

## **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.

# 2. AgRISTARS

E83-10015

7. SR-T2-04371  
5. NAS-9-14689

NASA-CR-167701

A Joint Program for  
Agriculture and  
Resources Inventory  
Surveys Through  
Aerospace  
Remote Sensing

"Made available under NASA sponsorship  
in the interest of early and wide dis-  
semination of Earth Resources Survey  
Program information and without liability  
for any use made thereof."

## Supporting Research

6. June, 1982

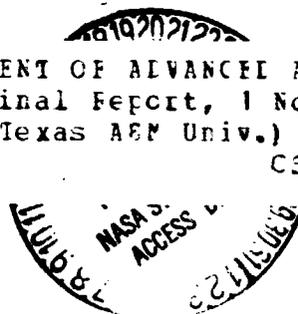
FINAL REPORT

DEVELOPMENT OF ADVANCED ACREAGE ESTIMATION METHODS

3. L. F. Guseman, Jr.

(E83-10015) DEVELOPMENT OF ADVANCED ACREAGE  
ESTIMATION METHODS Final Report, 1 Nov.  
1980 - 30 Jun. 1982 (Texas A&M Univ.) 275 p  
EC A12/MF A01 CSCI 05E

N83-12494  
In8U  
N83-12498  
Unclass  
G3/43 00015



NASA



4. DEPARTMENT OF MATHEMATICS  
TEXAS A&M UNIVERSITY  
COLLEGE STATION, TEXAS 77843

FINAL REPORT  
DEVELOPMENT OF ADVANCED  
ACREAGE ESTIMATION METHODS  
Contract NAS-9-14689  
November 1, 1980 - June 30, 1982

Prepared for:

Earth Observations Division  
NASA/Johnson Space Center  
Houston, Texas 77058

by

L. F. Guseman, Jr.  
Principal Investigator  
Department of Mathematics  
Texas A&M University  
College Station, Texas 77843

## ACKNOWLEDGMENTS

The work reported herein was carried out for the Earth Observations Division, NASA/Johnson Space Center, Houston, Texas, under Contract NAS-9-14689 to the Texas A&M Research Foundation, College Station, Texas, 77843, during the period November 1, 1980 to June 30, 1982. The investigations were carried out by personnel at Texas A&M University, University of Houston, and University of Tulsa.

L. F. Guseman, Jr.  
Principal Investigator

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.	
4. Title and Subtitle FINAL REPORT: ✓ Development of Advanced Acreage Estimation Methods		5. Report Date June, 1982	
		6. Performing Organization Code	
7. Author(s) L. F. Guseman, Jr. ✓		8. Performing Organization Report No.	
		10. Work Unit No.	
9. Performing Organization Name and Address Department of Mathematics Texas A&M University College Station, Texas 77843		11. ✓ Contract or Grant No. NAS-9-14689	
		13. Type of Report and Period Covered FINAL (11/1/80-6/30/82)	
12. Sponsoring Agency Name and Address Earth Observations NASA/Johnson Space Center Houston, Texas 77858		14. Sponsoring Agency Code	
		15. Supplementary Notes	
10. Abstract  Work carried out under this contract was concerned with:  Refinements and Documentation of the AMOEBA Clustering Algorithm Rice Scene Radiation Characterization Applied Research Spectral-Spatial Classification Algorithm Development Use of the Akaike Information Criterion			
17. Key Words (Suggested by Author(s))		18. Distribution Statement	
19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*

## DEVELOPMENT OF ADVANCED ACREAGE ESTIMATION METHODS

### INTRODUCTION

A practical application of remote sensing which is of considerable interest is the use of satellite-acquired (LANDSAT) multispectral scanner (MSS) data to conduct an inventory of some crop of economic interest such as wheat over a large geographical area. Any such inventory requires the development of accurate and efficient algorithms for analyzing the structure of the data. The use of multi-images (several registered passes over the same area during the growing season) increases the dimension of the measurement space. As a result, characterization of the data structure is a formidable task for an unaided analyst.

Cluster analysis has been used extensively as a scientific tool to generate hypotheses about structure of data sets. Sometimes one can reduce a large data set to a relatively small data set by the appropriate grouping of elements using cluster analysis. In some cases, the algorithm which effects the grouping becomes the basis for actual classification. In other cases, the cluster analysis produces groupings of the data which in turn serve as a starting point for other algorithms which produce acreage estimates. Additional uses of cluster analysis arise in conjunction with dimensionality reduction techniques which are used to generate displays for purposes of further interactive analysis of the data structure.

Work carried out under this contract dealt with algorithm development, theoretical investigations, and empirical studies. The algorithm development tasks centered around the refinement of the AMOEBA clustering/classification algorithm, and its subsequent use as a starting point for HISSE, a maximum likelihood proportion estimation procedure. Theoretical results were obtained

which form a basis for the maximum likelihood estimation procedures. In addition, some investigations were made into the use of the Akaike information criterion (AIC) when applied to mixture models. Additional work was concerned with the development of a preliminary research plan which delineates some of the technical issues and associated tasks in the area of rice scene radiation characterization.

Specifically, investigations were carried out in the following areas:

Refinements and Documentation of the AMOEBA Clustering Algorithm

Rice Scene Radiation Characterization Applied Research

Spectral-Spatial Classification Algorithm Development

Use of the Akaike Information Criterion

Each of these investigations is discussed in turn in the sequel.

## 1. REFINEMENTS AND DOCUMENTATION OF THE AMOEBA CLUSTERING ALGORITHM

Detailed documentation of the AMOEBA clustering/classification algorithm for the version implemented on the HP-3000 System at EROS Data Center appears in an attached report entitled:

Jack Bryant, System support documentation--IDIMS FUNCTION--AMOEBA,  
Department of Mathematics, Texas A&M University, March, 1982.

Included throughout the documentation are comments which indicate where code changes could or should be made to transport the program to another system.

## 2. RICE SCENE RADIATION CHARACTERIZATION APPLIED RESEARCH

Work for this task was performed by Dr. James Heilman, Remote Sensing Center, Texas A&M University. The results of his investigations are presented in the attached report entitled:

James Heilman, Rice Scene Radiation Research Plan, Remote Sensing Center, Texas A&M University, December, 1981.

### 3. SPECTRAL-SPATIAL CLASSIFICATION ALGORITHM DEVELOPMENT

The objective of this study was to formulate and test algorithms based on a likelihood function which respected the integrity of some predetermined structure in the data.

For purposes of these investigations, the "pure field data" (patches) determined by the AMOEBA algorithm were used as the predetermined structure. A maximum likelihood parameter estimation procedure (HISSE) was designed to respect (take into account) field integrity.

A mathematical description and implementation of the procedure, along with results from preliminary tests, appear in the report:

Charles Peters and Frank Kampe, Numerical trials of HISSE, Contract NAS-9-14689, SR-HO-00477, Department of Mathematics, University of Houston, August, 1980.

Theoretical results underlying the approach used in the HISSE algorithm are discussed in the report:

Charles Peters, On the existence, uniqueness, and asymptotic normality of a consistent solution of the likelihood equations for nonidentically distributed observations--applications to missing data problems.

Contract NAS-9-14689, SR-HO-00492, Department of Mathematics, University of Houston, September, 1980.

Additional theoretical results were obtained which address the convergence of a particular iterative form of the likelihood equations in the case of a mixture of densities from (possibly distinct) exponential families. These results appear in the report:

Richard A. Redner, An iterative procedure for obtaining maximum likelihood estimates in a mixture model, Contract NAS-9-14689, SR-T1-0481, Division of Mathematical Sciences, University of Tulsa, September, 1980.

Use of a modification of the HISSE model for the case of pure LANDSAT agricultural data sets are discussed in the attached report:

Charles Peters, On possible modifications of the HISSE model for pure agricultural data, Contract NAS-9-14689, SR-H1-04037, Department of Mathematics, University of Houston, February, 1981.

#### 4. USE OF THE AKAIKE INFORMATION CRITERION

The objective of this study was to investigate the application of the Akaike Information Criterion (AIC) to a mixture model. In particular, investigations were carried out concerning the use of the AIC in selecting the number of components of a mixture model. The results of these investigations are discussed in the attached report:

Richard A. Redner, The Akaike information criterion and its application to mixture proportion estimation, Contract NAS-9-14689, SR-T1-04207, Division of Mathematical Sciences, University of Tulsa, November, 1981.

ATTACHED REPORTS

D<sub>1</sub> Jack Bryant, System support documentation--IDIMS FUNCTION--AMOEB, Department of Mathematics, Texas A&M University, March, 1982.

D<sub>2</sub> James Heilman, Rice scene radiation characterization applied research, Remote Sensing Center, Texas A&M University, December, 1981.

D<sub>3</sub> Charles Peters, On possible modifications of the HISSE model for pure agricultural data, Contract NAS-9-14689, SR-H1-04037, Department of Mathematics, University of Houston, February, 1981.

D<sub>4</sub> Richard A. Redner, Genshiro Kitagawa, and William A. Coberly, The Akaike information criterion and its application to mixture proportion estimation, Contract NAS-9-14689, SR-T1-04207, Division of Mathematical Science, University of Tulsa, November, 1981.

N83 12495

D/

SYSTEM SUPPORT DOCUMENTATION

IDIMS FUNCTION

AMOEBA

Jack Bryant  
Texas A&M University  
College Station, Texas

March 1982

## COMPUTER PROGRAM ABSTRACT

Sites At Which Developed: Texas A&M University and EROS Data Center

Symbolic Name: AMOEBA

Parent System: IDIMS

Language: FORTRAN 100%

Key Words: Clustering, Boundary detection, Classification, Spatial model, Pair probability of misclassification

Contact: Jack Bryant, Department of Mathematics, Texas A&M University,  
College Station, TX, 77843, 713-845-3169  
Susan K. Jenson, Applications Branch, EROS Data Center,  
Sioux Falls, South Dakota

Status: Completed

## ABSTRACT

AMOEBA is a clustering program based on a spatial-spectral model for image data. It is fast and automatic (in the sense that no parameters are required), and classifies each picture element into classes which are determined internally. As an IDIMS function, no limit on the size of the image is imposed.

## TABLE OF CONTENTS

1.	INTRODUCTION . . . . .	1
	Bugs? . . . . .	1
	The Program . . . . .	1
	Wide Image Logic . . . . .	2
	The Mask . . . . .	3
	Four Neighbors . . . . .	3
	Circular Buffers . . . . .	4
	Tricking FORTRAN . . . . .	4
	Rejection Thresholds . . . . .	4
	Memory Management and Subroutine Linkage . . . . .	5
	Organization of Detailed Documentation . . . . .	8
	Acknowledgment . . . . .	9
2.	MAIN . . . . .	11
3.	DETAILED DOCUMENTATION OF SUBROUTINES . . . . .	21
	AMSTATS . . . . .	23
	ASELECT . . . . .	27
	CLASSIFY . . . . .	35
	CLOSEC . . . . .	43
	COLAPS . . . . .	47
	CONNCT . . . . .	51
	DIAMTR . . . . .	55
	FILLLR . . . . .	59
	FIXUP . . . . .	63
	GETN25 . . . . .	67
	IIIFN . . . . .	71
	MAPP . . . . .	75
	MARKLR . . . . .	79
	MARKUP . . . . .	83
	MARKUPDN . . . . .	87

PRECEDING PAGE BLANK NOT FILMED

MOREQUES . . . . .	91
MRKIVL . . . . .	99
MSORT. . . . .	99
NUMCLU . . . . .	103
PERPIXEL . . . . .	115
REJECTH. . . . .	119
SETSYM . . . . .	123
SHELL. . . . .	127
SORT . . . . .	131
START. . . . .	135
THINTSTM . . . . .	143
THRFDND . . . . .	153
UNCLE. . . . .	163
APPENDIX A. The Theoretical Foundation of AMOEBA. . . . .	167
APPENDIX B. Summary of System Subroutines Used. . . . .	173
APPENDIX C. IDIMS User Documentation. . . . .	175
APPENDIX D. Sample Session Using AMOEBA . . . . .	181
APPENDIX E. Sample Batch Job. . . . .	185

## 1. INTRODUCTION

AMOEBA is a clustering program designed during the Large Area Crop Inventory Experiment in 1977-78. The original idea\* was developed in an agricultural setting (large fields, few real classes). It was a nice surprise to discover that the program solves other problems. It has an uncanny ability to discover structure in image data, at least when the structure exists. Because of the nature of the method, it operates efficiently on small (16 bit) computers lacking floating point hardware. In some sense, the smaller the computer the better it works.

Bugs? Before going on, we report a problem experienced at the EROS Data Center (EDC), a U.S. Geological Survey installation at Sioux Falls, South Dakota. Unquestionably correct FORTRAN source code produces nonsensical results. The bug is easily demonstrated, and may be related to the file management system of IDIMS. It is not encountered unless huge images with many bands are being processed. We do not know whether it is FORTRAN, IDIMS, the operating system, or local hardware. We do know it is not a problem in the source code. The source code is believed to be without error. Scores of hours with the Hewlett-Packard IDIMS-DEBUG utility only prove the program is lost. We welcome suggestions of any kind whatever which may indicate what is wrong. Fortunately, the bug shows up as a simple failure with meaningless cluster centers, or a bounds violation where, according to the source, none is possible. That is, it seems unlikely that the bug causes real damage since the user will be informed of garbage answers. A cynical systems programmer could suggest that a disappointed user use ISOCLS instead. A diligent one would find the bug, whatever it is. We are neither. We are only exhausted, and wish you luck if you look for it.

The Program. The idea underlying the program is easy to state. A full description is given in Appendix A. Here we sketch the idea. Our goal is to sort the pixels of an image into classes that will show an

---

\*Jack Bryant, On the clustering of multidimensional pictorial data, Pattern Recognition 11, pp. 115-125, 1979.

analyst the structure of the image. Suppose one has two partitions of a set into a family of disjoint subsets. A measure of the distance between the partitions is the probability that points are clustered alike in the first and differently in the second, plus the probability they are classified alike in the second and differently in the first. Using a boundary finding algorithm, we can extract samples from the data we believe are alike. These will reside in spatially connected patches in the complement of the boundary. By ordering samples on some one-dimensional attribute, we are able to find some we believe are likely to be in different real classes. The samples, called test pixels, come in test sets of five each, and are used to evaluate our clustering. Test set means form starting cluster centers.

The number of clusterings of an image is astronomical. Rather than evaluate all clusterings, AMOEBA successively eliminates cluster centers which are involved in nearest neighbor assignments that split pairs from the same test set, or gather pairs from widely separated test pixels. Clusters are never combined, chained, or split. They are merely eliminated, starting with the set of test set means, and ending with a set of clusters between the user-supplied maximum and minimum numbers.

There are a number of general features about the program which have nothing to do with the clustering and classification method, but which do make the program harder to understand (harder, that is, than the big system original version in which data was assumed to be all in memory at once). Before we start detailed documentation, we comment on some of the trickier details which apply to more than one component. Nomenclature for the following:

- COUNT -- a counter of the number of elements in each class.
- ND -- the dimensionality or number of bands.
- REJECT -- a vector of thresholds used to check classification based on a spatial mixture model.

Wide Image Logic. There is no limit to the size of input image which can be processed (other than disk storage). Yet there is a severe limitation on data storage: subroutines START and CLASSIFY each require three lines of data storage and three lines of labels. About 20,000 words

are available, so the maximum width is about  $20,000/((ND+3)*3)$ . If ND is 4, for example, fewer than 1000 samples per line can be processed. Therefore, the program segments the image into strips of width NC, with the actual width NZ being passed to various subroutines.

The Mask. In IDIMS, imagery is organized in rectangular arrays; however, the image itself is often not rectangular. The value 0 is usually stored in each band or channel of the "Mask". In AMOEBA, a logical flag MASK (optional parameter, default .TRUE.) is used to tell the program whether a value 0 in channel 1 is to be used as a mask. If set, no processing is wasted on these pixels. They are labelled with the label 99 and counted in COUNT(100). If some are found, their count is printed at the conclusion of the program.

Four Neighbors. Each pixel inside an image with rectangular organization has exactly four nearest neighbors. There are concepts for which more than the four neighbors can be considered. Discrete connectedness, however, is not one of them. AMOEBA uses discrete connectedness to form patches in the complement of the boundary. More precisely, a path is a sequence  $p_1, \dots, p_n$  of pixels such that  $p_i$  is a neighbor of  $p_{i+1}$  for  $i = 1, \dots, n-1$ . A set of pixels is said to be connected if each pair of points in the set is contained in a path lying entirely in the set. For example, a singleton (a set containing only one pixel) is a connected set, as is the entire image. In this discrete setting, the concept is simple, but has considerable power. Let A be an arbitrary set of pixels. For each a in A, the set of all elements of A which can be joined to a by a path within A is a connected subset of A, and is, in fact, the largest (maximal) connected subset of A containing a. Maximal connected subsets of a set are called components of the set. The patches of AMOEBA are the components of the complement of the boundary. For this to work, only the four nearest neighbors can be considered when deciding what a path is.

In the classification step, again only four neighbors are considered. Here, however, we are really making a concession to the relatively low resolution of Landsat MSS data, to poor registration of multi-temporal imagery, and to computer-time spent in classification. It would

certainly be possible to consider 8 neighbors; this has, in fact, been done in areas dominated by agricultural activity, but for general usage with Landsat resolution four neighbors are enough.

Circular Buffers. The boundary-finding and classification sections of the program require not only a pixel but the neighborhood of the pixel. With only the four nearest neighbors to consider, three lines suffice. Rather than move the data or maps around, we simply switch pointers, rolling old labels or maps out to disk and new data in. The logic is simple, and is programmed as follows: Initially, pointers I1, I2, and I3 are 1, 2, and 3. I1 is the eldest, I2 the current time to be processed, and I3 the newest. After a line has been processed, the line pointed to by I1 is stashed, data is read into the data slot pointed to by I1, and the data buffer is "rotated" by saving I1, setting I1 = I2, I2 = I3, and then I3 = the old I1.

Tricking FORTRAN. Short FORTRAN integers are 16 bits long; in the HP-3000, this is the word size. Other than 16 bit processing consumes time all out of proportion to the benefit. An obvious case is floating point processing, but long (32 bit) integer processing is also expensive.

However, standard FORTRAN two's complement arithmetic is simply not adequate. We "trick" FORTRAN by biasing all distance calculations by -32768 (in octal, 100000, in hex 8000). That is, "zero" is actually the bit pattern 1 followed by 15 zeros, and numbers grow from there to the largest two's complement integer, 32767. The same trick is used in forming labels for patches of sure pixels. We allow for 64K labels by starting the labels at -32767 (-32768 is used to mark boundary). In several places, the bias needs to be removed, and care must be taken to insure this is properly executed.

Rejection Thresholds. Boundary pixels come in two flavors: pure and contaminated. The pure ones are rare (usually not more than 20% of the image), but are easy to model. Consider the spectral picture in which a pixel represents the sensor-average of two pure classes. This we call a pure boundary pixel. A case can be made for classification of such a pixel in the nearer of the classes of which it is a mixture. Obviously, such a pure mixture is nearer the one to which it is assigned than half the distance between. This is the basis underlying the

Rejection Thresholds. For each cluster, the Rejection Threshold (calculated in subroutine REJECTH and kept in REJECT(ND)) is half the distance between this cluster and the other cluster the greatest distance away. (Actually, the square of the distance, biased by -32768, is maintained.)

The model is strictly applicable in the absence of registration error, but the model can be extended. The result, easily obtained from Jensen's inequality, is: for registration errors, the otherwise uncontaminated distances should be no farther than  $\sqrt{2}$  times the pure model. Pixels which fail this test are not classified (or reclassified). However, test pixels are believed to be pure, particularly the third in a test set of five. Therefore, in MOREQUES, the stricter test is imposed. New classes are introduced when the center test pixel fails the strict test.

Memory Management and Subroutine Linkage. Although the HP-3000 system allows dynamic array definition, we only use this in opening files (these routines must be changed to move the program to another system anyway). Memory is managed in the main program in an interger array called WORK (which is equivalenced to a logical array LWORK). This gets memory managed but makes subroutine linkage difficult to follow. As an aid to the Systems Analyst who must maintain the program, we give the exact calling sequences as they appear in MAIN, the subroutine version, and the various values of memory management parameters.

THRFD (finds integer vector thresholds, returned in WORK(1))

CALL THRFD (NFL,WORK(MM1), WORK(MM3), UICB, IND, WORK(MM4), NR, NC,  
ND, MASK, IMGIN)

SUBROUTINE THRFD(NFL,INTTHR, SCANLINE, UICB, IND, DOUNT, NR, NC, ND,  
MASK, IMGIN)

MM1 = 1	WORK(MM1)	INTTHR(ND)
MM3 = MM1 + ND	WORK(MM3)	SCANLINE(NC,ND)
MM4 = MM3 + ND*NC	WORK(MM4)	KOUNT(ND)

START (using the thresholds, estimate the boundary and create a disk  
file of boundary labels and patch labels)

```
CALL START(WORK(MM1), ND, NR, NC, NZ, WORK(MM3), WORK(MM4), WORK(MM5),
          UICB, IND, IMGIN, IMGCLAS, LABF, MASK)
```

```
SUBROUTINE START(1NTTHR, ND, NR, NC, NZ, DATBUF, LABBUF, ISCAN, UICB,
                IND, IMGIN, IMGCLAS, LAD, MASK)
```

MM1,3 as before	WORK(MM1)	INTTHR(ND)
MM4 = MM3+NC*ND*3	WORK(MM3)	DATBUF(NC,ND,3)
MM5 = MM4+CN*3	WORK(MM4)	LABBUF(NC,3)
	WORK(MM5)	ISCAN(1)

Note: Parameters NC and NZ and their interaction are described in the documentation to START.

ASELECT (select test sets and write on temporary disk file)

```
CALL ASELECT(WORK(MM2), WORK(MM3), WORK(MM4), NL, NS, NR, NC, NZ, ND,
            NTS, WORK(MM5), WORK(MM6), FILENO, UICB, IND, IMGEN, IMGCLAS)
```

```
SUBROUTINE ASELECT(DAT, LAB, KNT, NL, NS, NR, NC, NA, ND, NTS, DATA,
                  LABEL, FILENO, UICB, INC, IMGEN, IMGCLAS)
```

MM2 = MM1+ND	WORK(MM2)	DAT(NL,ND,NS)
MM3 = MM2+NL*ND*NS	WORK(MM3)	LAB(NL)
MM4 = MM3+NL	WORK(MM4)	KNT(NL)
MM5 = MM4+NL	WORK(MM5)	DATA(NC,ND)
MM6 = MM5+NC*ND	WORK(MM6)	LABEL(NC)

Note: Parameters NL and NS are described in the documentation to ASELECT.

