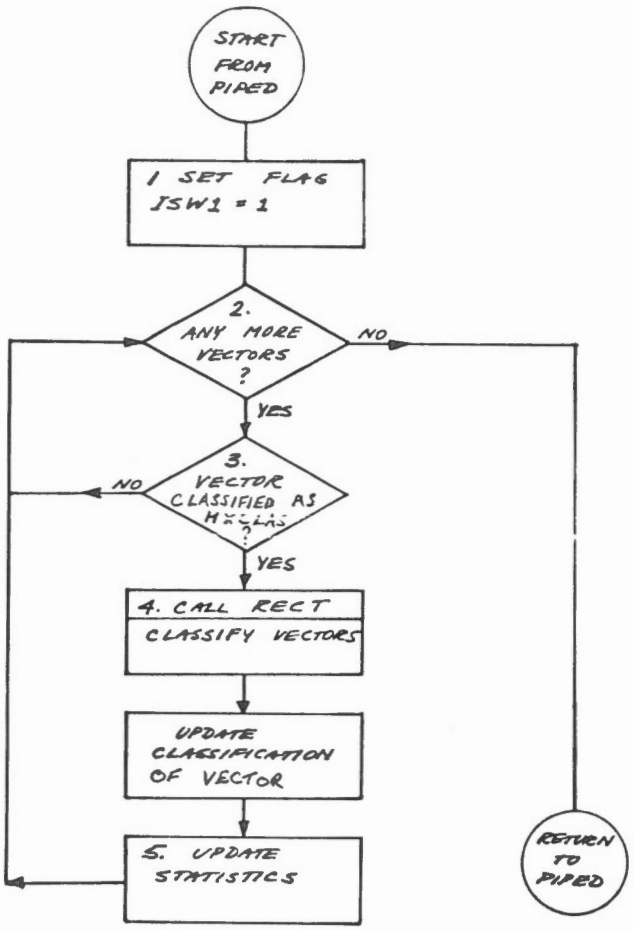
1. The flag ISW1 is set to 1, to indicate to subroutine RECT that the vectors are only to be classified, and not used to create or expand classes.

2. KLASS checks each vector in the histogram list.

3. If the vector is not in the chosen class MXCLAS, then it is ignored. Initially, MXCLAS = ∅, and all the vectors are checked.

4. RECT determines if the vector belongs to an existing class using the connectedness rule. This information is returned in the common variable KLASS.

5. The arrays updated are NOHIST, NOVECT, and RMEAN.



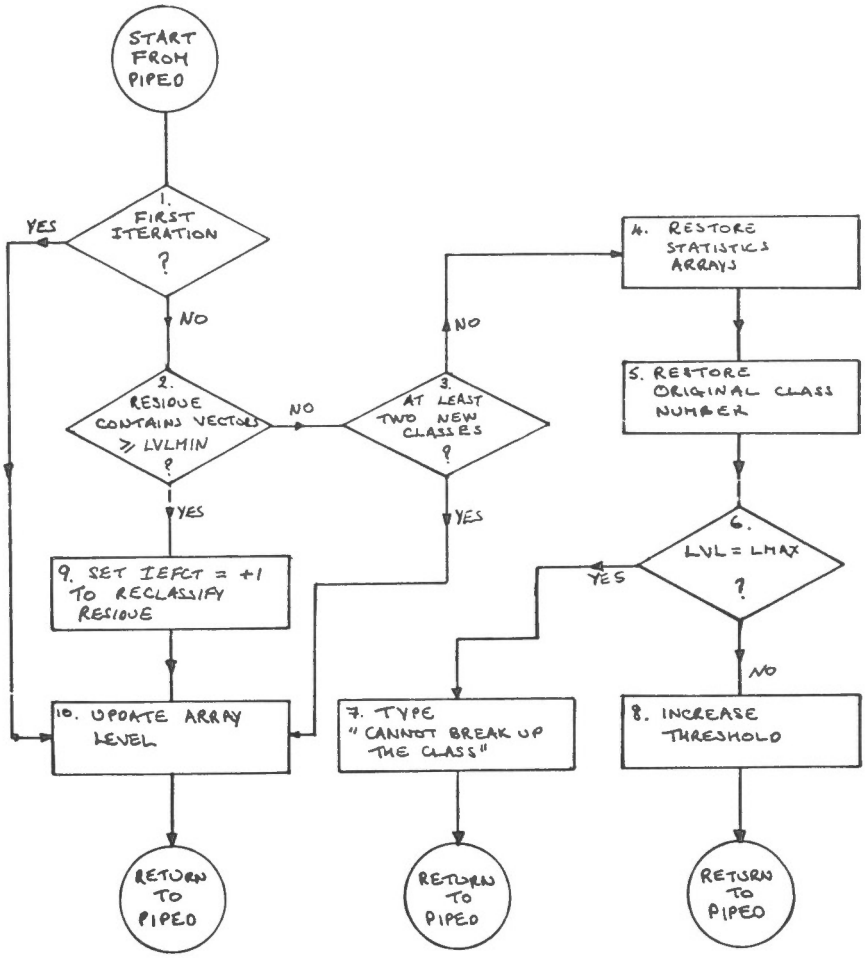
KLASS

EFFECT: Subroutine EFFECT monitors the results of the classification. The subsequent action to be taken by PIPED is returned in the flag IEFCT. There are only three possible returns:

(i) IEFCT = 0: Using the current threshold value, the class MXCLAS cannot be broken up. The threshold value must be raised and a new attempt made at breaking up the class. This condition is detected by comparing the number of classes presently generated with the number found after the previous iteration. If there is only one new class, then a check must be made of the vectors still unclassified. If there are no vectors still unclassified or if all the unclassified vectors occur with frequency less than a preset value LVLMIN, then the threshold value LVL must be raised.

(ii) IEFCT = +1: Some of the vectors of MXCLAS have not been assigned to any of the new clusters. These vectors, the residue, are recycled through the clustering algorithm and new clusters may be identified.

(iii) IEFCT = -1: The present iteration of the algorithm must terminate and the results are displayed to the user. There are two possibili-



EFFECT

ties, either class MXCLAS has been successfully split into two or more classes or MXCLAS cannot be broken up. EFFECT concludes that the class cannot be broken up if IEFCT = ∅, and the current threshold value is already at the maximum value, LMAX.

1. For the first iteration no checks are made.

2. The residue or unclassified vectors of MXCLAS are checked for the presence of at least one significant vector; that is, a vector occurring at least LVLMIN times. LVLMIN is the smallest value that the threshold LVL may assume.

3. If there are at least two new classes, IEFCT is set to -1 and the results will be displayed to the user.

4. The arrays LEVEL, NOHIST, NOVECT, must be reset to their values of the previous iteration.

5. The class numbers of the vectors must be changed back to MXCLAS.

6. If LVL equals LMAX, then the threshold cannot be increased.

7. IEFCT is set to -1, and control will pass to the user.

8. The threshold is increased in a roundabout manner. LEVEL(MXCLAS) is increased to the current threshold, so that on the subsequent pass through subroutine LEVEL, a higher threshold will be set.

9. The unclassified vector or residue must be reclassified. IEFCT is set to + 1.

10. The array LEVEL stores the threshold value at which each class is generated, and must be updated each time a new class is created.