THINTSTM ("thin" test sets and form mean vectors)

```
CALL THINTSTM(WORK(MM2), WORK(MM3), LWORK(MM4), WORK(MM5), WORK(MM6),
            N25, N60, N288, N140, N388, N428, ND, NTS, FILENO, UICB, IND)
```

SUBROUTINE THINTSTM(MP, TSP, TTP, CLASS, COUNT, N25, N60, N288, N140,  
N388, N428, ND, NTSI, FILENO, UICB, IND)

MM2 as before	WORK(MM2)	MP(ND,N140)
MM3 = MM2+ND*N428	WORK(MM3)	TSP(ND,5,N428)
MM4 = MM3+ND*N428*5	LWORK(MM4)	TTP(ND,5,N25)
MM5 = MM4+N25*ND*5	WORK(MM5)	CLASS(N25)
MM6 = MM5+MM428	WORK(MM6)	COUNT(N140)

Note: N25, N60, N140, N428, N388, and N288 are described in the documentation to THINTSTM.

SORT (sorts test pixel sets in average odd channel order)

CALL SORT(WORK(MM3), WORK(MM4), WORK(MM5), ND, N428T5, N428)

SUBROUTINE SORT(TSPXL, DUMMY, INDEX, ND, NP, NP5)

MM3 as before	WORK(MM3)	TSPXL(ND,NP)
MM4 = MM3+ND*N428T5	WORK(MM4)	DUMMY(NP5)
MM5 = MM4+NP5	WORK(MM5)	INDEX(NP5)
N428T5 = N428*5		

NUMCLU (determines the number of clusters and their centers)

CALL NUMCLU(WORK(MM2), ND, N140, N428T5, WORK(MM3), NFCLUS, MINCLN,  
MAXCLN, WORK(MM4), WORK(MM5), WORK(MM6), WORK(MM7), WORK(MM8), WORK(MM9),  
UICB, IND, WORK(MM10))

SUBROUTINE NUMCLU(MEAN, ND, NP5, NP, TSPXL, NFCLUS, MINCLN, MAXCLN,  
CLASS, COUNT, ERROR, SAVE, DUM, CSAVE, UICB, IND, NUM)

MM2, MM3 as before	WORK(MM2)	MEAN(ND,NP5)
N140 and N428 were modified	WORK(MM3)	TSPXL(ND,NP)
MM4 = MM3+ND*N428T5	WORK(MM4)	CLASS(NP)
MM5 = MM4+N428T5	WORK(MM5)	COUNT(NP5)
MM6 = MM5+N140	WORK(MM6)	ERROR(NP5)
MM7 = MM6+N140	WORK(MM7)	SAVE(NP5)
MM8 = MM7+N140	WORK(MM8)	DUM(NP5)
MM9 = MM8+N140	WORK(MM9)	CSAVE(NP)
MM10 = MM9+N428T5	WORK(MM10)	NUM(NP5)

MOREQUES (Classifies center test pixel and adds more clusters if they are needed; also initializes REJECT)

CALL MOREQUES(WORK(MM2), WORK(MM3), MAXCLUS, NFCLUS, ND, N428T5, WORK(MM5))

SUBROUTINE MOREQUES(MEANS, TESTS, MAXCLUS, NFCLUS, ND, NTS, REJECT)

MM2, MM3 as before	WORK(MM2)	MEANS(ND, MAXCLUS)
MM5 = 20900	WORK(MM3)	TESTS(ND, NTS)
	WORK(MM5)	REJECT(MAXCLUS)

CLASSIFY (performs a spatially checked per pixel nearest neighbor classification)

CALL CLASSIFY(WORK(MM3), WORK(MM2), WORK(MM4), NR, NC, NZ, ND, WORK(MM5), NFCLUS, UICB, IND, IMGIN, IMGCLAS, MAXCLUS, COUNT, MASK)

SUBROUTINE CLASSIFY(PIXELS, CLUSTERS, LABELS, NR, NC, NZ, ND, REJECT, NFCLUS, UICB, IND, IMGIN, IMGCLAS, MAXCLUS, COUNT, MASK)

MM2 as before	WORK(MM3)	PIXELS(NC,ND,3)
MM3 = MM2+N288*ND	WORK(MM2)	CLUSTERS(ND,MAXCLUS)
MM4 = MM3+NC*ND*3	WORK(MM4)	LABELS(NC,3)
MM5 as before	WORK(MM5)	REJECT(MAXCLUS)
COUNT is INTEGER*4	COUNT(100)	COUNT(100)

These comments should make it easier to follow subroutine linkage and memory management.

Organization of Detailed Documentation. There are only five sub-routines with logic complex enough to require a detailed description of the algorithm. These are THRFND, START, ASELECT, THINTSTM, and NUMCLU, and will receive more attention in the documentation which follows. As for the main program, IDIMS parameter prompting and file management is easily followed from the source. Because of the elaborate interface IDIMS puts between the user and the outside world, the I/O portion of the program is at least three times the length it would be in a normal FORTRAN environment. But mere length does not make a program hard to understand, and the main

program of AMOEBA is truly self-documenting. The remainder of this document consists of a listing of the main program followed by documentation of each subroutine (in alphabetical order). Appendix A contains a detailed description of the theoretical foundation of the program, Appendix B contains a summary of each of the system subroutines used in AMOEBA, with references to IDIMS and HP documentation, Appendix C contains IDIMS User documentation, Appendix D shows the listing obtained in an interactive sample use of the function, and Appendix E shows a batch job to use the function.

Acknowledgement. We would like to express our gratitude to each of the many scientists who took time to evaluate the results of AMOEBA clustering. Their suggestions and critical remarks led to several improvements.

2. MAIN

Program AMOEBA Listing

PRECEDING PAGE BLANK NOT FILMED



ORIGINAL PAGE IS  
OF POOR QUALITY

PAGE 0002 AMOEBA

```

00715 00050000      NDDS = UICB(62)
00720 00059000      IF (NIDS.NE.1) CALL PABORT(UICB,1,0)
00733 00060000      IF (NDDS.NE.1) CALL PABORT(UICB,1,0)
00746 00061000      C CHECK IF STATFILE NAME IS NOT SUPPLIED
00746 00062000      IF (CODES(1)(011).EQ.0) GO TO 1009
00754 00063000      DO 3 I = 0,1,-1
00761 00064000      IF (SHANE(I+1).NE.' ') GO TO 6
01002 00065000      3 CONTINUE
01002 00066000      CALL PABORT(UICB,-63,0)
01013 00067000      6 SHANE(I+117) = 'STATS '
01032 00068000      IA = 1
01034 00069000      IB = 1
01036 00070000      RECSIZE = 140 + 2*ND
01042 00071000      C OPEN STATFILE AS OLD TO CHECK FOR DUPLICATE NAME
01042 00072000      FILNUM = FOPEN(SHANE,LA,LB,RECSIZE)
01053 00073000      CALL FCHECK(0,ERR,..)
01060 00074000      IF (ERR.EQ.0) CALL PABORT(UICB,45,0)
01073 00075000      IA = 0
01075 00076000      FILNUM = FOPEN(SHANE,LA,LB,RECSIZE)
01106 00077000      CALL FCHECK(0,ERR,..)
01113 00078000      IF (ERR.EQ.0) GO TO 1110
01120 00079000      CALL PABORT(UICB,45,0)
01130 00080000      1009 STATFILE = .FALSE.
01137 00081000      1110 CONTINUE
01137 00082000      C-- ASSIGN LOGICALS
01137 00083000      IF (EQCHMAP.EQ.'Y') CHANMAP = .TRUE.
01144 00084000      IF (EQLMAP.EQ.'Y') LABLMAP = .TRUE.
01156 00085000      IF (EQCLMAP.EQ.'Y') CLASHAP = .TRUE.
01170 00086000      IF (EQMASK.EQ.'N') MASK = .FALSE.
01202 00087000      C -- DO MAPPING PARAMS
01202 00088000      INGIN = 1
01204 00089000      INGCAS = -1
01206 00090000      C
01206 00091000      C OPEN INPUT TO GET ND & NC, THEN CLOSE
01206 00092000      CALL OPENFI(UICB,IND,INGIN,INTYPE,ND,NR,NC,1)
01222 00093000      IF (IND(1).LT.0) CALL CNKID(UICB,IND,INGIN,ND,NR,NC,100)
01243 00094000      IF (ND.GT.16.OR.ND.LT.2) CALL PABORT(UICB,48,0)
01261 00095000      OUTTYPE = 2
01262 00096000      CALL CLOSEP(UICB,IND,INGIN,0)
01274 00097000      IF (IND(1).LT.0) CALL CNKID(UICB,IND,INGIN,1,2,3,99)
01317 00098000      IF (.NOT.(CHANMAP.OR.LABLMAP.OR.CLASHAP)) GO TO 8
01326 00099000      NPARS = 3
01330 00100000      PRINTSL = 1
01332 00101000      PRINTNL = NR
01334 00102000      PRINTSS = 1
01336 00103000      ECHAN = 'PRINTSL PRINTNL PRINTSS '
01362 00104000      CODES(1) = 21
01363 00105000      CODES(2) = 21
01370 00106000      CODE(1) = 21
01373 00107000      ADDR(1) = IADDRESS(PRINTSL)
01401 00108000      ADDR(2) = IADDRESS(PRINTNL)
01405 00109000      ADDR(3) = IADDRESS(PRINTSS)
01412 00110000      CALL PARANS(UICB,NAVES,CODES,ADDRS,NPARS)
01424 00111000      8 CONTINUE
01424 00112000      C
01424 00113000      C TAKE CARE OF WIDE IMAGES:
01424 00114000      C

```

ORIGINAL PAGE IS  
OF POOR QUALITY

PAGE 0003 ANDEBA

```

01424 00115000 C  NZ IS ACTUAL WIDTH
01424 00116000 C  NC IS TARGET NUMBER GRABBED AT A TIME
01424 00117000 C  EACH SUBROUTINE MUST MANAGE ACTUAL PARAMETERS FOR
01424 00118000 C  READP, WRITEP, AND THEIR SUBROUTINES.
01424 00119000      NZ = NC
01426 00120000      NCT = 20000/(3*(ND+1))
01434 00121000      DO 1001 I = 1,99
01441 00122000          NC = NZ/I+1
01446 00123000          IF (NC LE NCT) GO TO 1002
01494 00124000      1001 CONTINUE
01455 00125000      1002 CONTINUE
01455 00126000 C
01455 00127000 C  INITIAL ESTIMATES FOR BUFFERS
01455 00128000 C  NBR -- NUMBER OF BUFFERS ON READ
01455 00129000 C  NBU -- NUMBER OF BUFFERS ON WRITE
01455 00130000 C  GET AS MANY READ AS POSSIBLE!
01455 00131000      NBR = MIN(ND+2,2)
01464 00132000      NBU = 2
01466 00133000 C
01466 00134000 C  OPEN IMAGES FOR REAL NOW
01466 00135000 C  OPEN OUTPUT IMAGE
01466 00136000      CALL OPENPOC(UCB,IND,IMGCLAS,ZERO,OUTTYPE,1,NR,NZ,NBU)
01500 00137000      IF (IND(1).LT.0) CALL CHKIO(UCB,IND,IMGCLAS,NR,NZ,NBU,101)
01523 00138000 C  OPEN INPUT -- TRY UNTIL NBR IS 2
01523 00139000      994 CALL OPENPIC(UCB,IND,INGIN,INTYPE,ND,NR,NZ,NBR)
01540 00140000      IF (IND(1).GE.0) GO TO 996
01545 00141000 C  OUT OF VIRTUAL MEMORY?
01545 00142000      IF (IND(2).NE.97) CALL CHKIO(UCB,IND,INGIN,
01545 00143000      * NBR,NBU,0,102)
01566 00144000      IF (NBR LE 2) GO TO 995
01572 00145000      NBR = NBR - 1
01573 00146000      GO TO 994
01574 00147000 C  NBR IS LESS THAN OR EQUAL TO 2, SO...
01574 00148000      995 IF (NBU EQ 1 AND NBR EQ 1) CALL PABORT(UCB,17,0)
01612 00149000      IF (NBU EQ 2) GO TO 993
01616 00150000      NBR = 1
01620 00151000      993 NBU = 1
01622 00152000 C  REDO OUTPUT WITH FEWER BUFFERS & TRY AGAIN
01622 00153000      CALL CLOSEPIC(UCB,IND,IMGCLAS,1)
01673 00154000      IF (IND(1).LT.0) CALL CHKIO(UCB,IND,IMGCLAS,0,0,0,89)
01673 00155000      CALL OPENPOC(UCB,IND,IMGCLAS,ZERO,OUTTYPE,1,NR,NZ,NBU)
01674 00156000      IF (IND(1).LT.0) CALL CHKIO(UCB,IND,IMGCLAS,NZ,NBU,NBR,103)
01714 00157000      GO TO 994
01715 00158000 C  CAN'T DO IT - INPUT IMAGE TOO BIG
01715 00159000      997 CALL PABORT(UCB,17,0)
01723 00160000 C  -- ALL OPENS COMPLETE AND SUCCESSFUL
01725 00161000      996 CONTINUE
01725 00162000 C
01725 00163000 C  TELL ME HOW MANY BUFFERS I GOT
01725 00164000      WRITE (THING,1010) NBR,NBU
01750 00165000      1010 FORMAT('M YOU HAVE,13,19M READ BUFFER(S) AND,12,
01750 00166000      * 17M WRITE BUFFER(S) )
01751 00167000      CALL PRINTP(UCB,IND,1,THINGEQ,50,0,0,0,0,0,0,0,0)
01777 00168000 C  JOP(701) IS INTNR -- USED BY START
01777 00169000 C
01777 00170000 C  DETERMINE IF THE DATA CONTAINS ANY VALUE GE 128
01777 00171000 C  FIRST FEEL THE DATA

```

PAGE 0004 AMOEBA

```

01777 00172000      NRS = 1+NR/99
02003 00173000      NCS = 1+NZ/199
02007 00174000      DO 32 IR = 1, NR, NRS
02014 00175000      DO 32 K = 1, ND
02021 00176000      CALL READP(UICB, IND, INGIN, WORK, 2, 1, IR, 1, NZ, IR, K+1, 1, NZ)
02071 00177000      IF (IND(1).LT.0) CALL CHKIO(UICB, IND, INGIN, IR, NZ, R, 02)
02071 00178000      DO 32 J = 1, NZ, NCS
02076 00179000      IF (WORK(J).GE.128) GO TO 33
02107 00180000      32 CONTINUE
02110 00181000      GO TO 34
02111 00182000      33 WRITE(THING,35)
02130 00183000      35 FORMAT(30H YOUR IMAGE CONTAINS A VALUE OVER 127.)
02130 00184000      CALL PRINTP(UICB, IND, 1, THINGEO, 38, 0, 0, 0, 0, 0, 0, 0)
02157 00185000      WRITE(THING,36)
02175 00186000      36 FORMAT(40H PLEASE USE MAP TO PUT INTO THE RANGE 0-127.)
02175 00187000      CALL PRINTP(UICB, IND, 1, THINGEO, 44, 0, 0, 0, 0, 0, 0, 0)
02224 00188000      CALL PABORT(UICB, 40, 0)
02234 00189000      34 CONTINUE
02234 00190000      X = (20760.-FLOAT(NC+ND+1))/FLOAT(ND)
02245 00191000      NL = IFIX(SORT(X))
02254 00192000      NS = IFIX((21000.-FLOAT(2*NL)-FLOAT(NC+ND+1))/FLOAT(ND+NL))
02273 00193000      C
02273 00194000      C THRFND DOESN'T NEED AS MUCH ROOM...
02273 00195000      NCS = NC
02273 00196000      NC = NC+3
02275 00197000      IF (NC.GT.NZ) NC = NZ
02300 00198000      NR1 = 1
02307 00199000      NR3 = NR1+ND
02312 00200000      NR4 = NR3+ND+NC
02316 00201000      CALL THRFND(NFL, WORK(NR1), WORK(NR3), UICB, IND, WORK(NR4),
02316 00202000      * NF, NC, ND, NASK, INGIN)
02337 00203000      DO 37 I = 1, NC
02344 00204000      IF (WORK(I).LE.0) WORK(I) = 1
02353 00205000      37 CONTINUE
02354 00206000      WRITE(THING,1111) (WORK(K), K = 1, ND)
02404 00207000      1111 FORMAT(' INTNR = ', I6I3)
02404 00208000      CALL PRINTP(UICB, IND, 1, THINGEO, 10+3*ND, 0, 0, 0, 0, 0, 0, 0)
02437 00209000      C
02437 00210000      C RESTORE NC
02437 00211000      NC = NCS
02437 00212000      NR4 = NR3+NC+ND+3
02444 00213000      CALL SETSYR(SYRROL)
02450 00214000      NR5 = NR4+NC+3
02454 00215000      CALL START(WORK(NR1), ND, NR, NC, NZ, WORK(NR3), WORK(NR4), WORK(NR5),
02454 00216000      * UICB, IND, INGIN, INCLAS, LABF, NASK)
02511 00217000      DLAB = LABF
02515 00218000      DLAB = DLAB+32768
02521 00219000      WRITE(THING,2222) DLAB
02542 00220000      2222 FORMAT(' DLABELS = ', I5)
02542 00221000      CALL PRINTP(UICB, IND, 1, THINGEO, 18, 0, 0, 0, 0, 0, 0, 0)
02571 00222000      IF (CHANINAP) CALL HAPP(PRINTSL, PRINTHL, PRINTSS, UICB, IND, INGIN,
02571 00223000      * NF, NZ, SYRROL)
02571 00224000      * NF, NZ, SYRROL)
02611 00225000      2033 IF (LABLHAP) CALL HAPP(PRINTSL, PRINTNL, PRINTSS, UICB, IND, INCLAS,
02611 00226000      * NF, NZ, SYRROL)
02631 00227000      NR2 = NR1+ND
02634 00228000      NR3 = NR2+NL+ND+NS
02641 00229000      PR3 = NR3+NL

```

ORIGINAL PAGE IS  
OF POOR QUALITY

PAGE 0005 AN05BA

```

02644 00230000      MM5 = MM4+ML
02647 00231000      MM6 = MM5+NC+ND
02653 00232000      CALL ASELECT(WORK(MM2),WORK(MM3),WORK(MM4),ML,MS,MR,
02653 00233000      * NC,NZ,ND,NTS,WORK(MM5),WORK(MM6),FILENO,UICB,IND,INGIN,INGCLAS)
02704 00234000      C NTS IS THE NUMBER OF TEST SETS STASHED BY SELECT
02704 00235000      WRITE(THING,3333) NTS
02720 00236000      3333 FORMAT(' 0TSTSTS = ',I5)
02723 00237000      CALL PRINTP(UICB,IND,1,THING,16,0,0,0,0,0,0,0)
02754 00238000      M25 = 25
02756 00239000      M60 = 60
02760 00240000      M140 = 140
02762 00241000      M428 = 3000/(ND+3)
02767 00242000      M308 = M428-40
02772 00243000      M268 = M308-100
02775 00244000      IF (M200.LT.100) F200 = 100
03002 00245000      MM3 = MM2+ND+M428
03006 00246000      MM4 = MM3+ND+M428.5
03013 00247000      MM5 = MM4+M25+ND+5
03020 00248000      MM6 = MM5+M428
03023 00249000      C WORK(MM2) IS NEAR(ND,M140)
03023 00250000      C WORK(MM3) IS TEST(XL(ND,M428+5)
03027 00251000      C HOWEVER, ALLOW FOR M428 IN CASE WE HAVE FEW
03027 00252000      CALL THINTST(WORK(MM2),WORK(MM3),LWORK(MM4),WORK(MM5),
03027 00253000      * WORK(MM6),M25,M60,M200,M140,M308,M428,ND,NTS,
03027 00254000      * FILENO,UICB,IND)
03033 00255000      M428TS = M428+5
03036 00256000      MM4 = MM3+M428TS+ND
03070 00257000      MM5 = MM4+M428
03073 00258000      CALL SORT(WORK(MM3),WORK(MM4),WORK(MM5),ND,M428TS,M428)
03100 00259000      MM5 = MM4+M428TS
03110 00260000      MM6 = MM5+M140
03113 00261000      MM7 = MM6+M140
03116 00262000      MM8 = MM7+M140
03121 00263000      MM9 = MM8+M140
03124 00264000      MM10 = MM9+M428TS
03127 00265000      CALL MURCLU(WORK(MM2),ND,M140,M428TS,WORK(MM3),MFCLUS,
03127 00266000      * MINCLN,WORK(MM4),WORK(MM5),WORK(MM6),WORK(MM7),
03127 00267000      * WORK(MM8),WORK(MM9),UICB,IND,WORK(MM10),MAXCLN)
03164 00268000      MAXCLUS = 98
03166 00269000      MM5 = 20900
03170 00270000      CALL MREQUES(WORK(MM2),WORK(MM3),MAXCLUS,MFCLUS,ND,
03170 00271000      * M428TS,WORK(MM5))
03203 00272000      CALL MSORT(WORK(MM2),ND,MFCLUS,WORK(MM6),WORK(MM7))
03214 00273000      C
03214 00274000      C CLOSE TEST IMAGE AND OPEN CLUSTER MAP
03214 00275000      CALL CLOSEP(UICB,IND,INGCLAS,1)
03220 00276000      IF (IND(1).LT.0) CALL CHKIO(UICB,IND,INGCLAS,0,0,0,77)
03247 00277000      CALL DELUDB(UICB,IND)
03254 00278000      IF (IND(1).LT.0) CALL CHKIO(UICB,IND,INGCLAS,0,0,0,78)
03276 00279000      INGCLAS = 2
03300 00280000      CALL OPENPO(UICB,IND,INGCLAS,ZERO,1,1,MR,NZ,NBW)
03320 00281000      IF (IND(1).LT.0) CALL CHKIO(UICB,IND,INGCLAS,ND,MR,NZ,79)
03340 00282000      MM3 = MM2+ND+M208
03346 00283000      MM4 = MM3+NC+ND+3
03351 00284000      CALL CLASSIFY(WORK(MM3),WORK(MM2),WORK(MM4),MR,NC,NZ,ND,
03351 00285000      * WORK(MM5),MFCLUS,UICB,IND,INGIN,INGCLAS,MAXCLUS,COUNT,MARK)
03401 00286000      MM5 = ND+4+7

```

PAGE 0006 ANOEBA

```

03405 00287000      WRITE(THING,222)NFCLUS
03426 00288000      222 FORMAT(' FINAL NUMBER OF CLUSTERS =',I3)
03426 00289000      CALL PRINTP(UICB,IND,1,THINGE0,31,0,0,0,0,0,0,0)
03455 00290000      DO 10 I = 1,NFCLUS
03462 00291000      IF (COUNT(I+1).LE.0) GO TO 10
03475 00292000      WRITE(THING,333) COUNT(I+1),(WORK(K+1*ND)
03475 00293000      * ,K = 1,ND)
03534 00294000      333 FORMAT(17,16I4)
03534 00295000      CALL PRINTP(UICB,IND,1,THINGE0,HPP,0,0,0,0,0,0,0,0)
03562 00296000      10 CONTINUE
03562 00297000      IF (COUNT(1).EQ.0) GO TO 445
03573 00298000      WRITE(THING,444) COUNT(1)
03616 00299000      444 FORMAT(' THERE ARE ',I7,' UNCLASSIFIED. ')
03616 00300000      CALL PRINTP(UICB,IND,1,THINGE0,32,0,0,0,0,0,0,0)
03642 00301000      445 IF (COUNT(100).EQ.0) GO TO 556
03655 00302000      WRITE(THING,555) COUNT(100)
03700 00303000      555 FORMAT(18H THE MASK CONTAINS ,I7,8H POINTS. )
03700 00304000      CALL PRINTP(UICB,IND,1,THINGE0,33,0,0,0,0,0,0,0)
03727 00305000      556 CONTINUE
03727 00306000      IF (CLASHAP)CALLHAPP (PRINTSL,PRINTNL,PRINTSS,UICB,IND,INGCLAS,
03727 00307000      * NR,NZ,SYMBOL)
03747 00308000      CALL CLOSEP(UICB,IND,INGCLAS,0)
03760 00309000      IF (IND(1).LT.0) CALL CHKIO(UICB,IND,INGCLAS,0,0,0,400)
04002 00310000      CALL CLOSEP(UICB,IND,INGIN,0)
04013 00311000      IF (IND(1).LT.0) CALL CHKIO(UICB,IND,INGIN,0,0,0,300)
04035 00312000      C-- WRITE STAT FILE
04035 00313000      IF (STATFILE) CALL ANSTATS (FILNUM,ND,WORK(NH2),COUNT,NFCLUS,UICB)
04051 00314000      R E T U R N
04051 00315000      END
    
```

SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
ADDRS	INTEGER	ARRAY	0+217 .I	ANOEBA		SUBROUTINE	
ANSTATS		SUBROUTINE		ASELECT		SUBROUTINE	
ATHMDS		SUBROUTINE		BLKSIZE	INTEGER	ARRAY	0+223 .I
CHANIRAF	LOGICAL	SIMPLE VAR	0+2113	CHKIO		SUBROUTINE	
CHRAF	INTEGER	SIMPLE VAR	0+25 .I	CLSHAP	LOGICAL	SIMPLE VAR	0+2107
CLASSIFY		SUBROUTINE		CLNAP	INTEGER	SIMPLE VAR	0+211 .I
CLOSEP		SUBROUTINE		CODES	INTEGER	ARRAY	0+221 .I
COUNT	INTEGER+4	ARRAY	0+220 .I	DELWDS		SUBROUTINE	
DLAB	INTEGER+4	SIMPLE VAR	0+2125	EOCHRAF	CHARACTER	SIMPLE VAR	0+26 .I
EGCLNAP	CHARACTER	SIMPLE VAR	0+212 .I	EOLNAP	CHARACTER	SIMPLE VAR	0+210 .I
EGRAF	CHARACTER	SIMPLE VAR	0+214 .I	ESMAR	CHARACTER	SIMPLE VAR	0+216 .I
ERR	INTEGER	SIMPLE VAR	0+2122	FCHECK		SUBROUTINE	
FILENO	INTEGER	SIMPLE VAR	0+2104	FILNUM	INTEGER	SIMPLE VAR	0+2106
FOPEN	INTEGER	FUNCTION		I	INTEGER	SIMPLE VAR	0+226
IA	INTEGER	SIMPLE VAR	0+227	IADDRESS	INTEGER	FUNCTION	
IB	INTEGER	SIMPLE VAR	0+230	IMASK	INTEGER	SIMPLE VAR	0+213 .I
INGCLAS	INTEGER	SIMPLE VAR	0+231	INGIN	INTEGER	SIMPLE VAR	0+273
INC	INTEGER	ARRAY	0+222 .I	INTYPE	INTEGER	SIMPLE VAR	0+271
IPCBAF	INTEGER	ARRAY	0+224 .I	IF	INTEGER	SIMPLE VAR	0+267
J	INTEGER	SIMPLE VAR	0+270	F	INTEGER	SIMPLE VAR	0+2117
LA	LOGICAL	SIMPLE VAR	0+227	LARF	INTEGER	SIMPLE VAR	0+232

PAGE 0007 ANOEBA

LAELMAP	LOGICAL	SIMPLE VAR	0+236	LB	LOGICAL	SIMPLE VAR	0+230
LHAP	INTEGER	SIMPLE VAR	0+237 .I	LUORK	LOGICAL	ARRAY	0+213 .I
MAFF		SUBROUTINE		MAK	LOGICAL	SIMPLE VAR	0+260
MAXCLN	INTEGER	SIMPLE VAR	0+263	MAXCLUS	INTEGER	SIMPLE VAR	0+276
MINCLN	INTEGER	SIMPLE VAR	0+262	MM1	INTEGER	SIMPLE VAR	0+232
MP10	INTEGER	SIMPLE VAR	0+2103	MM2	INTEGER	SIMPLE VAR	0+233
MM3	INTEGER	SIMPLE VAR	0+234	MM4	INTEGER	SIMPLE VAR	0+235
MM5	INTEGER	SIMPLE VAR	0+237	MM6	INTEGER	SIMPLE VAR	0+241
MP7	INTEGER	SIMPLE VAR	0+244	MM8	INTEGER	SIMPLE VAR	0+245
MP9	INTEGER	SIMPLE VAR	0+250	NOREQUES		SUBROUTINE	
MSOPT		SUBROUTINE		N140	INTEGER	SIMPLE VAR	0+257
M25	INTEGER	SIMPLE VAR	0+275	N200	INTEGER	SIMPLE VAR	0+254
N700	INTEGER	SIMPLE VAR	0+253	N420	INTEGER	SIMPLE VAR	0+2102
N420TS	INTEGER	SIMPLE VAR	0+2116	N60	INTEGER	SIMPLE VAR	0+240
NAMES	INTEGER	ARRAY	0+215 .I	NBR	INTEGER	SIMPLE VAR	0+247
N6V	INTEGER	SIMPLE VAR	0+261	NC	INTEGER	SIMPLE VAR	0+246
NCS	INTEGER	SIMPLE VAR	0+2110	NCT	INTEGER	SIMPLE VAR	0+2112
NC	INTEGER	SIMPLE VAR	0+251	NFCLUS	INTEGER	SIMPLE VAR	0+2120
NFL	INTEGER	SIMPLE VAR	0+2121	NIDS	INTEGER	SIMPLE VAR	0+242
NL	INTEGER	SIMPLE VAR	0+272	MOD5	INTEGER	SIMPLE VAR	0+243
NPARMS	INTEGER	SIMPLE VAR	0+255	NPP	INTEGER	SIMPLE VAR	0+274
NR	INTEGER	SIMPLE VAR	0+2100	NRS	INTEGER	SIMPLE VAR	0+266
NS	INTEGER	SIMPLE VAR	0+2101	NTS	INTEGER	SIMPLE VAR	0+256
NURCLU		SUBROUTINE		NZ	INTEGER	SIMPLE VAR	0+2111
OPENFI		SUBROUTINE		OPENPO		SUBROUTINE	
OUTTYPE	INTEGER	SIMPLE VAR	0+2114	PABORT		SUBROUTINE	
PARAMS		SUBROUTINE		PRINTNL	INTEGER	SIMPLE VAR	0+265
PRINTP		SUBROUTINE		PRINTSL	INTEGER	SIMPLE VAR	0+2105
PPINTSE	INTEGER	SIMPLE VAR	0+2115	READP		SUBROUTINE	
RECSIZE	INTEGER	SIMPLE VAR	0+264	SETSYN		SUBROUTINE	
SHAPE	CHARACTER	SIMPLE VAR	0+22 .I	SHANE0	INTEGER	SIMPLE VAR	0+21 .I
SCRT		SUBROUTINE		SORT	REAL	FUNCTION	
STAPT		SUBROUTINE		STATFILE	LOGICAL	SIMPLE VAR	0+277
SYMBOL	CHARACTER	ARRAY	0+225 .I	THING	CHARACTER	SIMPLE VAR	0+24 .I
THINGR	INTEGER	SIMPLE VAR	0+23 .I	THINTSTN		SUBROUTINE	
THFFND		SUBROUTINE		UIC0	INTEGER	ARRAY	0-25 .I
VCF	INTEGER	ARRAY	0+213 .I	X	REAL	SIMPLE VAR	0+2123
ZERO	INTEGER	ARRAY	0-24 .I				

PROGRAM UNIT ANOEBA COMPILED

**3. DETAILED DOCUMENTATION OF SUBROUTINES**

**PRECEDING PAGE BLANK NOT FILMED**

Parent: MAIN

AMSTATS

AMSTATS(FILNUM,ND,MEAN,COUNT NFCLUS,UICB)

After the classification step, subroutine AMSTATS writes the means of the clusters to a disk file using the standard IDIMS format for statistics files.

Method: AMSTATS receives the means and counts as parameters. It then writes all the means and counts to a previously opened disk file, one mean vector per record. Vectors with count equal zero are not written. The file is written and closed using HP standard intrinsics. Since this statistics file does not contain a covariance matrix, it cannot be used in maximum likelihood classification or ellipse plotting.

Program Variables

BUFFER	INTEGER ARRAY	I/O array
CLASS	CHARACTER	Creator ID for statistics file
CONTROL	LOGICAL	Carriage control bit mask for FWRITE
COUNT	DOUBLE INTEGER	Populations of classes
FCLOSE	INTRINSIC	To close files
FILNUM	INTEGER	File number
FWRITE	INTRINSIC	To write a record
I	INTEGER	Index for number of classes loop
IM	INTEGER	I minus 1
J	INTEGER	Class number counter
K	INTEGER	Index for number of dimensions loop
LBUFF	LOGICAL ARRAY	Equivalenced to I/O array because FWRITE requires a logical array
MEAN	INTEGER ARRAY	Array of mean vectors
ND	INTEGER	Number of dimensions
NFCLUS	INTEGER	Number of final clusters
NFCP	INTEGER	Number of final clusters plus one
NUMPTS	INTEGER	Field that must hold COUNT, overflow possible
PABORT	SYSTEM SUBROUTINE	

REUFF	REAL ARRAY	Equivalenced to I/O buffer to allow stuffing of means in mandatory real format.
RESIZE	INTEGER	Record size
UICB	INTEGER ARRAY	User Information Control Block

PAGE 0016 HEWLETT-PACKARD 321029 01 03 FORTRAN/3000 TUE, OCT 13, 1981, 9:37 AM

```

00024 00493000 8CONTROL SEGMT=ANOEBASEG
00024 00494000 C-- THIS SUBROUTINE WRITES THE STATISTICS FILE AT THE
00024 00495000 C-- COMPLETION OF THE PROGRAM
00024 00496000 SUBROUTINE ANSTATS(FILNUM,ND,NEAN,COUNT,NFCLUS,UICB)
00024 00497000 REAL RBUFF(1)
00024 00498000 SYSTEM INTRINSIC FWRITE,FCLOSE
00024 00499000 INTEGER*2 FILNUM,NEAN(ND,NFCLUS),RECSIZE,
00024 00500000 * BUFFER(220),NUMPTS,UICB(1)
00024 00501000 INTEGER*4 COUNT(100)
00024 00502000 LOGICAL CONTROL,LBUFF(220)
00024 00503000 CHARACTER*8 CLASS
00024 00504000 EQUIVALENCE (BUFFER,CLASS),(BUFFER(5),NUMPTS),
00024 00505000 * (BUFFER(141),RBUFF),(LBUFF,BUFFER)
00024 00506000 RECSIZE = 140+2*ND
00030 00507000 BUFFER(7) = 0
00033 00508000 NFCLP = NFCLUS + 1
00036 00509000 CLASS = 'ANOEBA '
00039 00510000 BUFFER(8) = ND
00037 00511000 CONTROL = .FALSE.
00037 00512000 J = 1
00037 00513000 C CLASS & COUNT 1 ARE UNCLASSIFIED PIXELS
00037 00514000 C
00037 00515000 DO 805 I = 2,NFCLP
00064 00516000 IF (COUNT(I).EQ.0) GO TO 805
00071 00517000 C
00077 00518000 C -- NUMPTS CAN OVERFLOW BUT THE CRUMMY STATFILE
00073 00519000 C -- HAS ONLY AN INTEGER*2 FIELD AVAILABLE
00073 00520000 C
00072 00521000 NUMPTS = COUNT(I)
00103 00522000 BUFFER(9) = J
00106 00523000 J = J + 1
00107 00524000 IM = I - 1
00112 00525000 DO 531 K = 1,ND
00117 00526000 RBUFF(K) = NEAN(K,IM)
00120 00527000 531 CONTINUE
00121 00528000 CALL FWRITE(FILNUM,LBUFF,RECSIZE,CONTROL)
00137 00529000 IF ( CC ) 803,805,803
00141 00530000 803 CALL PABORT(UICB,45,0)
00121 00531000 805 CONTINUE
00152 00532000 CALL FCLOSE(FILNUM,1,0)
00156 00533000 IF ( CC ) 807,809,807
00160 00534000 807 CALL PABORT (UICB,45,0)
00170 00535000 809 CONTINUE
00170 00536000 RETURN
00171 00537000 END

```

SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
ANSTATS		SUBROUTINE		BUFFER	INTEGER	ARRAY	0+23 .1
CLASS	CHARACTER	SIMPLE VAR	0+24 .1	CONTROL	LOGICAL	SIMPLE VAR	0+25
COUNT	INTEGER*4	ARRAY	0-26 .1	FCLOSE		SUBROUTINE	
FILNUM	INTEGER	SIMPLE VAR	0-211 .1	FWRITE		SUBROUTINE	
I	INTEGER	SIMPLE VAR	0+27	IM	INTEGER	SIMPLE VAR	0+210
J	INTEGER	SIMPLE VAR	0+212	K	INTEGER	SIMPLE VAR	0+214
LBUFF	LOGICAL	ARRAY	0+23 .1	NEAN	INTEGER	ARRAY	0-27 .1
ND	INTEGER	SIMPLE VAR	0-210 .1	NFCLUS	INTEGER	SIMPLE VAR	0-25 .1
NFCLP	INTEGER	SIMPLE VAR	0+213	NUMPTS	INTEGER	SIMPLE VAR	0+25 .1
PABORT		SUBROUTINE		RBUFF	REAL	ARRAY	0+26 .1
RECSIZE	INTEGER	SIMPLE VAR	0+211	UICB	INTEGER	ARRAY	0-24 .1

PROGRAM UNIT ANSTATS COMPILED

Parent: MAIN

ASELECT

Calls: CLOSEC

ASELECT(DAT,LAB,KNT,NL,NS,NR,NC,NZ,ND,NTS,DATA,LABEL,FILENO,UICB,  
IND,IMGIN,IMGCLAS)

Subroutine ASELECT takes a label map created by START and extracts test sets. Both the label map and test sets reside in temporary disk files. The test sets are passed to THINTSTM

Method. Using Wide Image Logic (see above), ASELECT segments the image into strips. Each strip is about  $6666/(ND+1)$  elements wide. Within a strip, the labels map is scanned looking for samples having the same label. The data values are collected as encountered in a buffer DAT(NL,ND,NS). One scan line of data and labels, requiring  $NC*(ND+1)$  words of memory, are resident. For each label active, buffers KNT counting how many and LAB pointing to which are required. Thus  $NL*ND*NS+2*NL+NC+(ND+1) \leq 21000$  is required. If  $NL \leq 120$  is estimated, we have  $NL*NS \leq (20760 - NC*(ND+1))/ND$ . We set NL equal to the square root of the right hand side, and NS as large as possible satisfying the first inequality. This memory allocation is performed in the main program. For example, suppose we are processing an image of  $NZ = 2048$  elements wide. For  $ND = 2, 4, 8, 12,$  and  $16$ , we tabulate NC, NL, and NS in Table 1. Even in the worst case, sufficient buffer space is available to collect 31 samples. Note that  $NC/NL$  is relatively constant, as is desirable.  $NC/NL$  is about  $56.2 \sqrt{ND}/(ND+1)$ , and  $\sqrt{ND}/(ND+1)$  varies slowly with ND, e.g. as ND goes from 4 to 16,  $NC/NL$  should vary from 11.6 to 13.2 (these estimates are for very large NZ).

Subroutine CLOSEC is called when:

- (a) The number of elements KNT(J) in buffer J for a particular label equals NS; or,
- (b) A new label is encountered and no slots are available to stash data having that label; or,

## ASLECT-2

- (c) in a new line, an old label is not found; or,
- (d) a line with no labels whatever is found; or,
- (e) when all lines have been processed.

CLOSEC performs the following functions:

- (1) It closes buffer J by setting  $LAB(J) = Z$  ( $Z = -32768$  marks no label).
- (2) It shows another slot available by decrementing NA, the slot available pointer. (No slots are available if  $NA = NL$ .)
- (3) If  $KNT(J)$  is at least 5, it selects five test pixels as spread out as possible and writes them on disk; a count is kept of this event, called NTS in ASELECT.
- (4) It sets  $KNT(J) = 0$ .

In case (a), that slot is made available. Action taken in (b) is to seek the eldest active label, close that one, and then begin the new label here. In case (c), each such buffer is closed (since this label will no longer be encountered). In (d) and (e), all active labels are closed.

In the Wide Image Logic, a boundary is generated when a new strip is started. This prevents the bottom labels of one strip from being joined to the top of the next, and also frees all buffers for a new start.

### Program Variables

CHKIO	SYSTEM SUBROUTINE
CLOSEC	SUBROUTINE Writes test sets on disk after sampling, and frees buffer.
DAT(NL,ND,NS)	INTEGER ARRAY Used for accumulating patches by label (the first variable), dimension (second) and by count.
DATA(NC,ND)	INTEGER ARRAY One line of data.
FILENO	INTEGER The file of test pixels, opened and written by CLOSEC.

I,IREAD,J,JS,K	INTEGER DO loop index.
IMGCLAS	INTEGER Label map file number.
IMGIN	INTEGER Data file number.
IND(1)	INTEGER ARRAY Error indicator.
KN	INTEGER Count number.
INT(NL)	INTEGER ARRAY Running count of number of each label found.
LAB(NL)	INTEGER ARRAY Label of particular slot. Note: For each J, KNT(J) is the number found so far with label LAB(J); these samples are stored in DAT(J,..,1) through DAT(J,..,KNT(J)).
LABEL(NC)	INTEGER ARRAY A line of labels.
LOLD	INTEGER Used in finding oldest active label.
NA	INTEGER Used to indicate when a search for an available slot should be undertaken. When NA = NS, no slots are available.
NC,NW,NX,NY,NZ	INTEGER Used in Wide Image Logic.
ND	INTEGER Dimensionality.
ND5	INTEGER ND*5; used as a dimension parameter for CLOSEC.
NL	INTEGER Number of labels collected at once.
NR	INTEGER Number of lines.
NS	INTEGER Number of samples for each label.
NTS	INTEGER Number of test sets written.
READP	SYSTEM SUBROUTINE
TSP(80)	LOGICAL ARRAY Buffer for writing test sets to disk in CLOSEC.

UICB(1)            INTEGER ARRAY User Information Control Block  
 Z                    INTEGER Boundary marker: -32768.