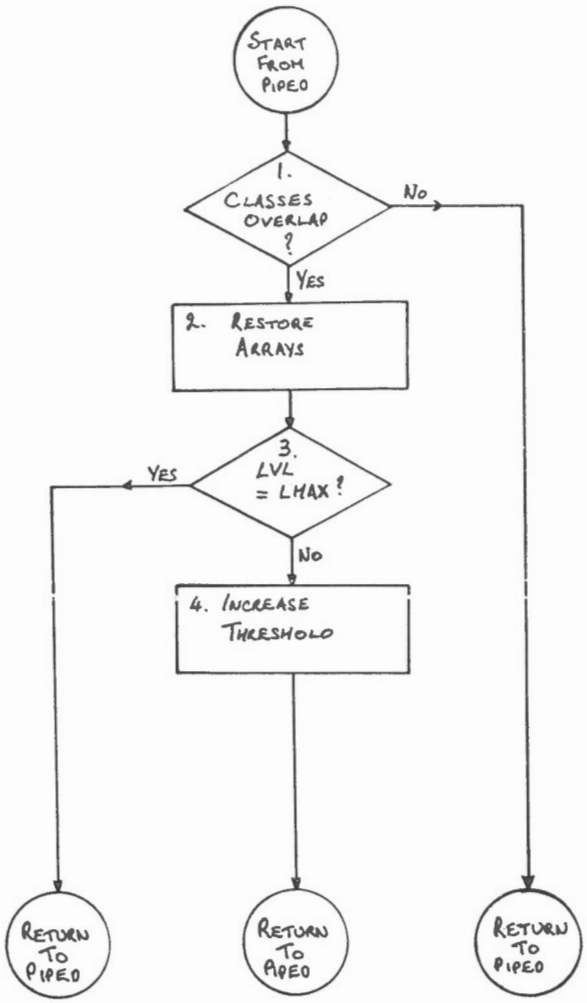
CHECK: This subroutine is called by PIPED in the recycle portion of the program. This subroutine checks if the classes found among the residue of MXCLAS overlap the classes previously generated. There are three possibilities and these are returned to PIPED by the flag ICHECK.

1. Two classes overlap if the parallelepiped corresponding to the classes intersect. Set ICHECK = ∅.

2. Array BOUND contains the boundaries of the classes; therefore, the positions corresponding to the newly found classes which overlap, must be reset to zero.

3. If the current value of the threshold is already at its maximum value, then ICHECK is set to 2, and control returns to PIPED.

4. The threshold is set to (LVL ± LMAX ± 1) /2, where LVL is the current threshold. Set ICHECK = 1.



CHECK

RECT: Subroutine RECT is called by TREE and by KLASS. When called by TREE (ISW1 = ∅), the vector is classified according to connectedness and the class boundaries are expanded if necessary. If called by KLASS (ISW1 = 1), the vectors are only classified and the class boundaries are not expanded.

1. Check if NCLASS = NOLD, where NCLASS is the current number of classes, and NOLD is the previous number of classes.

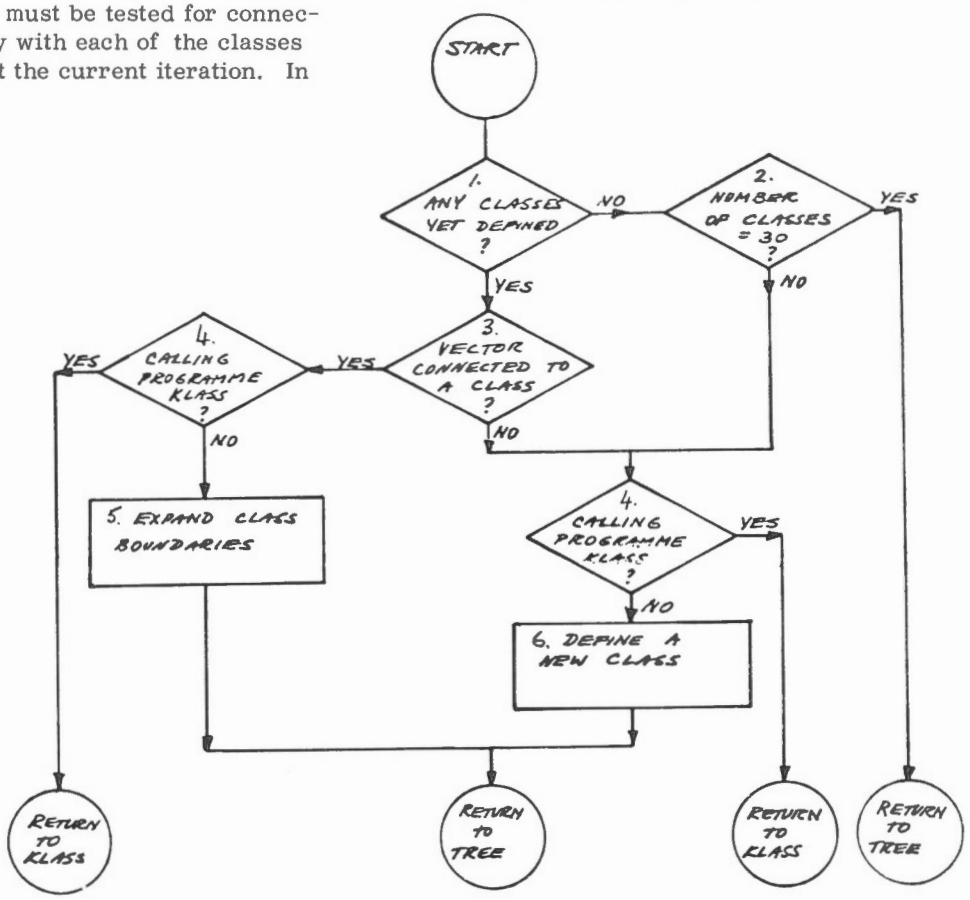
2. The maximum number of classes is 30.