Table 1. Example of Memory Allocation for ASELECT  
 Number of Samples = 2048

ND	NC	NL	NS
2	2048	85	86
4	1025	62	63
8	683	42	43
12	513	34	34
16	342	30	31

PAGE 0020 HEWLETT-PACKARD 321020 01 03 FORTRAN/3000 TUE, OCT 13, 1981, 9:30 AM

```

00020 00040000 SCONTROL SEGMENT=ANDBASED
00020 00041000 SUBROUTINE ASELECT(DAT,LAB,KNT,NL,ND,NR,NC,NZ,NO,NTS,DATA,LABEL,
00020 00042000 * FILENO,UICB,IND,INCLAS)
00020 00043000 INTEGER*2 DAT(NL,ND,NS),LAB(NL),KNT(NL),DATA(NC,ND),LABEL(NC),
00020 00044000 * Z,FILENO,UICB(1),IND(1)
00020 00045000 LOGICAL CONTROL,TSP(00)
00020 00046000 C
00020 00047000 C
00020 00048000 C PAPAPETERS:
00020 00049000 C
00020 00050000 C TSP -- BUFFER FOR ACLOSEC
00020 00051000 C DAT -- DATA BEING SAVED
00020 00052000 C LAB -- LABEL VECTOR
00020 00053000 C KNT -- COUNT VECTOR
00020 00054000 C KNT(1) = 0 MEANS SLOT 1 IS FREE
00020 00055000 C NL -- MAX NUMBER OF LABELS
00020 00056000 C NS -- MAX SAMPLES PER PATCH
00020 00057000 C ND -- DIMENSIONALITY
00020 00058000 C NC -- ELEMENTS PER SCAN LINE
00020 00059000 C NZ -- ACTUAL NUMBER OF ELEMENTS PER SCAN LINE
00020 00060000 C NU -- STARTING ELEMENT READ
00020 00061000 C NY -- LAST ELEMENT READ
00020 00062000 C NX -- NUMBER READ (NC IS TARGET, BUT NX IS ACTUAL NUMBER)
00020 00063000 C NR -- NUMBER OF SCAN LINES
00020 00064000 C NA -- NUMBER OF LABEL SLOTS BEING USED
00020 00065000 C NTS -- RUNNING COUNT OF NUMBER OF TEST SETS
00020 00066000 C
00020 00067000 C INITIALIZE
00020 00068000 ND5 = ND*3
00023 00069000 Z = -32768
00032 00070000 CONTROL = .TRUE.
00034 00071000 NTS = 0
00036 00072000 NA = 0
00040 00073000 DO 10 I = 1,NL
00043 00074000 KNT(I) = 0
00030 00075000 10 LAB(I) = Z
00034 00076000 C
00034 00077000 C PROCESS BY STRIPS ABOUT NC WIDE
00034 00078000 DO 96 NU = 1,NZ,NC
00061 00079000 NY = NU*NC-1
00063 00080000 IF (NY GT NZ) NY = NZ
00072 00081000 NX = NY-NU+1
00076 00082000 C
00076 00083000 C PROCESS BY SCAN LINES
00076 00084000 DO 500 IREAD = 1,NR
00103 00085000 C
00103 00086000 C SEE IF START OF A NEW STRIP
00103 00087000 IF (IREAD GT 1 OR NU EQ 1) GO TO 97
00114 00088000 DO 90 I = 1,NX
00121 00089000 90 LABEL(I) = Z
00123 00090000 GO TO 93
00126 00091000 C
00126 00092000 C READ LABELS
00126 00093000 97 CALL READP(UICB,IND,INCLAS,LABEL,Z,I,IREAD,NU,NX,
00126 00094000 * I,IREAD+1,NU,NX)
00133 00095000 IF (IND(I) LT 0) CALL CHKIO(UICB,IND,INCLAS,IREAD,NU,NZ,325)
00173 00096000 C

```

PAGE 0029 ASELECT

```

00175 00097000 C SEE IF THERE WAS A LABEL THIS LINE.
00175 00098000 95 00 50 I = 1,NX
00202 00099000 IF (LABEL(I).NE.Z) GO TO 99
00211 00900000 50 CONTINUE
00212 00901000 C
00212 00902000 C SEE IF ANY ACTIVE.
00212 00903000 IF (NA.EQ.0) GO TO 500
00217 00904000 C
00217 00905000 C CLOSE ALL ACTIVE LABELS.
00217 00906000 DO 60 J = 1,NL
00224 00907000 IF (LAB(J).NE.Z) CALL CLOSEC(Z,J,TSP,LAB,KNT,DAT,NL,
00224 00908000 * NS,ND,NA,NTS,FILENO,UICB,IND,CONTROL,NDS)
00257 00909000 60 CONTINUE
00260 00910000 GO TO 500
00261 00911000 C
00261 00912000 C CHECK FOR INACTIVE OLD LABEL
00261 00913000 99 00 100 J = 1,NL
00266 00914000 L = LAB(J)
00271 00915000 IF (L.EQ.Z) GO TO 100
00275 00916000 DO 110 I = 1,NX
00302 00917000 IF (LABEL(I).EQ.L) GO TO 100
00310 00918000 110 CONTINUE
00311 00919000 CALL CLOSEC(Z,J,TSP,LAB,KNT,DAT,NL,NS,ND,NA,NTS,
00311 00920000 * FILENO,UICB,IND,CONTROL,NDS)
00340 00921000 100 CONTINUE
00341 00922000 C
00341 00923000 C POINT TO START
00341 00924000 J = 1
00343 00925000 C
00343 00926000 C READ DATA
00343 00927000 DO 407 K = 1,ND
00350 00928000 CALL READP(UICB,IND,INGIN,DATA(1,K),2,K,I,READ,NU,NX,
00350 00929000 * K+1,I,READ,NU,NX)
00400 00930000 IF (IND(1).LT.0) CALL CMKIO(UICB,IND,INGIN,K,I,READ,NC,530)
00420 00931000 607 CONTINUE
00421 00932000 C PROCESS CURRENT SCAN LINE
00421 00933000 C
00421 00934000 DO 200 I = 1,NX
00426 00935000 L = LABEL(I)
00431 00936000 IF (L.EQ.Z) GO TO 200
00436 00937000 C
00436 00938000 C LABEL FOUND. LOOK FOR A DUPE
00436 00939000 IF (LAB(J).EQ.L) GO TO 210
00443 00940000 DO 220 J = 1,NL
00450 00941000 IF (LAB(J).EQ.L) GO TO 210
00456 00942000 220 CONTINUE
00457 00943000 C
00457 00944000 C FELL THROUGH -- NO MATCH FOUND
00457 00945000 C CHECK IF THERE IS ROOM
00457 00946000 IF (NA.LT.NL) GO TO 300
00463 00947000 C
00463 00948000 C NO ROOM. CLOSE THE ELDEST
00463 00949000 LOLD = LAB(I)
00466 00950000 J = 1
00470 00951000 DO 310 JS = 2,NL
00475 00952000 IF (LAB(JS) GE LOLD) GO TO 310
00502 00953000 J = JS

```

ORIGINAL FILE IS  
OF POOR QUALITY

PAGE 0030 ASELECT

```

00304 00954000          LOLO = LAB(JS)
00307 00955000      310 CONTINUE
00310 00956000      C
00310 00957000      C      OLDEST IS POINTED TO BY J
00310 00958000      CALL CLOSEC(Z,J,TSP,LAB,KNT,DAT,NL,NS,ND,NA,NTS,
00310 00959000          * FILENO,UICB,IND,CONTROL,ND3)
00337 00960000          GO TO 330
00340 00961000      C
00340 00962000      C      CATCH UP ON LOGIC:  ROOM FOR HERE HERE
00340 00963000      C
00340 00964000      C      FIND A SLOT
00340 00965000      DO 300 300 J = 1,NL
00343 00966000          IF (LAB(J).EQ.Z) GO TO 300
00353 00967000      320 CONTINUE
00354 00968000      330 KNT(J) = 0
00357 00969000          LAB(J) = L
00362 00970000          NA = NA+1
00363 00971000      210 KN = KNT(J)+1
00367 00972000          KNT(J) = KN
00372 00973000          DO 400 K = 1,ND
00377 00974000          400 DAT(KN,K,J) = DAT(I,K)
00617 00975000      C
00617 00976000      C      CLOSE IF KN = NL
00617 00977000          IF(KN.EQ.NL) CAL  CLOSEC(Z,J,TSP,LAB,KNT,DAT,NL,NS,ND,NA,NTS,
00617 00978000          * FILENO,UICB,IND,CONTROL,ND3)
00651 00979000      C
00651 00980000      C      COLUMN LOOP END
00651 00981000      200 CONTINUE
00652 00982000      C      SCAN LOOP END
00652 00983000      300 CONTINUE
00656 00984000      C
00656 00985000      C      STRIP LOOP END
00656 00986000      96 CONTINUE
00662 00987000      C
00662 00988000      C      CLOSE EVERYTHING IN SIGHT
00662 00989000          DO 600 J = 1,NL
00667 00990000          IF(LAB(J).NE.Z)CALL CLOSEC(Z,J,TSP,LAB,KNT,DAT,NL,NS,ND,NA,NTS,
00667 00991000          * FILENO,UICB,IND,CONTROL,ND3)
00722 00992000          600 CONTINUE
00723 00993000          R E T U R N
00724 00994000          END

```

## SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
ASELECT		SUBROUTINE		CHKIO		SUBROUTINE	
CLOSEC		SUBROUTINE		CONTROL		SUBROUTINE	
DAT	INTEGER	ARRAY	0-224 .I	DATA	LOGICAL	SIMPLE VAR	0-223
FILENO	INTEGER	SIMPLE VAR	0-210 .I	I	INTEGER	ARRAY	0-212 .I
INGCLAS	INTEGER	SIMPLE VAR	0-24 .I	IND	INTEGER	SIMPLE VAR	0-26
IND	INTEGER	ARRAY	0-26 .I	INGIN	INTEGER	SIMPLE VAR	0-25 .I
J	INTEGER	SIMPLE VAR	0-212	IREAD	INTEGER	SIMPLE VAR	0-215
K	INTEGER	SIMPLE VAR	0-221	JS	INTEGER	SIMPLE VAR	0-211
KNT	INTEGER	ARRAY	0-222 .I	KN	INTEGER	SIMPLE VAR	0-224
LAB	INTEGER	ARRAY	0-223 .I	L	INTEGER	SIMPLE VAR	0-210
LOLO	INTEGER	SIMPLE VAR	0-217	LABEL	INTEGER	ARRAY	0-211 .I
NC	INTEGER	SIMPLE VAR	0-216 .I	NA	INTEGER	SIMPLE VAR	0-27
ND3	INTEGER	SIMPLE VAR	0-213	ND	INTEGER	SIMPLE VAR	0-214 .I
NR	INTEGER	SIMPLE VAR	0-217 .I	NL	INTEGER	SIMPLE VAR	0-221 .I
NTS	INTEGER	SIMPLE VAR	0-213 .I	NS	INTEGER	SIMPLE VAR	0-220 .I
NA	INTEGER	SIMPLE VAR	0-216	NU	INTEGER	SIMPLE VAR	0-214
NZ	INTEGER	SIMPLE VAR	0-215 .I	NY	INTEGER	SIMPLE VAR	0-220
TSP	LOGICAL	ARRAY	0-25 .I	READP		SUBROUTINE	
Z	INTEGER	SIMPLE VAR	0-222	UICB	INTEGER	ARRAY	0-27 .I

PROGRAM UNIT ASELECT COMPILED

Parent: MAIN

CLASSIFY

Calls: PERPIXEL, MARKUP, FIXUP

CLASSIFY(PIXELS,CLUSTERS,LABELS,NR,NC,NZ,ND,REJECT,NFCLUS,UICB,IND,  
IMAGE,IMGCLAS,MAXCLUS,COUNT,MASK)

This subroutine performs a spatially supervised classification of multi-image data. The underlying spectral classifier is a nearest neighbor (Euclidean distance) per pixel classifier. Such a classifier behaves poorly on mixtures: a point on the spatial boundary between classes will sometimes be classified in another actual class. In CLASSIFY, the mildest possible reclassification is performed: only points classified unlike all four of their neighbors are reclassified, and even these are left alone if the nearest class of a neighbor is too far away.

Method: CLASSIFY uses: Wide Image Logic, the Mask, Four Neighbors, Circular Buffers, and Rejection Thresholds. These concepts are documented separately. Assuming they are understood, the method can be described briefly. In each (wide image) strip of data, a circular buffer of three scan lines of data and labels is formed. Initially, all three lines are classified (subroutine PERPIXEL). Then a big loop is entered (label 30), and the center, pointed to by I2, is marked (subroutine MARKUP) to indicate pixels classified like at least one neighbor. Subroutine FIXUP is entered to reclassify unmarked pixels like one of their solidly classified neighbors. (These subroutines could, of course, be differently restrictive.) Then the eldest label line, pointed to by I1, is written on disk, the buffer is rotated, and a new line of data is read and classified. When no more data can be read, the last two lines of labels are written, and the next Wide Image Strip is processed.

A count is kept of the number in each class. This is returned in vector COUNT, a long integer array. COUNT(1) is reserved for unclassified elements (which appear on disk with label 0). COUNT(2) through COUNT(99) are the number in class 1 through 98. COUNT(100) is the number of points in the Mask, given label 99 on disk.

The classification routine may introduce new classes; because the data is only scanned once, the new classes will not be attractors until they are formed. The reason for introducing new classes lies in the profound unpopularity of unclassified pixels, as well as the stubborn adherence to the mixture model. Since these classes are usually small, the percentage of errors is likely to be tiny.

### Program Variables

CHKIO                    SYSTEM SUBROUTINE

CLUSTERS(ND,MAXCLUS)    INTEGER ARRAY    The cluster centers or attractors. Their actual number is NFCLUS, which may be changed in PERPIXEL.

COUNT(100)              LONG INTEGER ARRAY    The count of the number in each cluster. COUNT(1) is the number unclassified, COUNT(100) is the number in the mask, and COUNT(I) is the number in the cluster I-1 for I = 2,...,99.

FIXUP                    SUBROUTINE    Processes points classified unlike each of their four neighbors, attempting reclassification according to the mixture model.

I                        INTEGER    DO loop index

I1,I2,I3                INTEGER    Circular buffer pointers

IMAGE                    INTEGER    Input image number.

IMGCLAS                 INTEGER    Output image number.

IND(1)                  INTEGER ARRAY    Error indicator.

IREAD,IROW              INTEGER    Line numbers on read and write, managed in each strip.

IT                        INTEGER    Scratch variable, used to rotate buffer.

JC                        INTEGER    Used to index into COUNT while counting LABELS.

JJ,K                     INTEGER    DO loop index.

LABELS(NC,3)      INTEGER ARRAY    Circular buffer of classifications.

MARKUP            SUBROUTINE    Adds 101 to the center label when that label is like at least one of the four neighbors.

MASK              LOGICAL       If .TRUE., a value of 0 in channel 1 of the data is classified "mask" and labelled 99.

MAXCLUS           INTEGER       The maximum number of clusters allowed.

NC                INTEGER       Width of each strip.

ND                INTEGER       Dimensionality of data.

NFCLUS            INTEGER       Dynamic number of clusters.

NR                INTEGER       Number of lines.

NW,NX,NY,NZ       INTEGER       Used, in loop 96, to segment the image into strips.

PERPIXEL          SUBROUTINE    Performs a per pixel nearest neighbor classification. Introduces new clusters when the nearest neighbor is too far away to fit the mixture model.

PIXELS(NC,ND,3)   INTEGER ARRAY   One circular buffer of data in a strip.

READP            SYSTEM SUBROUTINE

REJECT(MAXCLUS)   INTEGER ARRAY   The rejection thresholds.

UICB(1)           INTEGER ARRAY   User Information Control Block

WRITEP            SYSTEM SUBROUTINE

PAGE 0010 HEWLETT-PACKARD J21020.01.03 FORTRAN/3000 TUE, OCT 13, 1981, 9:37 AM

```

00010 00538000 #CONTROL SEGMENT=ANOEBASEC
00010 00539000 SUBROUTINE CLASSIFY(PIXELS,CLUSTERS,LABELS,NR,NC,NZ,ND,REJECT,
00010 00540000 * NFCLUS,UICB,IND,IMAGE,INCLAS,MAXCLUS,COUNT,MASK)
00010 00541000 INTEGER*2 UICB(1),IND(1),REJECT(MAXCLUS),CLUSTERS(ND,MAXCLUS),
00010 00542000 * PIXELS(NC,ND,3),LABELS(NC,3)
00010 00543000 INTEGER*4 COUNT(100)
00010 00544000 LOGICAL MASK
00010 00545000 DO 1 I = 1,100
00022 00546000 1 COUNT(I) = 0
00027 00547000 C
00027 00548000 C WIDE IMAGE LOGIC
00027 00549000 C NU IS STARTING COL IN IMAGE
00027 00550000 C NY IS ENDING
00027 00551000 C NX IS ACTUAL NUMBLR READ
00027 00552000 C
00027 00553000 DO 96 NU = 1,NZ,NC
00034 00554000 NY = NU+NC-1
00040 00555000 IF (NY.GT.NZ) NY = NZ
00045 00556000 NX = NY-NU+1
00051 00557000 C
00051 00558000 C SET READ/WRITE COUNTERS
00051 00559000 C
00051 00560000 IREAD = 3
00051 00561000 IROW = 1
00051 00562000 C
00051 00563000 C SET UP CIRCULAR BUFFER POINTER
00051 00564000 C
00051 00565000 I1 = 1
00057 00566000 I2 = 2
00061 00567000 I3 = 3
00061 00568000 C
00061 00569000 C GET STARTED: READ 3 SCAN LINES DATA
00061 00570000 C
00061 00571000 DO 20 I = 1,3
00070 00572000 DO 10 J = 1,NC
00070 00573000 CALL READP(UICB,IND,IMAGE,PIX 3(1,K,I),2,K,I,NU,
00070 00574000 * NY,K+1,I,NU,NX)
00130 00575000 IF (IND(I).LT.0) CALL CHA1(UICB,IND,IMAGE,I,K,NC,I1)
00150 00576000 10 CONTINUE
00151 00577000 C
00151 00578000 C CLASSIFY FIRST THREE SCAN LINES
00151 00579000 C
00151 00580000 20 CALL PERPIXEL(PIXELS(1,1,1),CLUSTERS,LABELS(1,1),
00151 00581000 * ND,NC,NFCLUS,REJECT,MAXCLUS,MASK,NX)
00201 00582000 C
00201 00583000 C REFERENCE FOR BIG LOOP
00201 00584000 C
00201 00585000 C MARK PIXELS CLASSED LIKE THEIR NEIGHBORS
00201 00586000 C
00201 00587000 30 CALL MARKUP(LABELS,NC,I1,I2,I3,NX)
00211 00588000 C
00211 00589000 C USE CONTEXT TO ATTEMPT RECLASSIFICATION
00211 00590000 C
00211 00591000 CALL FIXUP(LABELS,NC,I1,I2,I3,PIXELS(1,1,I2),REJECT,
00211 00592000 * CLUSTERS,ND,MAXCLUS,NX)
00236 00593000 C
00236 00594000 C WRITE A SCAN LINE OF LABELS

```

ORIGINAL PAGE IS  
OF POOR QUALITY

PAGE 0019 CLASSIFY

```

00236 00395000 C
00236 00396000 CALL WRITEP(UICB,IND,INGCLAS,LABELS(1,11),2,1,IR0V,NU,NX,
00236 00397000 * 1,IR0V+1,NU,NX)
00270 00398000 IF (IND(1).LT.0) CALL CHKIO(UICB,IND,INGCLAS,IR0V,0,0,21)
00311 00399000 DO 2 JJ = 1,NX
00316 00600000 JC = LABELS(JJ,11)+1
00326 00601000 2 COUNT(JC) = COUNT(JC)+1
00336 00602000 C
00336 00603000 C POINT TO NEXT SCAN LINE
00336 00604000 C
00376 00605000 IR0V = IR0V+1
00377 00606000 IREAD = IREAD+1
00340 00607000 IF (IREAD GT NR) GO TO 1400
00344 00608000 C
00344 00609000 C GRAB ANOTHER SCAN LINE
00344 00610000 C
00344 00611000 DO 40 K = 1,ND
00351 00612000 CALL READP(UICB,IND,IMAGE,PIXELS(1,K,11),2,K,IREAD,
00351 00613000 * NU,NX,K+1,IREAD,NU,NX)
00400 00614000 IF (IND(1).LT.0) CALL CHKIO(UICB,IND,IMAGE,K,11,IREAD,263)
00420 00615000 40 CONTINUE
00420 00616000 C
00420 00617000 C ROTATE CIRCULAR BUFFER
00420 00618000 C
00420 00619000 IT = 11
00427 00620000 II = 12
00431 00621000 I2 = 13
00437 00622000 I3 = IT
00439 00623000 C
00439 00624000 C CLASSIFY THE NEW CRITTER
00439 00625000 C
00439 00626000 CALL PERPIXEL(PIXELS(1,1,13),CLUSTERS,LABELS(1,13),ND,NC,
00439 00627000 * NPCLJS,REJECT,MAXCLUS,NAK,NX)
00464 00628000 GO TO 30
00466 00629000 C
00466 00630000 C FINISH UP
00466 00631000 C
00466 00632000 1000 CONTINUE
00466 00633000 CALL WRITEP(UICB,IND,INGCLAS,LABELS(1,12),2,1,IR0V,
00466 00634000 * NU,NX,1,IR0V+1,NU,NX)
00520 00635000 IF (IND(1).LT.0) CALL CHKIO(UICB,IND,INGCLAS,12,IR0V,NC,202)
00540 00636000 DO 3 JJ = 1,NX
00540 00637000 JC = LABELS(JJ,12)+1
00555 00638000 3 COUNT(JC) = COUNT(JC)+1
00560 00639000 CALL WRITEP(UICB,IND,INGCLAS,LABELS(1,13),2,1,IR0V+1,
00560 00640000 * NU,NX,1,1,1,1)
00622 00641000 IF (IND(1).LT.0) CALL CHKIO(UICB,IND,INGCLAS,13,IR0V,NC,207)
00642 00642000 DO 4 JJ = 1,NX
00647 00643000 JC = LABELS(JJ,13)+1
00657 00644000 4 COUNT(JC) = COUNT(JC)+1
00667 00645000 96 CONTINUE
00670 00646000 R E T U R N
00676 00647000 END

```

PAGE 0020 CLASSIFY

## SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
CMN10		SUBROUTINE		CLASSIFY		SUBROUTINE	
CLUSTERS	INTEGER	ARRAY	0-222 .I	COUNT	INTEGER**4	ARRAY	0-225 .I
FIXUP		SUBROUTINE		I	INTEGER	SIMPLE VAR	0-224
I1	INTEGER	SIMPLE VAR	0-214	I2	INTEGER	SIMPLE VAR	0-215
I3	INTEGER	SIMPLE VAR	0-216	IMAGE	INTEGER	SIMPLE VAR	0-210 .I
INCLAS	INTEGER	SIMPLE VAR	0-27 .I	IND	INTEGER	ARRAY	0-211 .I
IREAD	INTEGER	SIMPLE VAR	0-211	IRON	INTEGER	SIMPLE VAR	0-26
IT	INTEGER	SIMPLE VAR	0-27	JC	INTEGER	SIMPLE VAR	0-220
JJ	INTEGER	SIMPLE VAR	0-25	K	INTEGER	SIMPLE VAR	0-217
LABELS	INTEGER	ARRAY	0-221 .I	MARKUP		SUBROUTINE	
NASP	LOGICAL	SIMPLE VAR	0-24 .I	MAXCLUS	INTEGER	SIMPLE VAR	0-26 .I
NC	INTEGER	SIMPLE VAR	0-217 .I	ND	INTEGER	SIMPLE VAR	0-213 .I
MFCLUS	INTEGER	SIMPLE VAR	0-213 .I	NR	INTEGER	SIMPLE VAR	0-220 .I
NV	INTEGER	SIMPLE VAR	0-210	NX	INTEGER	SIMPLE VAR	0-212
NY	INTEGER	SIMPLE VAR	0-213	NZ	INTEGER	SIMPLE VAR	0-216 .I
PERPIXEL		SUBROUTINE		PIXELS	INTEGER	ARRAY	0-223 .I
READP		SUBROUTINE		REJECT	INTEGER	ARRAY	0-214 .I
UIC0	INTEGER	ARRAY	0-212 .I	WRITEP		SUBROUTINE	

PROGRAM UNIT CLASSIFY COMPILED

Parent: ASELECT

CLOSEC

CLOSEC(Z,J,TSP,LAB,KNT,DAT,NL,NS,ND,NA,N,FILENO,UICB,IND,CONTROL,ND5)

In this subroutine, a patch of pure pixels is closed by selecting a test set from the patch (5 representative pixels) and writing the test set to disk.

Method: If the file has not yet been opened (i.e., if CONTROL is true), then it is opened. The file name is TSTPXL\_\_ where \_\_ is a number from 1 to 99, depending on how many test pixel files are currently open in concurrently running sessions. The test pixel file is job-temporary, that is it is deallocated when it is closed.

If possible, five test pixels are selected from the patch by choosing them at equally spaced intervals along the array. This test set is written to a disk record. The test set count is incremented and the count of occupied labels is decremented.

Program Variables

CONTROL	LOGICAL	If CONTROL is true, the test pixel file is opened and CONTROL is set to false.
DAT(NL,ND,NS)	INTEGER ARRAY	At J, contains the list of pixels (brightness values) that constitute the current patch.
FCHECK	SYSTEM INTRINSIC	to check for I/O errors
FILEINX	INTEGER	Part of test pixel file name
FILENO	INTEGER	Test pixel file system file number
FILESIZE	INTEGER*4	Maximum number of records in test pixel file
FNAME	CHARACTER	Test pixel file name
FOPEN	SYSTEM INTRINSIC	Opens a file
FWRITE	SYSTEM INTRINSIC	Writes a record
I,IS,K,KN	INTEGER	Do Loop Index
IA	"	File options bit mask
IB	"	File access bit mask

IERR	INTEGER	I/O Error code
IMSG	"	ARRAY holds message to be printed
IND	"	ARRAY IDIMS error indicators
ITSP	"	Temporary test pixel storage
J	"	The pointer into KNT and DAT for the current patch to be processed.
KNT	INTEGER ARRAY	At J, the population of the current patch.
LA	LOGICAL	File options bit mask
LAB	INTEGER ARRAY	Label vector
LB	LOGICAL	File access bit mask
LTSP	LOGICAL	Test pixel temporary storage
MSG	CHARACTER	Message to be printed
N	INTEGER	Count of test sets written to disk
NA	"	Count of occupied labels
ND	"	Number of dimensions
ND5	" " " "	times five
NL	" " "	lines
NS	" "	samples in strip
PABORT	SYSTEM SUBROUTINE	
PIXELND	"	Counter for pixels written to a record (1 to 5*ND)
PRINTP	SYSTEM SUBROUTINE	
TSP	LOGICAL ARRAY	Test pixel record
UICB	INTEGER ARRAY	User information control block
Z	"	Absolute zero, see "Tricking Fortran"

PAGE 0026 HEWLETT-PACKARD 321020.01.03 FORTRAN/3000 TUE, OCT 13, 1981, 9:38 AM

```

00025 00776000 0CONTROL SEGHENT=AR0BASEG
00025 00777000  SUBROUTINE CLOSEC(Z,J,TSP,LAB,KNT,DAT,ML,NS,ND,NA,N,FILENO,
00025 00778000 1 UICB,IND,CONTROL,MS)
00025 00779000 C THIS SUBROUTINE IS A PORTION OF THE ANDEDA IDIRS FUNCTION.
00025 00780000 C IT CHOOSES TEST PIXELS FROM A PATCH THAT IS BEING CLOSED,
00025 00781000 C AND RESETS APPROPRIATE COUNTERS.
00025 00782000 C
00025 00783000 LOGICAL LTSP,TSP(MDS),LA,LB,CONTROL
00025 00784000 INTEGER*4 FILESIZE
00025 00785000 INTEGER*2 Z,ITSP,DAT(ML,ND,NS),KNT(1),LAB(1),INSG(39),FILEINX
00025 00786000 EQUIVALENCE (LTSP,ITSP)
00025 00787000 INTEGER*2 PIXELNO,IA,IB,UICB(1),IND(1),FILENO
00025 00788000 SYSTEM INTRINSIC FOPEN,FCHECK,FWRITE
00025 00789000 CHARACTER*74 MSG
00025 00790000 CHARACTER*9 FNAME
00025 00791000 EQUIVALENCE (LA,IA),(LB,IB),(INSG,MSG)
00025 00792000 IF (.NOT.CONTROL) GO TO 3
00030 00793000 IA = 0
00032 00794000 FILESIZE = 65536
00034 00795000 IB = 4
00036 00796000 CONTROL = .FALSE.
00040 00797000 C OPEN THE DISK FILE
00040 00798000 FNAME = 'TSTPXL.'
00046 00799000 DO 6 FILEINX = 1,99
00063 00800000 FNAME(7:2) = STR(FILEINX)
00100 00801000 FILENO = 0
00102 00802000 FILENO = FOPEN(FNAME,LA,LB,MS, ..., 2, FILESIZE, 16)
00120 00803000 IF (.CC.) 2, 3, 2
00123 00804000 2 CONTINUE
00123 00805000 CALL FCHECK(FILENO, IERR)
00130 00806000 IF (IERR.EQ.100.OR.IERR.EQ.101) GO TO 6
00140 00807000 MSG = ' OPEN ERROR ON TEST PIXEL DISK FILE'
00172 00808000 MSG(13:5) = STR(IERR)
00212 00809000 CALL PRINTP(UICB,IND,1,INSG,41,0,0,0,0,0,0,0)
00242 00810000 CALL PABORT(UICB,45,0)
00272 00811000 6 CONTINUE
00272 00812000 3 CONTINUE
00272 00813000 KN = KNT(J)
00276 00814000 IF(KN.LT.0) GO TO 100
00282 00815000 IS = (KN-1)/4
00286 00816000 PIXELNO = 0
00270 00817000 DO 10 I = 1,KN,IS
00275 00818000 DO 10 K = 1,ND
00302 00819000 PIXELNO = PIXELNO + 1
00303 00820000 ITSP = DAT(I,K,J)
00319 00821000 10 TSP(PIXELNO) = LTSP
00322 00822000 C WRITE TEST PIXELS TO DISK FILE
00322 00823000 CALL FWRITE(FILENO,TSP,MS,CONTROL)
00330 00824000 IF (.CC.) 4, 5, 4
00332 00825000 4 CALL FCHECK(FILENO, IERR)
00337 00826000 MSG = ' ERROR DURING WRITE OF TEST PIXEL FILE'
00376 00827000 MSG(8:6) = STR(IERR)
00413 00828000 CALL PRINTP(UICB,IND,1,INSG,45,0,0,0,0,0,0,0)
00442 00829000 CALL PABORT(UICB,45,0)
00492 00830000 5 CONTINUE
00492 00831000 C
00492 00832000 C INCREMENT COUNT OF TEST PIXEL GROUPS WRITTEN TO DISK

```

PAGE 0027 CLOSEC

```

00451 00833000      N = N + 1
00454 00834000      100 KNT(J) = 0
00457 00835000      LAB(J) = 2
00462 00836000      C DECREMENT COUNT OF OCCUPIED LABELS
00463 00837000      NA = NA-1
00463 00838000      RETURN
00464 00839000      END
    
```

SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
CLOSEC		SUBROUTINE		CONTROL	LOGICAL	SIMPLE VAR	0-15 .1
DAT	INTEGER	ARRAY	0-216 .1	FCHKC		SUBROUTINE	
FILEINX	INTEGER	SIMPLE VAR	0-216	FILENO	INTEGER	SIMPLE VAR	0-210 .1
FILESIZE	INTEGER*4	SIMPLE VAR	0-221	FNAME	CHARACTER	SIMPLE VAR	0-223 .1
FOPEN	INTEGER	FUNCTION		FMRITE		SUBROUTINE	
I	INTEGER	SIMPLE VAR	0-27	IA	INTEGER	SIMPLE VAR	0-210
ID	INTEGER	SIMPLE VAR	0-211	IERR	INTEGER	SIMPLE VAR	0-214
IRSG	INTEGER	ARRAY	0-25 .1	IND	INTEGER	ARRAY	0-26 .1
IS	INTEGER	SIMPLE VAR	0-212	ITSP	INTEGER	SIMPLE VAR	0-217
J	INTEGER	SIMPLE VAR	0-222 .1	K	INTEGER	SIMPLE VAR	0-215
KN	INTEGER	SIMPLE VAR	0-220	KNT	INTEGER	ARRAY	0-217 .1
LA	LOGICAL	SIMPLE VAR	0-210	LAB	INTEGER	ARRAY	0-220 .1
LB	LOGICAL	SIMPLE VAR	0-211	LTSP	LOGICAL	SIMPLE VAR	0-217
REG	CHARACTER	SIMPLE VAR	0-26 .1	N	INTEGER	SIMPLE VAR	0-211 .1
NA	INTEGER	SIMPLE VAR	0-212 .1	ND	INTEGER	SIMPLE VAR	0-213 .1
NDS	INTEGER	SIMPLE VAR	0-24 .1	NL	INTEGER	SIMPLE VAR	0-215 .1
NS	INTEGER	SIMPLE VAR	0-214 .1	PAORT		SUBROUTINE	
PIXELNO	INTEGER	SIMPLE VAR	0-213	PRINTP		SUBROUTINE	
TSP	LOGICAL	ARRAY	0-221 .1	UIFB	INTEGER	ARRAY	0-27 .1
Z	INTEGER	SIMPLE VAR	0-223 .1				

PROGRAM UNIT CLOSEC COMPILED

Parent: NUMCLU

COLAPS

COLAPS(MAX,MEAN,SUM,ND)

In this subroutine, each vector in MEAN with corresponding index in SUM zero is eliminated. The calling program NUMCLU uses SUM to mark vectors in MEAN which are no longer in force. Classification is more efficient when needless branches on  $SUM(.) = 0$  are avoided.

Method: The method is as simple-minded and as inefficient as a bubble sort. Any time  $SUM(.) = 0$  is encountered, move the entire array down one slot. (But) It is self-documenting.

#### Program Variables

I,J,K,MM,IPI	INTEGER	DO loop parameters
MAX	INTEGER	Number of vectors in MEAN.
MEAN(ND,MAX)	INTEGER ARRAY	The mean vectors, to be collapsed.
ND	INTEGER	Dimensionality
SUM(MAX)	INTEGER ARRAY	Pointer array; vectors in MEAN with $SUM(.) = 0$ should be eliminated.

PAGE 0060 HEWLETT-PACKARD 32102B.01.03 FORTRAN/3000 TUE, OCT 13, 1981, 9:41 AM

```

00006 02092000 SCONTROL SEGMENT=ANOEBASEC
00006 02093000 SUBROUTINE COLAPS(MAX,MEAN,SUM,ND)
00006 02094000 C
00006 02095000 C PARENT PROGRAM: MUMCLU
00006 02096000 C
00006 02097000 C SUBROUTINE COLAPS GOES THRU THE CLUSTERS IN MEAN
00006 02098000 C AND DELETES ANY CLUSTER WITH SUM(K) = 0, COMPRESSING
00006 02099000 C THE ARRAY MEAN; THIS ALLOWS SLIGHTLY MORE EFFICIENT
00006 02100000 C SEARCH WHEN TRYING TO CLASSIFY A POINT.
00006 02101000 INTEGER*2 MEAN(ND,MAX),SUM(MAX)
00006 02102000 NN = MAX-1
00011 02103000 DO 1 I = 1,NN
00016 02104000 IF (SUM(I).EQ.0) GO TO 1
00023 02105000 IPI = I+1
00026 02106000 IF (I'.EQ.MAX) R E T U R N
00032 02107000 DO 2 J = IPI,MAX
00037 02108000 IF (SUM(J).EQ.0) GO TO 2
00044 02109000 SUM(I) = SUM(J)
00050 02110000 SUM(J) = 0
00053 02111000 DO 3 K = 1,ND
00060 02112000 3 MEAN(K,I) = MEAN(K,J)
00075 02113000 GO TO 1
00077 02114000 2 CONTINUE
00100 02115000 R E T U R N
00101 02116000 1 CONTINUE
00102 02117000 R E T U R N
00103 02118000 END

```

## SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
COLAPS		SUBROUTINE		I	INTEGER	SIMPLE VAR	0+23
IPI	INTEGER	SIMPLE VAR	0+27	J	INTEGER	SIMPLE VAR	0+24
K	INTEGER	SIMPLE VAR	0+26	MAX	INTEGER	SIMPLE VAR	0-27 ,I
MEAN	INTEGER	ARRAY	0-26 ,I	NN	INTEGER	SIMPLE VAR	0+25
ND	INTEGER	SIMPLE VAR	0-24 ,I	SUM	INTEGER	ARRAY	0-25 ,I

PROGRAM UNIT COLAPS COMPILED

PRECEDING PAGE BLANK NOT FILMED

Parent: START

CONNCT

CONNCT(FINISHED,Z,N,A1,A2,BUF,IN,LAB)

This subroutine grows components from intervals.

Method: Label line A1 contains either Z or patch labels; Z marks boundary. Line A2 contains Z (boundary) or interval marks. A2 is scanned; when an interval is found, A1 is examined looking for a patch label. If a patch label is found, it is saved in BUF.

Then A2 is scanned again. An interval mark is replaced by the corresponding label in BUF. If none is found, the label counter LAB is incremented and LAB is stored as the new label. Labels begin at -32767 and are allowed to grow to as much as 32767. On reaching 32767, flag FINISHED is set to .TRUE. so the calling program will know the supply of patches has exhausted the labels.

"U" shaped components will not be found by this method. Rather, they will be pieced together as two different fields of labels. We do back up one line in loop 50, which sometimes removes single element patches.

Program Variables

A1(N),A2(N)	INTEGER ARRAY A1 is the elder line with patches already marked. A2 is the new line of intervals to be turned into patch labels. (The first patches are created in the calling program.)
BUF(IN)	INTEGER ARRAY Used to stash labels to be transferred to intervals.
FINISHED	LOGICAL When we run out of labels, FINISHED is set to .TRUE. The calling program uses this flag to terminate processing (gathering patches).
I,K	INTEGER DO loop index.

IN	INTEGER	Number of intervals in A2.
IS,IT	INTEGER	Temporary labels.
LAB	INTEGER	Current label pointer.
N	INTEGER	Number of elements in a line.
Z	INTEGER	Boundary marker, -32768.

PAGE 0037 HEULETT-PACKARD 321020.01.02 FORTRAN/3000 TUE, OCT 13, 1981, 9:39 AM

```

00007 01169000 $CONTROL SEGMENT=AMOBASEC
00007 01170000 SUBROUTINE CONNCT(FINISHED,Z,N,A1,A2,BUF,IN,LAB)
00007 01171000 LOGICAL FINISHED
00007 01172000 INTEGER*2 Z,A1(N),A2(N),BUF(IN)
00007 01173000 DO 10 K = 1,IN
00014 01174000 10 BUF(K) = Z
00020 01175000 DO 20 I = 1,N
00025 01176000 IT = A2(I)
00030 01177000 IF (IT.EQ.Z) GO TO 20
00034 01178000 IS = A1(I)
00037 01179000 IF (IS.EQ.Z) GO TO 20
00043 01180000 BUF(IT) = IS
00046 01181000 20 CONTINUE
00047 01182000 DO 30 K = 1,IN
00054 01183000 IF (BUF(K).NE.Z) GO TO 30
00061 01184000 LAB = LAB+1
00062 01185000 IF (LAB.EQ.32767) GO TO 60
00070 01186000 BUF(K) = LAB
00073 01187000 30 CONTINUE
00074 01188000 C
00074 01189000 C NOW TRANSFER ACTUAL LABELS
00074 01190000 DO 40 I = 1,N
00101 01191000 IT = A2(I)
00104 01192000 IF (IT.NE.Z) A2(I) = BUF(IT)
00113 01193000 40 CONTINUE
00114 01194000 C
00114 01195000 C BACK UP AND CLEAN UP SINGLETONS (MAYBE)
00114 01196000 DO 50 I = 1,N
00121 01197000 IT = A2(I)
00124 01198000 IF (IT.EQ.Z) GO TO 50
00130 01199000 IF (A1(I).NE.Z) A1(I) = IT
00137 01200000 50 CONTINUE
00140 01201000 R E T U R N
00141 01202000 60 FINISHED = .TRUE.
00143 01203000 R E T U R N
00144 01204000 END

```

## SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
A1	INTEGER	ARRAY	0-210 ,I	A2	INTEGER	ARRAY	0-27 ,I
BUF	INTEGER	ARRAY	0-26 ,I	CONNCT		SUBROUTINE	
FINISHED	LOGICAL	SIMPLE VAR	0-213 ,I	I	INTEGER	SIMPLE VAR	0+23
IN	INTEGER	SIMPLE VAR	0-25 ,I	IS	INTEGER	SIMPLE VAR	0+24
IT	INTEGER	SIMPLE VAR	0+25	K	INTEGER	SIMPLE VAR	0+26
LAB	INTEGER	SIMPLE VAR	0-24 ,I	N	INTEGER	SIMPLE VAR	0-211 ,I
Z	INTEGER	SIMPLE VAR	0-212 ,I				

PROGRAM UNIT CONNCT COMPILED

Parent: NUMCLU

DIAMTR

DIAMTR(MEAN,ND,ITL,IDIAM)

This routine determines the square of the diameter of the vectors in array MEAN.

Method: Except for the bias of -32768, the method is self-documenting. The biased squared distance from MEAN(.,I) to MEAN(.,J), with  $1 \leq I < J \leq ITL$ , is computed, and the maximum such value is returned as IDIAM.

Program Variables

I,J,K,IP,ITLM	INTEGER	DO loop parameters.
IDIAM	INTEGER	Square of diameter of vectors in MEAN.
ITL	INTEGER	The total number of vectors in MEAN.
MEAN(ND,ITL)	INTEGER ARRAY	The array whose diameter is to be determined.

PRECEDING PAGE BLANK NOT FILMED

PAGE 0059 HEWLETT-PACKARD 321020 01.03 FORTRAN/3000 TUE, OCT 13, 1981, 9:41 AM

```

00005 02067000 $CONTROL SEGMENT=ANOEBASEC
00005 02068000     SUBROUTINE DIANTR(NEAN,ND,ITL,IDIAN)
00005 02069000 C
00005 02070000 C PARENT PROGRAM:     MUMCLU
00005 02071000 C
00005 02072000 C SUBROUTINE DIANTR FINDS THE SQUARE OF THE DIAMETER
00005 02073000 C OF THE SET OF CLUSTER CENTERS IN NEAN
00005 02074000     INTEGER*2 NEAN(ND,ITL)
00005 02075000     IFLN = ITL-1
00010 02076000     IDIAN = -32768
00017 02077000     DO 1 I = 1,ITLN
00024 02078000     IP = I+1
00027 02079000     DO 1 J = IP,ITL
00034 02080000 C
00034 02081000 C FIND THE DISTANCE **2 BETWEEN NEAN(.,I) AND NEAN(.,J)
00034 02082000     IS = -32767
00036 02083000     DO 2 K = 1,ND
00043 02084000     IF (IS.GE.16300) GO TO 1
00054 02085000     IT = NEAN(K,I)-NEAN(K,J)
00071 02086000     2 IS = IS+IT*IT
00076 02087000     IF (IS.LE.IDIAN) GO TO 1
00102 02088000     IDIAN = IS
00104 02089000     1 CONTINUE
00106 02090000     R E T U R N
00107 02091000     E N D

```

SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
DIANTR		SUBROUTINE		I	INTEGER	SIMPLE VAR	0+23
IDIAN	INTEGER	SIMPLE VAR	0-24 ,I	IP	INTEGER	SIMPLE VAR	0+24
IS	INTEGER	SIMPLE VAR	0+26	IT	INTEGER	SIMPLE VAR	0+27
ITL	INTEGER	SIMPLE VAR	0-25 ,I	ITLN	INTEGER	SIMPLE VAR	0+210
J	INTEGER	SIMPLE VAR	0+25	K	INTEGER	SIMPLE VAR	0+211
NEAN	INTEGER	ARRAY	0-27 ,I	ND	INTEGER	SIMPLE VAR	0-26 ,I

PROGRAM UNIT DIANTR COMPILED

Parents: START, MARKUPDN, MARKLR

FILLLR

FILLLR(Z,LAB,N)

This subroutine works along a line of labels and replaces a label with its two neighbors marked boundary with the label boundary. It, so to speak, fills in left-right cracks in the boundary map.

Method: Only one tricky point is involved. LL, LR, and LM are used to save labels down a line. This minimizes indexing while preventing propagation of boundary down lines. As usual, Z = -32768 is used to mark boundary.

Program Variables

I	INTEGER	DO loop index
IM	INTEGER	Pointer to current position.
LAB(N)	INTEGER ARRAY	The labels being processed.
LL,LM,LR	INTEGER	Labels down the line.
N	INTEGER	Number of labels.
Z	INTEGER	Boundary: -32768

PRECEDING PAGE BLANK NOT FILMED

ORIGINAL PAGE IS  
OF POOR QUALITY

PAGE 0040 HENLETT-PACFAPD 32102B.01.03 FORTRAN/3000 TUE, OCT 13, 1981, 9:39 AM

```

00004 01245000 $CONTROL SEGMENT=ANDBASEG
00004 01246000     SUBROUTINE FILLR(Z,LAB,M)
00004 01247000     INTEGER*2 Z,LAB(M)
00004 01248000     IF (M.LT.3) R E T U R N
00010 01249000     LL = LAB(1)
00013 01250000     LM = LAB(2)
00016 01251000     IM = 2
00020 01252000     DO 10 I = 3,M
00025 01253000     LR = LAB(I)
00030 01254000     IF (LR.EQ.Z.AND.LL.EQ.Z) LAB(IM) = Z
00041 01255000     LL = LM
00043 01256000     LM = LR
00045 01257000 10 IM = I
00050 01258000     R E T U R N
00051 01259000     END

```

## SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
FILLR		SUBROUTINE		I	INTEGER	SIMPLE VAR	0+22
IF	INTEGER	SIMPLE VAR	0+23	LAB	INTEGER	ARRAY	0-25 ,I
LL	INTEGER	SIMPLE VAR	0+24	LM	INTEGER	SIMPLE VAR	0+23
LR	INTEGER	SIMPLE VAR	0+26	M	INTEGER	SIMPLE VAR	0-24 ,I
Z	INTEGER	SIMPLE VAR	0-26 ,I				

PROGRAM UNIT FILLR COMPILED

PRECEDING PAGE DATA NOT PRINTED

Parent: CLASSIFY

FIXUP

FIXUP(LABELS,NC,I1,I2,I3,PIXELS,REJECT,CLUSTERS,ND,MAXCLUS)

This subroutine follows MARKUP and attempts reclassification of a pixel classified unlike each of its four neighbors. The attempt fails when each neighbor is too far from the pixel to stand that reclassification.

Method: As in MARKUP, a circular buffer LABELS is maintained. I1 points to the eldest, I2 to the current (which may have been modified by the addition of 101), and I3 to the nearest (unchecked) line of labels. We regard the classifications in line I1 as final, those in I3 as tentative. Those in I2 marked over 100 are fixed by subtraction of the 101 added by FIXUP. Of the rest, note they are classed unlike any neighbor. Collect, from the four neighbors, the classes of each neighbor (a) in line I1, (b) in line I2 and marked, or (c) in line I3 and like at least one neighbor in that line. This gives up to four classes. Reclassify the pixel in the nearest unrejected class of those up to four classes. If no unrejected class exists, leave the classification alone. All distances are biased by -32768.