3. Each vector must be tested for connectedness only with each of the classes generated at the current iteration. In order to be connected to a class it must be within the parallelepiped resulting when the class bounds are extended simultaneously by one in each direction.

4. If ISW1 = 1, the calling program is KLASS and no new classes are generated.

5. The bounds are expanded if necessary by values of plus or minus one.

6. If the vector is (i_1, i_2, i_3, i_4), then the new class is defined as: $(i_1-i_1, i_2-i_2, i_3-i_3, i_4-i_4)$; a parallelepiped consisting of one point.



RECT

OVRLAP: Subroutine OVRLAP is called only by TREE and checks whether the parallelepipeds defining the classes overlap. If there is an overlap, the two classes are merged.

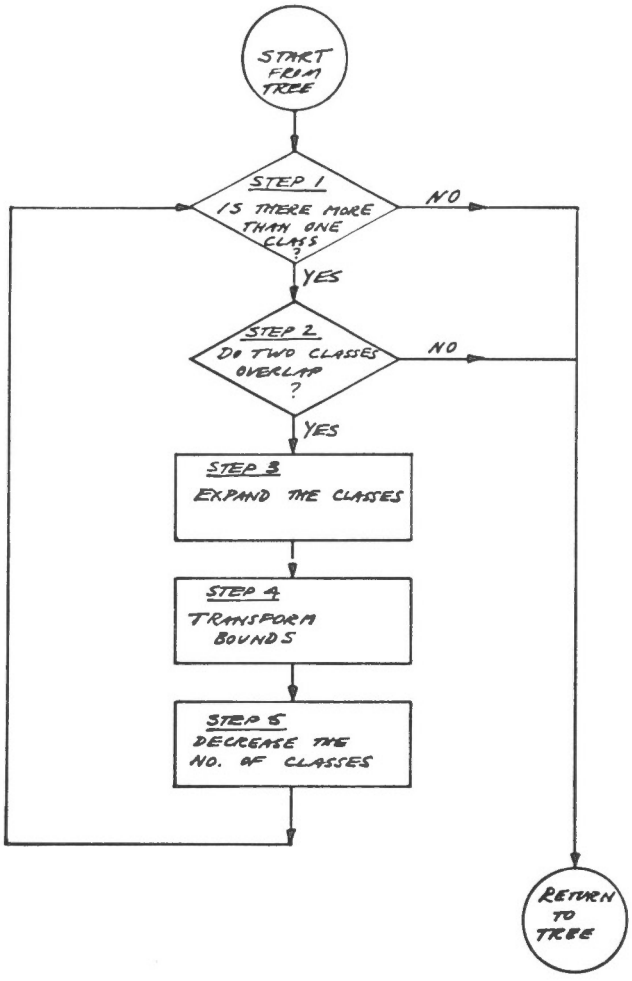
1. Verify that there are at least two classes before checking for overlap.

2. Two classes are said to overlap, if the parallelepipeds defining the two classes intersect, when the boundaries of both classes are increased by one in all 8 directions.

3. Find the smallest parallelepiped containing both classes.

4. Replace both classes by the class corresponding to the parallelepiped found.

5. The number of classes, NCLASS, is decreased by one.



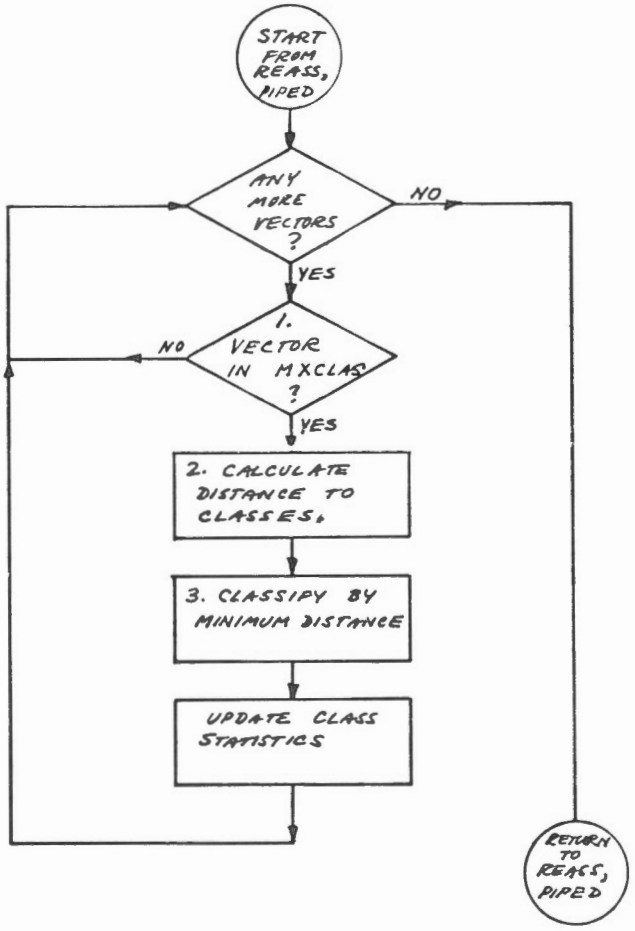
OVRLAP

UNCLA: This subroutine is called by subroutine PIPED and REASS. Its function is to reclassify vectors on the basis of the minimum Euclidean distance to the class means. When called by PIPED, these are the vectors which are still unclassified after recycling. When called by REASS, the vectors belong to the classes chosen by the user to be reassigned.

1. When called by PIPED, this corresponds to the remaining unclassified vectors after recycling.

2. The Euclidean distance function is defined as the square root of the sum of the squares of the differences between the individual components of the vector and the class mean vector.

3. The vector is reclassified into the closest class.

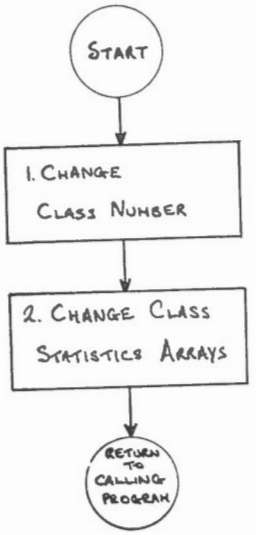


UNCLA

MOVE: This subroutine is called to squeeze out an empty class, which has resulted because all the vectors of the class have been reclassified.

1. Class MXCLAS is the empty class. The vectors of classes 1 to (MXCLAS-1) are left untouched. The other class numbers are decreased by one.

2. The positions in the arrays, NOHIST, NOVECT, LEVEL and BOUNDS corresponding to classes greater than MXCLAS are shifted down by one. The other positions are left untouched.



MOVE

OUT: After each iteration, the results are typed out on the user's terminal. The user can now either display the results, break up some class, combine classes, or reassign the vectors of a class, on an individual basis, to the closest class. In order to display, the entire common area is saved on a disc file called 'CLSSAV', so that re-entry to the program is possible. The vectors must then be reformatted and stored in a disc file called 'VECCLU', which is then read by the display program.