Program Variables

CLUSTERS(ND,MAXCLUS)	INTEGER ARRAY	The cluster centers.
FOUR(4)	INTEGER ARRAY	Used to store the (up to) four classifications of OK neighbors.
I	INTEGER	DO loop index.
I1	INTEGER	Pointer to eldest label line.
I2	INTEGER	Pointer to current label line.
I3	INTEGER	Pointer to newest label line.
IM	INTEGER	I-1; pointer to current slot.
IS	INTEGER	Sum accumulator for distance.

ITT	INTEGER	Scratch variable.
J	INTEGER	Index into FOUR: DO loop index.
J4	INTEGER	Value of FOUR(J) in loop.
K	INTEGER	DO loop index.
L	INTEGER	DO loop index.
LL	INTEGER	Label on left.
LM	INTEGER	Label in middle.
LR	INTEGER	Label on right.
MAXCLUS	INTEGER	Dimension for CLUSTERS.
NC	INTEGER	Number of samples per line.
ND	INTEGER	Dimensionality
NDIST	INTEGER	Distance from nearest neighbor classifier, pixel to nearest of up to four.
NRST	INTEGER	Index of nearest, or zero if all are too far away.
PIXELS(NC,ND)	INTEGER ARRAY	One line of data along line I2.
REJECT(MAXCLUS)	INTEGER ARRAY	Rejection thresholds.

PAGE 0022 HEWLETT-PACKARD 321020 01 03 FORTRAN/3000 TUE, OCT 13, 1981, 9:37 AM

```

00014 00669000 *CONTROL SEGMENT=ANDBASEG
00014 00670000 SUBROUTINE FIXUP(LABELS,NC,11,12,13,PIXELS,REJECT,
00014 00671000 * CLUSTERS,ND,MAXCLUS,MAX
00014 00672000 * INTEGER*2 LABELS(NC,3),PIXELS(NC,ND),REJECT(MAXCLUS),
00014 00673000 * CLUSTERS(ND,MAXCLUS),FOUR(4)
00014 00674000 IF (MAX.LT.3) R E T U R N
00020 00675000 LL = LABELS(1,12)
00026 00676000 LR = LABELS(2,12)
00033 00677000 DO 10 I = 3,MAX
00042 00678000 LR = LABELS(I,12)
00051 00679000 IN = I-1
00054 00680000 IF (LR.GT.100) GO TO 101
00060 00681000 J = 0
00062 00682000 ITT = LABELS(IN,11)
00071 00683000 IF (ITT.EQ.0) GO TO 30
00073 00684000 J = 1
00077 00685000 FOUR(J) = ITT
00102 00686000 30 IF (LL.LE.100) GO TO 40
00106 00687000 J = J+1
00107 00688000 FOUR(J) = LL-101
00113 00689000 40 IF (LR.LE.100) GO TO 50
00117 00690000 J = J+1
00120 00691000 FOUR(J) = LR-101
00124 00692000 50 ITT = LABELS(IN,13)
00133 00693000 IF (ITT.NE.LABELS(I-2,13) AND ITT.NE.LABELS(I,13))GO TO 60
00135 00694000 J = J+1
00156 00695000 FOUR(J) = ITT
00161 00696000 60 IF (J.EQ.0) GO TO 11
00163 00697000 NRST = 0
00167 00698000 NDIST = 32767
00171 00699000 DO 70 L = 1,J
00176 00700000 J4 = FOUR(L)
00201 00701000 IS = -32768
00210 00702000 DO 80 K = 1,ND
00213 00703000 IF (IS.GE.16383) GO TO 70
00226 00704000 ITT = PIXELS(IN,K)-CLUSTERS(K,J4)
00242 00705000 IS = IS+ITT*ITT
00247 00706000 IF (IS.GE.REJECT(J4)) GO TO 70
00253 00707000 80 CONTINUE
00256 00708000 IF (NDIST.LE.IS) GO TO 70
00262 00709000 NDIST = IS
00264 00710000 NRST = J4
00266 00711000 70 CONTINUE
00267 00712000 IF (NRST.NE.0) LABELS(IN,12) = NRST
00301 00713000 GO TO 11
00302 00714000 101 LABELS(IN,12) = LR-101
00312 00715000 11 LL = LR
00314 00716000 10 LR = LR
00317 00717000 R E T U R N
00320 00718000 END

```

PAGE 0023 FIXUP

## SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
CLUSTEPS	INTEGER	ARRAY	0-27 .I	FIXUP		SUBROUTINE	
FOUR	INTEGER	ARRAY	0-24 .I	I	INTEGER	SIMPLE VAR	0-25
I1	INTEGER	SIMPLE VAR	0-214 .I	I2	INTEGER	SIMPLE VAR	0-213 .I
I3	INTEGER	SIMPLE VAR	0-212 .I	IN	INTEGER	SIMPLE VAR	0-27
IS	INTEGER	SIMPLE VAR	0-212	ITY	INTEGER	SIMPLE VAR	0-211
J	INTEGER	SIMPLE VAR	0-210	J4	INTEGER	SIMPLE VAR	0-213
K	INTEGER	SIMPLE VAR	0-216	L	INTEGER	SIMPLE VAR	0-26
LABELS	INTEGER	ARRAY	0-216 .I	LL	INTEGER	SIMPLE VAR	0-214
LM	INTEGER	SIMPLE VAR	0-215	LR	INTEGER	SIMPLE VAR	0-217
MAXCLUS	INTEGER	SIMPLE VAR	0-25 .I	NC	INTEGER	SIMPLE VAR	0-215 .I
NO	INTEGER	SIMPLE VAR	0-26 .I	NDIST	INTEGER	SIMPLE VAR	0-221
NRST	INTEGER	SIMPLE VAR	0-220	NX	INTEGER	SIMPLE VAR	0-24 .I
PIXELS	INTEGER	ARRAY	0-211 .I	REJECT	INTEGER	ARRAY	0-210 .I

PROGRAM UNIT FIXUP COMPILED

Parent: THINTSTM

GETN25

GETN25(FRSTFLG,TTP,N25,NDS,FILENO,UICB,IND)

This program fetches N25 test sets from a disk file written by the subroutine CLOSEC.

Method: On the first call to the subroutine the disk file is rewound. On each call, N25 test sets are read from the file. A test set is five test pixels. The file is job-temporary.

#### Program Variables

FCHECK	SYSTEM INTRINSIC	Error checking
FCONTROL	SYSTEM	" Used to rewind file
FILENO	INTEGER	System file number
FREAD	SYSTEM INTRINSIC	File read
FRSTFLG	LOGICAL	Marks first call
I	INTEGER	DO loop index
ICALL	"	Number of words returned by read
IERR	"	Stuffed with error code
IMSG	"	ARRAY Message to be printed
IND	INTEGER ARRAY	Error code location
MSG	CHARACTER	Message to be printed
N25	INTEGER	Number of test sets requested
NDS	"	5 times number of dimensions
PABORT	SYSTEM SUBROUTINE	
PRINTP	"	" Prints a message
TTP	LOGICAL ARRAY	Stuffed with test sets
UICB	INTEGER ARRAY	User information control block
ZERO	LOGICAL	Dummy argument for FCONTROL

PAGE 0046 HEWLETT-PACKARD 32102B.01.03 FORTRAN/3000 TUE, OCT 13, 1981, 9:40 AM

```

00015 01497000 8CONTROL SEGMT=ARGBASEC
00015 01498000     SUBROUTINE GETN25(FRSTFLG, TYP, N25, NDS, FILENO, UICB, IND)
00015 01499000 C THIS SUBROUTINE FETCHES N25 SETS OF 3 TEST PIXELS FROM A DISK FILE TNA
00015 01500000 C WAS CREATED EARLIER
00015 01501000 C
00015 01502000     INTEGER*2 UICB(1), IND(1), FILENO,
00015 01503000     ' INSG(39)
00015 01504000     SYSTEM INTRINSIC FCONTROL, FREAD, FCHECK
00015 01505000     LOGICAL TYP(NDS, N25), ZERO, FRSTFLG
00015 01506000     CHARACTER*78 MSG
00015 01507000     EQUIVALENCE (INSG, MSG)
00015 01508000 C
00015 01509000     IF (.NOT. FRSTFLG) GO TO 10
00020 01510000     ZERO = .FALSE.
00022 01511000     FRSTFLG = .FALSE.
00024 01512000 C ON FIRST CALL, REWIND FILE
00024 01513000     CALL FCONTROL (FILENO, 3, ZERO)
00026 01514000     IF (.CC.) 0, 10, 6
00022 01515000 6 MSG = ' FAILURE TO REWIND TEST PIXEL FILE'
00063 01516000     CALL PRINTP(UICB, IND, 1, INSG, 34, 0, 0, 0, 0, 0, 0, 0, 0)
00112 01517000     CALL PABORT(UICB, 45, 0)
00122 01518000 10 CONTINUE
00122 01519000 C READ N25 SETS OF 3 TEST PIXELS
00123 01520000     DO 30 I = 1, N25
00130 01521000         ICALL = FREAD(FILENO, TYP(I, 1), NDS)
00142 01522000         IF (.CC.) 20, 30, 20
00144 01523000 20 CALL FCHECK(FILENO, IERR)
00151 01524000         MSG = ' ERROR 0      ON TEST PIXEL FILE READ '
00205 01525000         MSG(10:6) = STR(IERR)
00222 01526000         CALL PRINTP(UICB, IND, 1, INSG, 40, 0, 0, 0, 0, 0, 0, 0, 0)
00232 01527000         CALL PABORT(UICB, 45, 0)
00262 01528000 30 CONTINUE
00263 01529000     RETURN
00264 01530000     END

```

## SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
FCHECK	INTEGER	SUBROUTINE		FCONTROL		SUBROUTINE	
FILENO	INTEGER	SIMPLE VAR	0-26 .1	FREAD	INTEGER	FUNCTION	
FRSTFLG	LOGICAL	SIMPLE VAR	0-212 .1	GETN25		SUBROUTINE	
I	INTEGER	SIMPLE VAR	0+23	ICALL	INTEGER	SIMPLE VAR	0+27
IERR	INTEGER	SIMPLE VAR	0+26	INSG	INTEGER	ARRAY	0+23 .1
IND	INTEGER	ARRAY	0-24 .1	MSG	CHARACTER	SIMPLE VAR	0+24 .1
N25	INTEGER	SIMPLE VAR	0-210 .1	NDS	INTEGER	SIMPLE VAR	0-27 .1
PABORT		SUBROUTINE		PRINTP		SUBROUTINE	
TYP	LOGICAL	ARRAY	0-211 .1	UICB	INTEGER	ARRAY	0-25 .1
ZERO	LOGICAL	SIMPLE VAR	0+210				

PROGRAM UNIT GETN25 COMPILED

PRECEDING PAGE BLANK NOT FILMED

PARENT: MAPP

IIIFN

FUNCTION IIIFN(I)

Maps -32768 to 32767 into 1 to 59 so that: -32768, 0, and 99 all map into 0; other integers I map to numbers between 2 and 59. This function is used by MAPP to index into SYMBOL. Special numbers -32768, 0, and 99 are all mapped to print as blanks. Others print as symbols so that when I and J are close, the symbols are different.

Method: Self-documenting.

Program Variables

I                    INTEGER   Function argument

UNRECORDED PAGE BLANK NOT FILMED

ORIGINAL PAGE IS  
OF POOR QUALITY

PAGE 0015 HEWLETT-PACKARD 32102B.01.03 FORTRAN/3000 TUE, OCT 13, 1981, 9:37 AM

```

00002 00479000 #CONTROL SEGMENT=ANOEBASEC
00002 00480000 FUNCTION IIIFN(I)
00002 00481000 IF (I GT 0) GO TO 3
00006 00482000 IF (I LE 0 OR I.EQ.-32768) GO TO 4
00022 00493000 I = -I
00025 00494000 GO TO 5
00026 00495000 3 IF (I EQ 99) GO TO 4
00032 00486000 5 I = MOD(I,60)
00036 00487000 IF (I LE 1) I = I+30
00044 00488000 IIIFN = I
00046 00499000 R E T U R N
00047 00490000 4 IIIFN = I
00051 00491000 R E T U R N
00052 00492000 END

```

SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
I	INTEGER	SIMPLE VAR	0-24 ,I	IIIFN	INTEGER	SIMPLE VAR	0-25
IIIFN	INTEGER	FUNCTION					

PROGRAM UNIT IIIFN COMPILED

PRECEDING PAGE BLANK NOT FILMED

Parent: MAIN

MAPP

Calls: IIIFN

MAPP(N1,N2,N3,UICB,IND,IMAGE,NR,NC,SYMBOL)

Produces a quick look at a segment of data, labels map, or cluster map. Output is directed to the default device. This program is intended for debugging.

Method: The subroutine prints N2 lines in band one of an image, starting at line N1 sample N3. Because of the limitations of PRINTP, only 64 samples can be printed. Line numbers are printed, but not column numbers.

Program Variables

CHKIO	SYSTEM SUBROUTINE
IC	INTEGER DO loop index
IIIFN(IV)	INTEGER FUNCTION
IMAGE	INTEGER Image number.
IND(1)	INTEGER ARRAY Error information.
IPX(36)	INTEGER ARRAY Dummy array to compensate for the inadequacies of PRINTP. Equivalenced to PICTURE.
IR	INTEGER DO loop index.
N1	INTEGER Starting line number.
N1PN2	INTEGER Last line number to print.
N2	INTEGER Number of lines
N3	INTEGER Starting sample number
NC	INTEGER Number of samples in image.
NCL	INTEGER Last sample to be printed.
NCP	INTEGER Actual number printed.

PRECEDING PAGE BEING NOT FILMED

NCP8	INTEGER NCP+8, the number sent to PRINTP.
NR	INTEGER Number of lines in image.
PICTURE	CHARACTER*72 area for core-to-core write under FORMAT control.
PRINTP	SYSTEM SUBROUTINE
READP	SYSTEM SUBROUTINE
SCAN(64)	INTEGER ARRAY Array to read from image.
SYMBOL(59)	CHARACTER*1 AKRAY The symbols printed; created by SETSYM.
UICB	INTEGER ARRAY

ORIGINAL PAGE IS  
OF POOR QUALITY

PAGE 0013 HEWLETT-PACKARD 32102B 01.03 FORTRAN/3000 TUE, OCT 13, 1981, 9:37 AM

```

00026 00443000 80CONTROL SEGMENT=ANDBASEG
00026 00444000  SUBROUTINE MAP(N1,N2,N3,UICB,IND,IMAGE,NR,NC,SYMBOL)
00026 00445000  INTEGER*2 IPX(20),MIPN(10),IND(10),SCAN(64)
00026 00446000  CHARACTER*72 PICTURE
00026 00447000  CHARACTER*1 SYMBOL(59)
00026 00448000  EQUIVALENCE(PICTURE,IPX)
00026 00449000  MIPN(1) = N1+N2-1
00026 00450000  IF (MIPN(1) GT NR) MIPN(1) = NR
00027 00451000  C
00027 00452000  C PRINT FROM COL N3 TO COL N3+63, ROWS N1 TO MIPN2.
00027 00453000  NCL = MINO(NC,N3+63)
00027 00454000  MCF = NCL-N3+1
00027 00455000  MCFB = MCF+8
00027 00456000  DO 100 IR = N1,MIPN2
00027 00457000  CALL READP(UICB,IND,IMAGE,SCAN,2,1,IR,N3,MCF,1,IR+1,N3,MCF)
00027 00458000  WRITE(PICTURE,111)IR,(SYMBOL(I)IFN(SCAN(IC))),IC = 1,MCF)
00027 00459000  111 FORMAT(16,2X,64A1)
00027 00460000  100 CALL PRINTF(UICB,IND,1,IPX,MCFB,0,0,0,0,0,0,0)
00027 00461000  IF (IND(1) LT 0) CALL CHRIO(UICB,IND,1,IR,0,0,111)
00027 00462000  R E T U R N
00027 00463000  END

```

## SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
MAP		SUBROUTINE		IC	INTEGER	SIMPLE VAR	0+25
IND	INTEGER	FUNCTION		IMAGE	INTEGER	SIMPLE VAR	0+27 ,1
IPX	INTEGER	ARRAY	0+20 ,1	IPX	INTEGER	ARRAY	0+21 ,1
IF	INTEGER	SIMPLE VAR	0+26	MAP		SUBROUTINE	
N1	INTEGER	SIMPLE VAR	0+24 ,1	MIPN2	INTEGER	SIMPLE VAR	0+24
N2	INTEGER	SIMPLE VAR	0+213 ,1	N3	INTEGER	SIMPLE VAR	0+212 ,1
NC	INTEGER	SIMPLE VAR	0+25 ,1	NCL	INTEGER	SIMPLE VAR	0+27
MCF	INTEGER	SIMPLE VAR	0+210	MCFB	INTEGER	SIMPLE VAR	0+211
NR	INTEGER	SIMPLE VAR	0+26 ,1	PICTURE	CHARACTER	SIMPLE VAR	0+22 ,1
PRINTF		SUBROUTINE		READP		SUBROUTINE	
SCAN	INTEGER	ARRAY	0+23 ,1	SYMBOL	CHARACTER	ARRAY	0+24 ,1
UICB	INTEGER	ARRAY	0+211 ,1				

PROGRAM UNIT MAP COMPILER

Parent: START

MARKLR

Calls: FILLLR

MARKLR(Z,DATA,INTTHR,NC,ND,LABELS,MASK)

This subroutine marks boundary points left-right.

Method: A line of data is scanned. Points in the mask are marked boundary (under control of MASK). Points which, in any band, are closer to their left neighbor than INTTHR(.) are marked boundary. Finally, subroutine FILLLR is used to fill in a single point gap along the line.

Program Variables

DATA(NC,ND)	INTEGER ARRAY One line of data
FILLLR	SUBROUTINE Fills in gaps along a line.
I,K	INTEGER DO loop index
IM	INTEGER I-1
INTTHR(ND)	INTEGER ARRAY The thresholds for boundary finding.
LABELS(NC)	INTEGER ARRAY A line of labels
MASK	LOGICAL The mask flag. When .TRUE., zero in band 1 marks a point off the image, i.e., masked.
NC	INTEGER Number of samples per line.
ND	INTEGER Dimensionality
Z	INTEGER -32768, passed as a parameter, and used to mark boundary points.

PRECEDING PAGE BLANK NOT FILMED

Parent: CLASSIFY

MARKUP

MARKUP(LABELS,NC,I1,I2,I3)

This subroutine "marks" pixels classified like at least one neighbor. This follows a nearest neighbor classification by PERPIXEL, and is followed by a possible reclassification of errant pixels by FIXUP.

Method: The pixel labels reside in a circular buffer LABELS of NC labels per line. I1 points to the eldest label, I2 to the current label, and I3 to the newest. The eldest label may have been spatially modified, but labels along scan line I2 are not propagated down that scan line. The number 101 is added to any label classified like at least one of the four neighbors along scan line I2.

Program Variables

I	INTEGER	DO loop index
I1	INTEGER	Eldest line pointer.
I2	INTEGER	Current line pointer.
I3	INTEGER	Newest line pointer.
IM	INTEGER	I-1
LABELS(NC,3)	INTEGER ARRAY	The labels; LABELS(.,I2) is modified by adding 101 when at least one neighbor has the same label.
LL	INTEGER	Label on left.
LM	INTEGER	Label in middle.
RL	INTEGER	Label on right
NC	INTEGER	Number of samples per line.

PRECEDING PAGE LABELS NOT MARKED

PAGE 0021 HEWLETT-PACKARD 32102B.01.03 FORTRAN/3000 TUE, OCT 13, 1981, 9:37 AM

```

00004 00648000 $CONTROL SECHENT=AMOEBASEG
00004 00649000 SUBROUTINE MARKUP(LABELS,NC,11,12,13,NX)
00004 00650000 C THIS PROGRAM ADDS TO TO ANY CLASSIFICATION WHICH IS LIKE
00004 00651000 C AT LEAST ONE NEIGHBOR 101
00004 00652000 INTEGER*2 LABELS(NC,?)
00004 00653000 IF (NX LT 3) R E T U R N
00010 00654000 IM = 2
00012 00655000 LL = LABELS(1,12)
00020 00656000 LM = LABELS(2,12)
00027 00657000 DO 10 I = 3,NX
00034 00658000 LR = LABELS(I,12)
00043 00659000 IF (LR EQ.0) GO TO 30
00047 00660000 IF (LL EQ.LM OR LR EQ.LM) GO TO 20
00056 00661000 IF (LABELS(IM,13) EQ.LM) GO TO 20
00067 00662000 IF (LABELS(IM,11) NE.LM) GO TO 30
00100 00663000 20 LABELS(IM,12) = LM+101
00110 00664000 30 LL = LM LABELS(IM,11) - 101
00112 00665000 LM = LR
00114 00666000 10 IM = I
00117 00667000 R E T U R N
00120 00668000 END

```

## SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
I	INTEGER	SIMPLE VAR	0-22	I1	INTEGER	SIMPLE VAR	0-27 ,I
I2	INTEGER	SIMPLE VAR	0-26 ,I	I3	INTEGER	SIMPLE VAR	0-29 ,I
IM	INTEGER	SIMPLE VAR	0-23	LABELS	INTEGER	ARRAY	0-211 ,I
LL	INTEGER	SIMPLE VAR	0-24	LM	INTEGER	SIMPLE VAR	0-25
LR	INTEGER	SIMPLE VAR	0-26	MARKUP	SUBROUTINE		
NC	INTEGER	SIMPLE VAR	0-210 ,I	NX	INTEGER	SIMPLE VAR	0-24 ,I

PROGRAM UNIT MARKUP COMPILED

1. THE PROGRAM UNIT MARKUP IS A SUBROUTINE  
 WHICH TAKES A CLASSIFICATION AND ADDS  
 TO IT AT LEAST ONE NEIGHBOR.

Parent: START

MARKUPDN

Calls: FILLLR

MARKUPDN(Z,D1,D2,L1,L2,L3,NC,ND,INTTHR,MASK)

This subroutine marks boundary points up-down.

METHOD: If, in any band, data in D1 is closer to or equal to data in D2 than the threshold INTTHR, then labels L1 and L1 and L2 at that sample are marked by being set equal to Z. The reason for the "less than or equal to" rather than "less than" as appears in MARKLR is that the data is generally less variable down scan lines than across scan lines. Points off the image are also marked as boundary provided MASK is set. At conclusion of this scan through the data, FILLLR is called on center line L2.