1. The mean vector and covariance matrix of each class is calculated and stored in a disc file called CLUST. ST1.

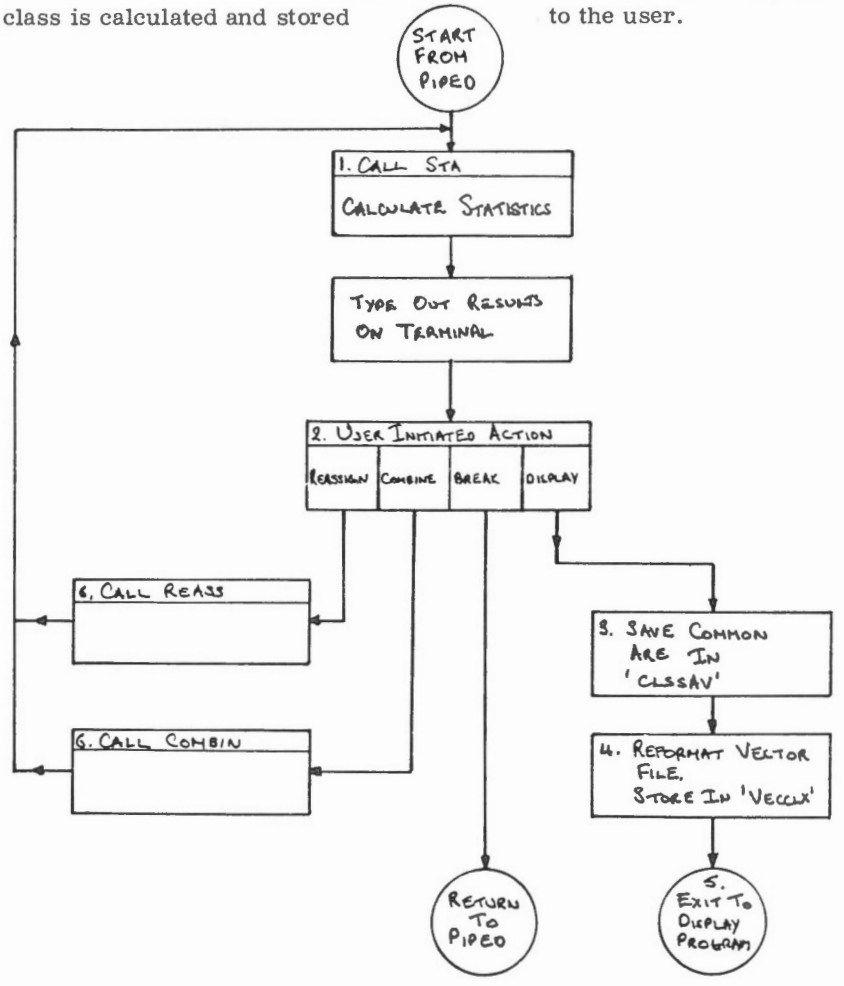
2. The user must decide what the next action is to be.

3. The common area is stored in disc file 'CLSSAV'.

4. The vectors are reformatted and stored in disc file 'VECCLU'.

5. Control is passed to the display programs.

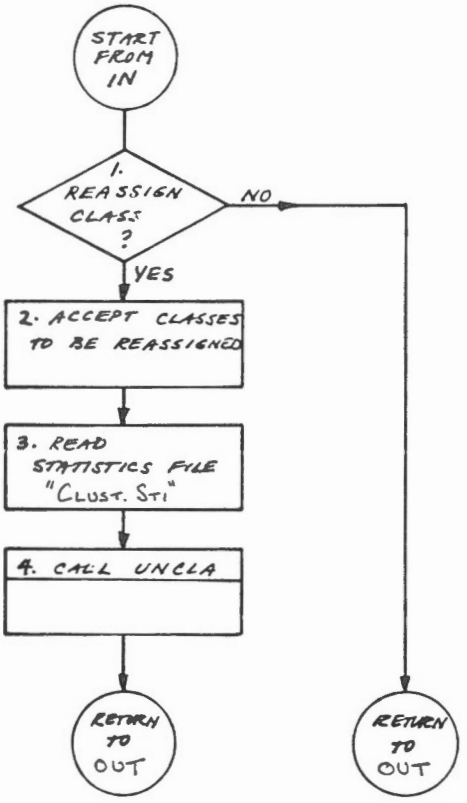
6. After the requested classes are combined or re-assigned, the statistics are recalculated, the new results are typed out on the terminal, and control returns to the user.



OUT

REASS: This subroutine, called by subroutine OUT, permits the user to choose up to 15 classes, the vectors of which are then reassigned on an individual basis amongst the remaining classes according to the minimum distance to the class means.

1. The user decides whether to reassign.
2. Up to 15 classes can be reassigned.
3. The CLUST.STI disc file is created in subroutine OUT.
4. Subroutine UNCLA reclassifies the vectors of a class using minimum distance. UNCLA is called once for each class to be reassigned.



REASS

COMBIN: This subroutine, which is called by OUT, merges up to 15 classes specified by the user. The new class is now called class 1, and the numbering of the remaining classes is rearranged.

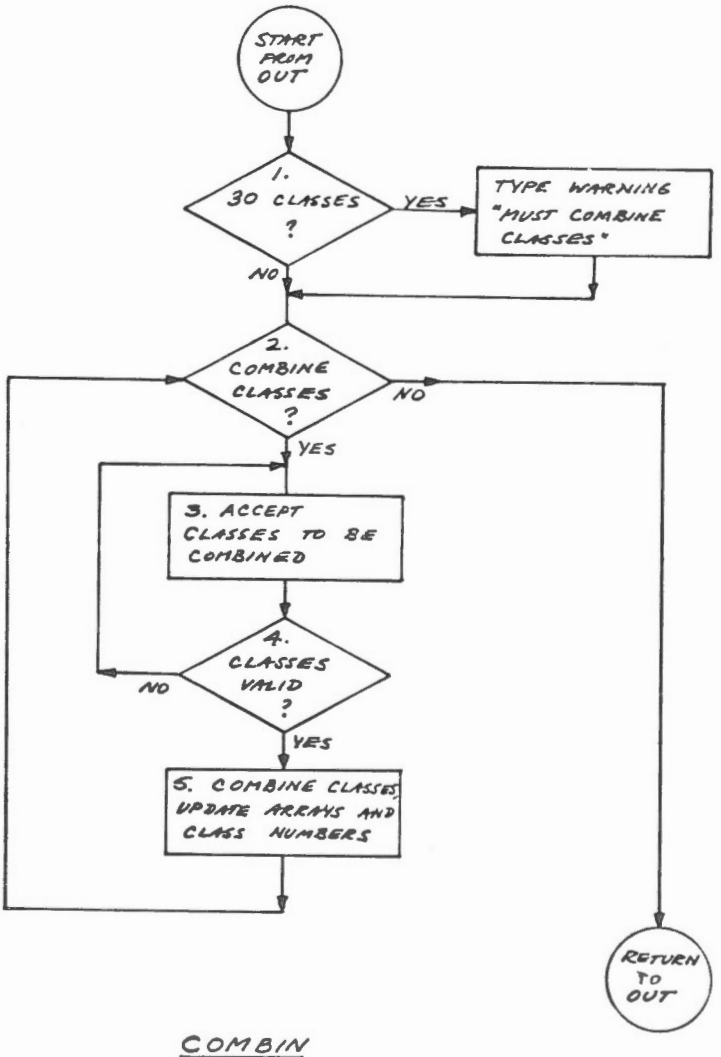
1. When the number of class is 30, the maximum, the user is warned.

2. User chooses whether to combine classes.

3. The user enters up to 15 classes to be merged.

4. A check is made that these numbers are valid.

5. A transformation array containing the new class numbers is calculated and the class numbers are changed. The various class statistics are also updated.



COMBIN