Program Variables

D1(NC,ND) D2(NC,ND)	INTEGER ARRAY Two adjacent lines of data; D1 is the eldest, D2 newer.
FILLLR	SUBROUTINE Fills in gaps along a line.
I,K	INTEGER DO loop index.
INTTHR(ND)	INTEGER ARRAY Vector thresholds for deciding boundary.
L1(NC) L2(NC) L3(NC)	INTEGER ARRAY Three scan lines of labels. L1 is oldest, L3 newest.
MASK	LOGICAL Flag used to decide whether a value of 0 in D1(.,1) or D2(.,1) marks the mask. Mask points are marked boundary.
NC	Number of samples per line
ND	Dimensionality
7	-32768, used to mark boundary.

```

00012 01205000 *CONTROL SEGMENT=ARCDBASEC
00012 01206000 SUBROUTINE HARKUPDN(Z,D1,D2,L1,L2,L3,NC,ND,INTTHR,MASK,MX)
00012 01207000 INTEGER*2 Z,D1(NC,ND),D2(NC,ND),L1(NC),L2(NC),L3(NC),INTTHR(ND)
00012 01208000 LOGICAL MASK
00012 01209000 DO 10 I = 1,MX
00017 01210000 IF (L1(I).EQ.Z.AND.L2(I).EQ.Z) GO TO 20
00030 01211000 IF (.NOT.MASK) GO TO 1
00037 01212000 IF (D1(I,1).EQ.0) L1(I) = Z
00043 01213000 IF (D2(I,1).EQ.0) L2(I) = Z
00057 01214000 1 DO 30 K = 1,ND
00064 01215000 IF (ABS(D1(I,K)-D2(I,K)).LE.INTTHR(K)) GO TO 30
00106 01216000 L1(I) = Z
00111 01217000 GO TO 20
00113 01218000 30 CONTINUE
00114 01219000 GO TO 10
00115 01220000 20 L2(I) = Z
00120 01221000 10 CONTINUE
00121 01222000 CALL FILLR(Z,L2,MX)
00126 01223000 R E T U R N
00127 01224000 END
    
```

SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
D1	INTEGER	ARRAY	0-215 ,I	D2	INTEGER	ARRAY	0-216 ,I
FILLR		SUBROUTINE		I	INTEGER	SIMPLE VAR	0-23
INTTHR	INTEGER	ARRAY	0-26 ,I	K	INTEGER	SIMPLE VAR	0-24
L1	INTEGER	ARRAY	0-213 ,I	L2	INTEGER	ARRAY	0-212 ,I
L3	INTEGER	ARRAY	0-211 ,I	HARKUPDN		SUBROUTINE	
MASK	LOGICAL	SIMPLE VAR	0-25 ,I	NC	INTEGER	SIMPLE VAR	0-210 ,I
ND	INTEGER	SIMPLE VAR	0-27 ,I	MX	INTEGER	SIMPLE VAR	0-24 ,I
Z	INTEGER	SIMPLE VAR	0-216 ,I				

PROGRAM UNIT HARKUPDN COMPILED

ORIGINAL PAGE IS  
OF POOR QUALITY

PRECEDING PAGE BLANK NOT FILMED

Parent: MAIN

MOREQUES

Calls: REJECTH

MOREQUES(MEANS,TESTS,MAXCLUS,NFCLUS,ND,NTS,REJECT)

The purpose of MOREQUES is to detect when a necessary cluster has been lost and add it in. The test is based on the pure rejection thresholds applied to classification of the center (i.e., third of 5) test pixel in each test set.

Method: Initially, REJECTH is called to determine the pure rejection thresholds. Then a loop on center of test sets is entered. Each time no cluster center is closer than its rejection threshold to the test pixel, that test pixel is added as a new cluster center, the reject thresholds are calculated, and the next test set examined. On completion of all examinations, the reject thresholds are multiplied by 2 (effectively multiplying the Euclidean distance test by  $\sqrt{2}$ , preparing for misregistration mixtures). Since there is a bias of -32768, the actual calculation goes as follows: If  $r$  is the unbiased rejection threshold and  $R$  the biased, then  $r = R + 32768$ , so the new unbiased threshold  $2r$  has its biased threshold  $R' = 2r - 32768 = 2R + 32768$ . (If  $R'$  is greater than 16000, use  $R' = 16000$ .)

Program Variables

A0	INTEGER*4	Long integer used to perform long calculations
I,J,K	INTEGER	DO loop parameters
IS,IT	INTEGER	Used in accumulating distance.
MAXCLUS	INTEGER	Maximum number of clusters.
MEANS(ND,MAXCLUS)	INTEGER ARRAY	The cluster centers.
ND	INTEGER	Dimensionality
NFCLUS	INTEGER	Current number of clusters.
NTS	INTEGER	Number of test sets.

REJECT(MAXCLUS)    INTEGER ARRAY    Rejection thresholds.  
REJECTH            SUBROUTINE    Calculates REJECT.  
TESTS(ND,NTS)     INTEGER ARRAY    Test pixels.

PAGE 024 HONEYWELL-PACARD J21020 01.03 FORTRAN/3000 TUE, OCT 13, 1981, 9:37 AM

```

00010 00719000 8CONTROL SEGMENT=HOREBASEC
00010 00720000 SUBROUTINE HOREQUES(MEANS,TESTS,MAXCLUS,MFCLUS,MD,NTS,REJECT)
00010 00721000 INTEGER*2 MEANS(MD,MAXCLUS),TESTS(MD,NTS),REJECT(MAXCLUS)
00010 00722000 INTEGER*4 AD
00010 00723000 CALL REJECTM(MEANS,MD,REJECT,MFCLUS,MAXCLUS)
00020 00724000 DO 10 I = 1,NTS,3
00025 00725000 DO 20 J = 1,MFCLUS
00032 00726000 IS = -32768
00041 00727000 DO 20 K = 1,MD
00046 00728000 IF (IS.GE.16393) GO TO 30
00056 00729000 IT = TESTS(K,I)-MEANS(K,J)
00073 00730000 IS = IS+IT*IT
00077 00731000 IF (IS.GE.REJECT(J)) GO TO 30
00109 00732000 20 CONTINUE
00106 00733000 C
00106 00734000 C GETTING HERE MEANS WE SUCCESSFULLY CLASSIFIED TESTS(.I)
00106 00735000 GO TO 10
00110 00736000 30 CONTINUE
00111 00737000 C
00111 00738000 C GETTING HERE MEANS WE DIDN'T. ADD ONE.
00111 00739000 MFCLUS = MFCLUS+1
00112 00740000 DO 40 K = 1,MD
00117 00741000 40 MEANS(K,MFCLUS) = TESTS(K,I)
00134 00742000 CALL REJECTM(MEANS,MD,REJECT,MFCLUS,MAXCLUS)
00144 00743000 IF (MFCLUS.GE.MAXCLUS) GO TO 50
00151 00744000 10 CONTINUE
00152 00745000 50 DO 60 I = 1,MFCLUS
00157 00746000 AD = REJECT(I)
00164 00747000 AD = AD+32768
00174 00748000 IF (AD.GT.16000) AD = 16000
00206 00749000 60 REJECT(I) = AD
00217 00750000 R E T U R N
00227 00751000 END

```

## SYMBOL MAP

VAR.	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
AD	INTEGER*4	SIMPLE VAR	0-211	I	INTEGER	SIMPLE VAR	0-24
IS	INTEGER	SIMPLE VAR	0-26	IT	INTEGER	SIMPLE VAR	0-27
J	INTEGER	SIMPLE VAR	0-25	K	INTEGER	SIMPLE VAR	0-210
MAXCLUS	INTEGER*2	SIMPLE VAR	0-210 .I	MEANS	INTEGER	ARRAY	0-212 .I
HOREQUES		SUBROUTINE		MD	INTEGER	SIMPLE VAR	0-26 .I
MFCLUS	INTEGER	SIMPLE VAR	0-27 .I	NTS	INTEGER	SIMPLE VAR	0-25 .I
REJECT	INTEGER	ARRAY	0-24 .I	REJECTM		SUBROUTINE	
TESTS	INTEGER	ARRAY	0-211 .I				

PROGRAM UNIT HOREQUES COMPILED

Parent: START

MRKIVL

MRKIVL(Z,A,N,M)

This subroutine prepares the complement of the boundary for accumulation of components.

Method: Recall that boundary points are marked with Z (-32768). The rest are counted left-to-right along line A in intervals, replacing the slot in A by the interval number. For example, a 20 point line on input might be

Z Z 0 0 0 Z Z Z Z 0 Z 0 0 Z 0 Z Z Z 0 0

and the line returned would be

Z Z 1 1 1 Z Z Z Z 2 Z 3 3 Z 4 Z Z Z 5 5 .

On return, M is the number of intervals found.

Program Variables

A(N)	INTEGER ARRAY	One line of boundary labels, in which intervals are to be found.
I	INTEGER	DO loop index
IM	INTEGER	I-1
M	INTEGER	Interval counter.
N	INTEGER	Number in a line
Z	INTEGER	Boundary marker.

PRECEDING PAGE BLANK NOT FILMED

PAGE 0036 HEWLETT-PACKARD J2102B.01.03 FORTRAN/3000 TUE, OCT 13, 1981, 9:39 AM

```

00004 01150000 #CONTROL SEGMENT=ANDBASEG
00004 01151000     SUBROUTINE MRKIVL(Z,A,N,M)
00004 01152000     INTEGER*2 Z,A(N)
00004 01153000     N = 1
00006 01154000     IF (A(1).EQ.Z) GO TO 10
00013 01155000     A(1) = N
0001C 01156000     N = N+1
00017 01157000 10 DO 20 I = 2,N
00024 01158000     IN = I-1
00027 01159000     IF (A(I).EQ.Z) GO TO 20
00034 01160000     IF (A(IN).EQ.Z) GO TO 30
00041 01161000     A(I) = A(IN)
00047 01162000     GO TO 20
00046 01163000 30 A(I) = N
00051 01164000     N = N+1
00052 01165000 20 CONTINUE
00053 01166000     N = N-1
00054 01167000     R E T U R N
00055 01168000     END
    
```

SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
A	INTEGER	ARRAY	0-26 .I	I	INTEGER	SIMPLE VAR	0-22
I	INTEGER	SIMPLE VAR	0-23	M	INTEGER	SIMPLE VAR	0-24 .I
MRKIVL		SUBROUTINE		N	INTEGER	SIMPLE VAR	0-25 .I
Z	INTEGER	SIMPLE VAR	0-27 .I				

PROGRAM UNIT MRKIVL COMPILED

Parent: MAIN

MSORT

Calls: SHELL

MSORT(MEAN,ND,::FC,DUM,INDX)

Sorts final clusters by average odd channels to aid interpretation of clustering.

Method: First the sums are accumulated. At the same time, an index is set so that  $INDX(I) = I$ . Then SHELL is called. On return from SHELL, the means are reordered by the permutation of INDX. The actual means are now switched in place.

Program Variables

DUM(NFC)	INTEGER ARRAY	Used to accumulate sums in odd bands, and then as temporary storage to switch MEAN.
I	INTEGER	DO loop index
INDX(NFC)	INTEGER ARRAY	The pointer array, used by SHELL to indicate actual order of DUM.
J	INTEGER	DO loop index.
K	INTEGER	DO loop index.
MEAN(ND,NFC)	INTEGER ARRAY	The means to be sorted.
ND	INTEGER	Dimensionality of MEAN.
NFC	INTEGER	Number of vectors in MEAN.
SHELL	SUBROUTINE	Sorts vector in increasing order.

PAGE 0010 HENLETT-FACOMPC 321028.01.03 FORTRAN/3000 TUE, OCT 13, 1961, 9:36 AM

```

00007 00360000 #CONTROL SEGMENT=ANDEBASEG
00007 00361000     SUBROUTINE NSORT(NEAN,ND,NFC,DUM,INDX)
00007 00362000     INTEGER*2 NEAN(ND,NFC),DUM(NFC),INDX(NFC)
00007 00363000     DO 1 I = 1,NFC
00014 00364000     DUM(I) = 0
00017 00365000     DO 100 J = 1,ND,2
00024 00366000     100 DUM(I) = DUM(I)+NEAN(J,I) -
00036 00367000     1 INDX(I) = I
00042 00368000     CALL SHELL(DUM,INDX,NFC)
00050 00369000     DO 110 K = 1,ND
00055 00370000     DO 120 I = 1,NFC
00062 00371000     120 DUM(I) = NEAN(K,INDX(I))
00074 00372000     DO 130 I = 1,NFC
00101 00373000     130 NEAN(K,I) = DUM(I)
00112 00374000     110 CONTINUE
00113 00375000     R E T U R N
00114 00376000     E N D
    
```

SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
DUM	INTEGER	ARRAY	0-23 ,I	I	INTEGER	SIMPLE VAR	0+23
INDX	INTEGER	ARRAY	0-24 ,I	J	INTEGER	SIMPLE VAR	0+24
K	INTEGER	SIMPLE VAR	0+25	NEAN	INTEGER	ARRAY	0-210 ,I
NSORT		SUBROUTINE		ND	INTEGER	SIMPLE VAR	0-27 ,I
NFC	INTEGER	SIMPLE VAR	0-26 ,I	SHELL		SUBROUTINE	

PROGRAM UNIT NSORT COMPILED

PRECEDING PAGE BLANK NOT FILMED

Parent: MAIN

NUMCLU

Calls: COLAPS, DIAMTR, UNCLE

NUMCLU(MEAN,ND,NP5,NP,TSPXL,NFCLUS,MINCLN,MAXCLN,CLASS,COUNT,ERROR,SAVE,  
DUM,CSAVE,UICB,IND,NUM)

NUMCLU is the main clustering segment in AMOEBA. It carries out cluster formation according to the strategy suggested by the model (Appendix A).

Method: On entry to NUMCLU, the set of all possible means is MEAN, and the test sets are in TSPXL. Also furnished is MINCLN (which, if negative, requests exactly MINCLN (too many) clusters, or if positive at least MINCLN, and MAXCLN (which specifies the maximum number of clusters to seek). The following steps are carried out;

- (1) Classify the first and last of each test set in the nearest cluster (Euclidian distance). Save the classification in CLASS and count it in COUNT. Set LIVING = the initial number NP5 of clusters.
- (2) For each I, if COUNT(I) = 0, eliminate that cluster by setting NUM(I) = 0 and LIVING = LIVING -1.
- (3) If fewer than 101 clusters are present, go to step (6), else set IF = 1.
- (4) Eliminate successively each cluster I with COUNT(I) = IE; reclassify test pixels assigned to eliminated classes; exit (5) when the number of viable clusters falls below 101.
- (5) Increment IE and repeat (4).
- (6) Call COLAPS to remove eliminated centers.
- (7) Call DIAMTR to determine the diameter of starting clusters.
- (8) Set the initial elimination protection threshold IDIAMP. Except for the bias, IDIAMP is the diameter divided by 4\*MINCLN.
- (9) Classify each test pixel by nearest neighbor and save the classification.

- (10) Set NERR, an error counter, to zero; also set  $ERROR(I) = 0, I = 1, \dots, LIVING$ .
- (11) Count the number of times a cluster attracts a pixel from a test set and a pixel from a "far away" test set (in the order implied by the pre-sorting of test sets). Also accumulate in ERROR the count of this event per cluster.
- (12) Count the number of times a pair from the same test set is split, and credit to each of the clusters by incrementing ERROR of each.
- (13) Save the running minimum of NERR in MIN provided the current number of clusters is not greater than MAXCLN.
- (14) Determine which cluster I has ERROR(I) maximum.
- (15) Tentatively reassign test pixels assigned to class I; however, if any test pixel J is assigned more than IDIAMP away, execute (17).
- (16) Now dummy eliminate class I, set  $IDIAMP = IDIAMP - ND$ , and decrement the running number of clusters. If this number is less than or equal to MINCLN, execute (18). (Similar logic implements exactly so many clusters.) Otherwise execute step (10).
- (17) The biased distance is ID; set  $IDIAMP = (IDIAMP/3)*2 + ID/3 + ND$  and replace the mean in question by the (far away) test pixel. Replace the error counter here by half its former value and repeat step (14).
- (18) Now actually eliminate the clusters to the minimum NERR (the earlier eliminations were only dummies), and again call COLAPS to move means to the beginning of MEAN. The number of clusters is now NFCLUS. Exit.

#### Program Variables

CHKIO	SYSTEM SUBROUTINE
CLASS(NP)	INTEGER ARRAY The class a test pixel is nearest.
COLAPS	SUBROUTINE Moves the vectors in MEAN to start, eliminating gaps.
COUNT(NP5)	INTEGER ARRAY. The number of classifications a mean receives.

CSAVE(NP)            INTEGER ARRAY   Scratch array used to save classifications while checking distances.

DIAMTR              SUBROUTINE   Finds biased square of diameter of starting clusters.

DUM(NP5)             INTEGER ARRAY   Dummy, used to mark which are eliminated while computing the minimum number of errors.

ERROR(NP5)          INTEGER ARRAY   Number of errors for each cluster center; see above.

I,J,K,                INTEGER   DO loop index.

I1,I2,I3,I4,I5      INTEGER   The classification of the first through fifth of each test set pixel.

IC                    INTEGER   The classification of a test pixel far away.

ID                    INTEGER   The biased distance returned by UNCLE.

IDIAM                INTEGER   Square of diameter (biased) of starting cluster centers.

IDIAMP               INTEGER   Elimination protection threshold (biased by -32768).

IE                    INTEGER   Number of classifications to eliminate (a DO loop index).

IN                    INTEGER   Running current number of clusters being tested.

IND(1)                INTEGER ARRAY   Error indicator.

IPMC                  INTEGER   Estimate of PPMC (see Appendix A).

IPRINT                INTEGER   IDIAM with bias removed.

IT                    INTEGER   Far away index.

LIVING                INTEGER   Current number of living clusters.

MAXCLN	INTEGER	Maximum number of clusters sought.
MEAN(ND, NP5)	INTEGER ARRAY	The cluster centers.
MIN	INTEGER	Running minimum number of errors found.
MINCLN	INTEGER	Minimum number of clusters sought.
MJ	INTEGER	The class with most errors during elimination.
MP	INTEGER	Error count for seeking maximum number of errors.
ND	INTEGER	Dimensionality
NERR	INTEGER	Local count of number of errors each trial.
NFCLUS	INTEGER	Final number of clusters.
NP	INTEGER	Number of test pixels.
NP5	INTEGER	Number of means.
NPA	INTEGER	NP/4; 1/4 of the way through the set of test pixels.
NPA2	INTEGER	NPA*2; 1/2 of the way.
NUM(NP5)	INTEGER ARRAY	Indicator to point to classes eliminated. Used at first and last of the program.
PRINTEQ(66)	INTEGER ARRAY	Used for PRINTP I/O.
PRINTIT	CHARACTER	Used for core-to-core formatted write.
PRINTP	SYSTEM SUBROUTINE	
SAVE(NP5)	INTEGER ARRAY	Used to save classes eliminated (tentatively) in order.
SEEK	LOGICAL	Used to carry out logic for exactly so many clusters (if .TRUE.).
TSPXL(ND, NP)	INTEGER ARRAY	The test pixels.
UICB(1)	INTEGER ARRAY	User Information Control Block.
UNCLE	SUBROUTINE	Finds closest MEAN with NUM $\neq$ 0 to TSPXL; distance is ID, class is II.

PAGE 0030 HENLETT-P. CKARD 32102B.01.03 FORTRAN/3000 TUE, OCT 13, 1981, 9:40 AM

```

00266 01637000 SCONTROL SEGMENT=ANOEBASEG
00266 01638000 SUBROUTINE MUNCLU(NEAN,ND,NPS,HP,TSPXL,MFCLUS,MINCLM,
00266 01639000 * CLASS,COUNT,ERROR,SAVE,DUM,CSAVE,UICB,IND,MUN,MAXCLM)
00266 01640000 C
00266 01641000 C PARENT PROGRAM: MAIN
00266 01642000 C DAUGHTER PROGRAMS: UNCLE,COLAPS,DIANTR
00266 01643000 C
00266 01644000 C ---PARAMETERS IN MUNCLU---
00266 01645000 C
00266 01646000 C CLASS -- INTEGER*2 VECTOR USED TO STORE THE CLUSTER
00266 01647000 C TO WHICH A TEST PIXEL (IN TSPXL ) IS
00266 01648000 C ASSIGNED.
00266 01649000 C
00266 01650000 C COUNT -- INTEGER*2 VECTOR USED TO TALLY THE NUMBER
00266 01651000 C OF TIMES A TEST PIXEL IS ASSIGNED TO
00266 01652000 C EACH CLASS (IN NEAN )
00266 01653000 C
00266 01654000 C ERROR -- INTEGER*2 VECTOR USED TO ESTIMATE THE
00266 01655000 C RELATIVE PERFORMANCE OF EACH CLUSTER
00266 01656000 C VS ALL THE OTHERS. NOTE
00266 01657000 C
00266 01658000 C ERROR(K) IS: INCREASED BY 1 WHEN
00266 01659000 C CLUSTER K IS INVOLVED IN SPLITTING
00266 01660000 C A PAIR FROM THE SAME PATCH)
00266 01661000 C
00266 01662000 C INCREASED BY 1 WHEN
00266 01663000 C CLUSTER K ATTRACTS A PAIR FROM
00266 01664000 C DIFFERENT REAL CLASSES.
00266 01665000 C
00266 01666000 C AT EACH ELIMINATION CYCLE, THE CLUSTER WITH ERROR RELATIVELY
00266 01667000 C LARGEST IS ELIMINATED) THE AFFECT OF THIS IS
00266 01668000 C THAT A CLUSTER SURVIVES IF IT
00266 01669000 C
00266 01670000 C (A) DOES NOT SPLIT PAIRS
00266 01671000 C FROM THE SAME PATCH
00266 01672000 C
00266 01673000 C (B) DOES SPLIT PAIRS FROM
00266 01674000 C DIFFERENT PATCHES.
00266 01675000 C
00266 01676000 C SAVE -- INTEGER*2 VECTOR USED TO STORE THE ORDER
00266 01677000 C IN WHICH CLUSTERS ARE TO BE ELIMINATED
00266 01678000 C ONCE THEIR NUMBER IS DETERMINED.
00266 01679000 C
00266 01680000 C DUM -- INTEGER*2 VECTOR USED TO MARK CLUSTERS
00266 01681000 C WHICH WERE ELIMINATED IN THE INITIAL
00266 01682000 C ELIMINATION CYCLE PRIOR TO THEIR
00266 01683000 C ACTUAL ELIMINATION.
00266 01684000 C
00266 01685000 C CSAVE -- TEMPORARY VECTOR USED TO SAVE CERTAIN CLASSIFICATIONS
00266 01686000 C WHILE THE ENTIRE SET IS BEING CHECKED FOR ACCURACY.
00266 01687000 C
00266 01688000 C LIVING -- PARAMETER MARKING THE CURRENT NUMBER
00266 01689000 C OF CLUSTERS.
00266 01690000 C
00266 01691000 C NEFF -- THE NUMBER OF TEST PIXEL PAIRS WHICH
00266 01692000 C ARE SPLIT PLUS THE NUMBER OF DIFFERENT
00266 01693000 C PAIRS WHICH ARE JOINED DURING A CYCLE

```

ORIGINAL PAGE IS  
OF POOR QUALITY

PAGE 0051 MUMCLU

```

00266 01694000 C
00266 01695000 C MINCLN -- THE MINIMUM NUMBER OF CLUSTERS SOUGHT.
00266 01696000 C IF MINCLN IS NEGATIVE, EXACTLY -MINCLN
00266 01697000 C CLUSTERS ARE SOUGHT (RATHER THAN GE.)
00266 01698000 C
00266 01699000 C MAXCLN -- THE MAXIMUM NUMBER OF CLUSTERS SOUGHT (IN MUMCLU)
00266 01700000 C IN THE CURRENT VERSION, MAXCLN = 90,
00266 01701000 C ALTHOUGH IT COULD EASILY BE MADE A PROGRAM PARAMETER.
00266 01702000 C
00266 01703000 C SEEK -- A FLAG TO SWITCH BETWEEN EXACTLY AND GE
00266 01704000 C MINCLN CLUSTERS.
00266 01705000 C
00266 01706000 C IDIAN -- THE SQUARE OF THE DIAMETER OF THE STARTING CLUSTER SET
00266 01707000 C
00266 01708000 C IDIANP -- AN ELIMINATION PROTECTION THRESHOLD WHICH PREVENTS
00266 01709000 C THE LOSS OF CLUSTERS NEEDED FOR THE REASONABLE CLASS-
00266 01710000 C IFICATION OF SOME TEST PIXELS.
00266 01711000 C
00266 01712000 C MIN -- RUNNING MINIMUM NUMBER OF ERRORS.
00266 01713000 C
00266 01714000 C MIN -- NUMBER OF CLUSTERS WITH MIN ERRORS.
00266 01715000 C
00266 01716000 C INTEGER*2 TSPXL(MD,NP),REAN(MD,NP3),NUM(MPS),CLASS(NP)
00266 01717000 C ,COUNT(MPS),ERROR(MPS),SAVE(MPS),DUM(MPS),CSAVE(MP)
00266 01718000 C ,PRINTED(66),UICB(1),IND(4)
00266 01719000 C LOGICAL SEEK
00266 01720000 C CHARACTER*72 PRINTIT
00266 01721000 C EQUIVALENCE(PRINTIT,PRINTED)
00266 01722000 C MINCLN = MINCLN
00270 01723000 C DO 1111 I = 1,NP3
00275 01724000 C 1111 NUM(I) = 1
00301 01725000 C
00301 01726000 C THIS SEGMENT FOLLOWS ORDERS RE HOW MANY CLUSTERS ARE REQUIRED
00301 01727000 C SEEK = .FALSE.
00302 01728000 C IF (MINCLN.LT.0) GO TO 96
00307 01729000 C IF (MINCLN.LT.2.OR.MINCLN.GT.100) MINCLN = 5
00317 01730000 C 97 WRITE(PRINTIT,98) MINCLN
00340 01731000 C CALL PRINTP(UICB,IND,1,PRINTED,42,0,0,0,0,0,0,0)
00370 01732000 C IF (IND(1).LT.0) CALL CHKID(UICB,IND,1,0,0,0,1200)
00413 01733000 C GO TO 95
00419 01734000 C 96 MINCLN = -MINCLN
00420 01735000 C IF (MINCLN.GT.100.OR.MINCLN.LT.2) MINCLN = 15
00430 01736000 C SEEK = .TRUE
00432 01737000 C WRITE(PRINTIT,94) MINCLN
00452 01738000 C CALL PRINTP(UICB,IND,1,PRINTED,32,0,0,0,0,0,0,0)
00503 01739000 C IF (IND(1).LT.0) CALL CHKID(UICB,IND,1,0,0,0,1300)
00526 01740000 C 94 FORMAT(' EXACTLY',I3,' CLUSTERS SOUGHT ')
00526 01741000 C 95 FORMAT(' MINIMUM NUMBER OF CLUSTERS SOUGHT:',I3)
00526 01742000 C 98 MINCLN = MINCLN-1
00527 01743000 C MAXCLN = 90 *****MAXCLN NOW A PROGRAM PARAM*****
00527 01744000 C
00527 01745000 C PRINT STARTING CONDITIONS.
00527 01746000 C WRITE(PRINTIT,100) NPS,NP
00532 01747000 C CALL PRINTP(UICB,IND,1,PRINTED,47,0,0,0,0,0,0,0)
00602 01748000 C IF (IND(1).LT.0) CALL CHKID(UICB,IND,1,0,0,0,1400)
00623 01749000 C 100 FORMAT(' START WITH',I4,' CLUSTERS,',I5,' TEST POINTS ')
00623 01750000 C

```

ORIGINAL PAGE IS  
OF POOR QUALITY

PAGE 0052 MUMCLU

```

00629 01751000 C CLEAR A COUNTER OF THE NUMBER OF TIMES A CLUSTER IS HIT
00629 01752000 C BY A TEST PIXEL IN THE INITIAL CLASSIFICATION.
00629 01753000 DO 2 I = 1,NPS
00632 01754000 2 COUNT(I) = 0
00636 01755000 C
00636 01756000 C CLASSIFY AND COUNT THE CLASSIFICATION OF EVERY 5-TH TEST PIXEL.
00636 01757000 C
00636 01758000 C VERSION 11 USES AS STARTING CLUSTERS PATCH CENTERS FROM PATCHES
00636 01759000 C CONTAINING 3 OR MORE PIXELS. THIS MEANS THE TIME SPENT IN MUMCLU
00636 01760000 C IS 0(NP**3).
00636 01761000 C
00636 01762000 DO 1 I = 1,NP,5
00647 01763000 CALL UNCLC(TSPXL(I,I),NEAN,ND,NPS,11,MUR,10)
00661 01764000 COUNT(I) = COUNT(I)+1
00669 01765000 CLASS(I) = I1
00670 01766000 CALL UNCLC(TSPXL(I,I+4),NEAN,ND,NPS,11,MUR,10)
00707 01767000 COUNT(I) = COUNT(I)+1
00713 01768000 1 CLASS(I+4) = I1
00721 01769000 C
00721 01770000 C LIVING IS THE TEMPORARY NUMBER OF LIVING CLUSTERS.
00721 01771000 LIVING = NPS
00722 01772000 DO 3 I = 1,NPS
00730 01773000 IF (COUNT(I).GT.0) GO TO 3
00741 01774000 C
00741 01775000 C ELIMINATE A CLUSTER TO WHICH NOTHING IS ASSIGNED.
00741 01776000 MURCI) = 0
00744 01777000 LIVING = LIVING-1
00749 01778000 3 CONTINUE
00746 01779000 C
00746 01780000 C REPORT HOW MANY CLUSTERS WE START WITH.
00746 01781000 WRITE(PRINT11,101) LIVING
00767 01782000 101 FORMAT(' ',14,' CLUSTERS HAVE NON VOID ASSIGNMENTS.')
00767 01783000 CALL PRINTP(CUICB,IND,1,PRINTED,43,0,0,0,0,0,0,0)
01017 01784000 IF (IND(1).LT.0) CALL CHKID(CUICB,IND,1,0,0,0,1500)
01042 01785000 C
01042 01786000 C WE ARE AIMING FOR NO MORE THAN 100 TO SAVE TIME IN THE BAD LOGIC
01042 01787000 IF (LIVING LE 100) GO TO 7
01050 01788000 DO 4 IE = 1,100
01059 01789000 DO 5 I = 1,NPS
01062 01790000 IF (COUNT(I).NE.IE) GO TO 5
01067 01791000 C
01067 01792000 C ELIMINATE AN UNPOPULAR CLASS AND...
01067 01793000 LIVING = LIVING-1
01070 01794000 MURCI) = 0
01073 01795000 C
01073 01796000 C RECLASSIFY PIXELS IN THAT CLASS.
01073 01797000 DO 6 J = 1,NP,5
01100 01798000 IF (CLASS(J).NE.I) GO TO 99
01100 01799000 CALL UNCLC(TSPXL(I,J),NEAN,ND,NPS,11,MUR,10)
01123 01800000 CLASS(J) = I1
01126 01801000 COUNT(I) = COUNT(I)+1
01132 01802000 99 IF (CLASS(J+4) NE.I) GO TO 6
01141 01803000 CALL UNCLC(TSPXL(I,J+4),NEAN,ND,NPS,11,MUR,10)
01140 01804000 CLASS(J+4) = I1
01147 01805000 COUNT(I) = COUNT(I)+1
01171 01806000 6 CONTINUE
01172 01807000 C

```

PAGE 0035 HUNCLU

```

01172 01800000 C SEE IF WE ARE FINISHED.
01172 01809000 IF (LIVING.LE.100) GO TO 7
01177 01810000 3 CONTINUE
01200 01811000 4 CONTINUE
01201 01812000 C
01201 01813000 7 CALL COLAPS(MPS,NEAN,HUN,ND)
01210 01814000 IF (MINCLM.GT.LIVING) GO TO 807
01215 01815000 C
01215 01816000 C ESTIMATE THE DIAMETER OF THE STARTING SET OF CLUSTER CENTERS
01219 01817000 CALL DIANTR(NEAN,ND,LIVING,IDIAM)
01223 01818000 C
01223 01819000 C PRINT THE DIAMETER.
01223 01820000 IPRINT = IDIAM*32767+1
01227 01821000 IF (IPRINT.LT.0) IPRINT = 32767
01234 01822000 WRITE(PRINTIT,81) IPRINT
01235 01823000 81 FORMAT(' SQUARE OF DIAMETER OF STARTING CLUSTERS:',16)
01235 01824000 CALL PRINTF(UCID,IND,1,PRINTED,51,0,0,0,0,0,0,0)
01305 01825000 IF (IND(1).LT.0) CALL CHKID(UCID,IND,1,0,0,0,1566)
01320 01826000 C
01320 01827000 C SET THRESHOLD FOR ELIMINATION PROTECTION
01330 01828000 IDIAMP = IPRINT/MINCLM/4-32768
01343 01829000 C
01343 01830000 C INITIALIZE A DUMMY VECTOR OF POINTERS TO ELIMINATED CLUSTERS.
01345 01831000 DO 8 I = 1,LIVING
01352 01832000 8 DUM(I) = 1
01356 01833000 C
01356 01834000 C PREPARE TO COUNT ERRORS
01356 01835000 IN = LIVING
01360 01836000 MIN = 32767
01362 01837000 C
01362 01838000 C NPA IS 1/4 OF THE WAY ON ONE SIDE OF THE POINT WE ARE WORKING ON
01362 01839000 NPA = NP/4
01363 01840000 C
01363 01841000 C NPA2 IS 1/2 IN THE ORDER IMPLIED BY THE RESULT OF SORT.
01365 01842000 NPA2 = NPA*2
01370 01843000 C
01370 01844000 C CLASSIFY ALL TEST POINTS PRIOR TO GETTING STARTED.
01370 01845000 DO 9 I = 1,NP
01375 01846000 CALL UNCLE(TSPXL(I,1),NEAN,ND,LIVING,11,DUR,1D)
01413 01847000 9 CLASS(I) = 11
01417 01848000 C*****
01417 01849000 C
01417 01850000 C REFERENCE POINT FOR MAIN LOOP.
01417 01851000 C
01417 01852000 C INITIALIZE LOCAL COUNT OF ERRORS.
01417 01853000 11 NERR = 0
01421 01854000 DO 40 I = 1,LIVING
01426 01855000 40 ERROR(I) = 0
01433 01856000 C
01433 01857000 C GO THRU PATCHES (I.E. THRU TEST PIXELS IN SETS OF 5)
01437 01858000 DO 25 I = 1,NP,5
01437 01859000 C
01437 01860000 C GRAB A TEST SET OF 5 FROM THE SAME PATCH.
01437 01861000 I1 = CLASS(I)
01442 01862000 I2 = CLASS(I+1)
01447 01863000 I3 = CLASS(I+2)
01454 01864000 I4 = CLASS(I+3)

```

PAGE 0034 WURCLV

```

01461 01869000      IS = CLASS(I+4)
01466 01866000      C
01466 01867000      C MAKE AN INDEX INTO THE TEST PIXELS WHICH SHOULD BE FAR ENOUGH
01466 01868000      C AWAY TO BE IN A DIFFERENT REAL CLASS (ALTHOUGH THIS CANNOT BE
01466 01869000      C GUARANTEED). THEN CHECK IF IT IS OFF THE ARRAY.
01466 01870000      IT = I+NPA
01471 01871000      IF (IT.GT.NP) IT = I-NPAZ
01477 01872000      C-----
01477 01873000      C
01477 01874000      C THIS SEGMENT COUNTS THE NUMBER OF TIMES A CLUSTER ATTRACTS A
01477 01875000      C PIXEL FROM ONE OF THE TEST SETS AND FROM ONE IN THE FAR AWAY
01477 01876000      C CLASS. ONLY ONE CLUSTER IS INVOLVED IN THIS TYPE OF ERROR.
01477 01877000      DO 76 K = 1,2
01504 01878000      IC = CLASS(IT)
01507 01879000      IF (IC.NE.I1) GO TO 71
01520 01880000      ERROR(I1) = ERROR(I1)+1
01524 01881000      NERR = NERR+1
01527 01882000      71 IF (IC.NE.I2) GO TO 72
01531 01883000      ERROR(I2) = ERROR(I2)+1
01535 01884000      NERR = NERR+1
01576 01885000      72 IF (IC.NE.I3) GO TO 73
01542 01886000      ERROR(I3) = ERROR(I3)+1
01546 01887000      NERR = NERR+1
01547 01888000      73 IF (IC.NE.I4) GO TO 74
01553 01889000      ERROR(I4) = ERROR(I4)+1
01557 01890000      NERR = NERR+1
01560 01891000      74 IF (IC.NE.I5) GO TO 75
01564 01892000      ERROR(I5) = ERROR(I5)+1
01570 01893000      NERR = NERR+1
01571 01894000      C
01571 01895000      C MAKE ANOTHER INDEX FAR AWAY.
01571 01896000      75 IT = I-NPA
01574 01897000      IF (IT.LE.0) IT = I+NPAZ
01602 01898000      76 CONTINUE
01603 01899000      C-----
01603 01900000      C
01603 01901000      C
01603 01902000      C
01603 01903000      C THIS SEGMENT COUNTS THE NUMBER OF SAME PATCH -- DIFFERENT
01603 01904000      C CLUSTER ERRORS. EACH ERROR IS CREDITED TO TWO CLUSTERS
01603 01905000      C SINCE WE CAN BE CONFIDENT THAT SAME PATCH SAMPLES ARE
01603 01906000      C FROM THE SAME REAL CLASS.
01603 01907000      IF (I1.EQ.I2) GO TO 42
01607 01908000      NERR = NERR+2
01611 01909000      ERROR(I1) = ERROR(I1)+1
01615 01910000      ERROR(I2) = ERROR(I2)+1
01621 01911000      42 IF (I1.EQ.I3) GO TO 44
01623 01912000      NERR = NERR+2
01627 01913000      ERROR(I1) = ERROR(I1)+1
01633 01914000      ERROR(I3) = ERROR(I3)+1
01637 01915000      44 IF (I1.EQ.I4) GO TO 46
01643 01916000      NERR = NERR+2
01644 01917000      ERROR(I1) = ERROR(I1)+1
01651 01918000      ERROR(I4) = ERROR(I4)+1
01655 01919000      46 IF (I1.EQ.I5) GO TO 48
01661 01920000      NERR = NERR+2
01663 01921000      ERROR(I1) = ERROR(I1)+1

```

PAGE 0055 HUNCLU

```
01667 01922000      ERROR(15) = ERROR(15)+1
01673 01923000      40 IF (12.EQ.13) GO TO 50
01677 01924000      NERR = NERR+2
01701 01925000      ERROR(12) = ERROR(12)+1
01705 01926000      ERROR(13) = ERROR(13)+1
01711 01927000      50 IF (12.EQ.14) GO TO 52
01719 01928000      NERR = NERR+2
01717 01929000      ERROR(12) = ERROR(12)+1
01723 01930000      ERROR(14) = ERROR(14)+1
01727 01931000      52 IF (12.EQ.15) GO TO 54
01733 01932000      NERR = NERR+2
01735 01933000      ERROR(12) = ERROR(12)+1
01741 01934000      ERROR(15) = ERROR(15)+1
01749 01935000      54 IF (13.EQ.14) GO TO 56
01751 01936000      NERR = NERR+2
01753 01937000      ERROR(13) = ERROR(13)+1
01757 01938000      ERROR(14) = ERROR(14)+1
01763 01939000      56 IF (13.EQ.15) GO TO 58
01767 01940000      NERR = NERR+2
01771 01941000      ERROR(13) = ERROR(13)+1
01775 01942000      ERROR(15) = ERROR(15)+1
02001 01943000      58 IF (14.EQ.15) GO TO 25
02005 01944000      NERR = NERR+2
02007 01945000      ERROR(14) = ERROR(14)+1
02013 01946000      ERROR(15) = ERROR(15)+1
02017 01947000      C
02017 01948000      C ALL COMPILED FOR THIS PATCH .....
02017 01949000      C
02017 01950000      C GET THE NEXT PATCH.
02017 01951000      C 25 CONTINUE
02026 01952000      C
02026 01953000      C SEE IF ERRORS ARE LESS THAN BEFORE.
02026 01954000      C   IF (NERR GE. MIN. OR. IN. GE. MAXCLN) GO TO 102
02027 01955000      C
02027 01956000      C YES...UPDATE RUNNING MIN
02027 01957000      C   MIN = NERR
02041 01958000      C   MIN = IN
02033 01959000      C
02033 01960000      C THIS SEGMENT CARRIES OUT THE LOGIC DICTATED BY THE NEED FOR
02033 01961000      C EXACTLY SO MANY CLUSTERS. (IT JUST GREW THIS WAY.)
02033 01962000      C 102 IN = IN-1
02034 01963000      C   IF (BECK) MIN = IN+1
02041 01964000      C   IF (IN.LE. MINCLN) GO TO 103
02043 01965000      C   NP = -32767
02047 01966000      C
02047 01967000      C FIND THE CLUSTER WHICH SEEMS TO CAUSE THE MOST ERRORS.
02047 01968000      C   DO 61 J = 1,LIVING
02054 01969000      C   IF (OUN(J).EQ.0) GO TO 61
02062 01970000      C   IF (ERROR(J).LE.NP) GO TO 61
02067 01971000      C   NP = ERROR(J)
02072 01972000      C   NJ = J
02074 01973000      C 61 CONTINUE
02075 01974000      C
02075 01975000      C STACK THE INDEX OF THIS MOST OFFENSIVE CRITER
02075 01976000      C   SAVE(IN) = A1
02100 01977000      C
02100 01978000      C AND MARK IT DONE (IF NOT DEAD)
```

PAGE 0056 NUNCLU

```

02100 01970000      DUN(NJ) = 0
02103 01980000  C RECLASSIFY ALL POINTS WHICH WERE ASSIGNED TO THIS CLUSTER.
02103 01981000  C
02103 01982000  C HOWEVER, IF A TEST PIXEL ASSIGNED TO THIS CLUSTER IS, AFTER THE
02103 01983000  C ELIMINATION OF THIS CLUSTER, REASSIGNED TO ONE MORE THAN IDIAMP
02103 01984000  C AWAY, ELIMINATE THE CLUSTER WITH NEXT FEWEST ERRORS.
02103 01993000      DO 65 I = 1,NP
02110 01986000      IF (CLASS(I).NE.NJ) GO TO 65
02113 01987000      CALL UNCLE(TSPXL(I,I),NEAN,ND,LIVING-11,DUN,10)
02133 01988000  C SEE IF DISTANCE.LE.THRESHOLD
02133 01989000      IF (ID.GT.IDIAMP) GO TO 1000
02140 01990000  C
02140 01991000  C SAVE THIS CLASSIFICATION.
02140 01992000      CSAVE(I) = 11
02143 01993000      65 CONTINUE
02144 01994000      IDIAMP = IDIAMP-ND
02147 01993000  C
02147 01996000  C YES, ALL DISTANCES ARE OK...
02147 01997000  C NOW ACTUALLY CLASSIFY THE POINTS
02147 01998000      DO 66 I = 1,NP
02154 01999000      IF (CLASS(I).NE.NJ) GO TO 66
02161 02000000      CLASS(I) = CSAVE(I)
02164 02001000      66 CONTINUE
02165 02002000      GO TO 11
02167 02003000  C
02167 02004000  C DON'T ZAP CLASS NJ AFTER ALL
02167 02005000      1000 IN = IN+1
02170 02006000      DUN(NJ) = 1
02173 02007000      ERROR(NJ) = ERROR(NJ)/2
02177 02008000  C      DO 813 K = 1,ND
02204 02009000  C      813 NEAN(K,NJ) = TSPXL(K,I)
02221 02010000  C
02221 02011000  C INCR THE THRESHOLD TO INSURE NO INFINITE LOOP.
02221 02012000      IDIAMP = (IDIAMP/3)*2+ID/3+ND
02231 02013000      GO TO 102
02232 02014000  C*****
02232 02015000  C
02232 02016000  C I SUPPOSE WE ARE THROUGH NOW...REPORT!
02232 02017000      103 NFCLUS = NIN
02234 02018000      WRITE(PRINTIT,10) NFCLUS
02235 02019000      10 FORMAT('      NUMBER OF CLUSTERS:',I4)
02235 02020000      CALL PRINTP(QUICB,IND,1,PRINTED,20,0,0,0,0,0,0,0)
02305 02021000      IF (IND(1).LT.0) CALL CNKID(QUICB,IND,1,0,0,0,2000)
02330 02022000      IPNC = IFIX(25.*FLOAT(NIN)/FLOAT(NP))
02342 02022100      IPNC = IPNC+2/3
02346 02022200  C
02346 02022300  C ALLOWS FOR COUNTING ERRORS TWICE... JUST AN ESTIMATE.
02346 02023000      WRITE(PRINTIT,104) IPNC
02367 02024000      104 FORMAT('      ESTIMATE OF PAIR PNC',I4,' PERCENT. ')
02367 02025000      CALL PRINTP(QUICB,IND,1,PRINTED,39,0,0,0,0,0,0,0)
02417 02026000      IF (IND(1).LT.0) CALL CNKID(QUICB,IND,1,0,0,0,2100)
02442 02027000      NIN = NIN+1
02443 02028000      LVNCH = LIVING-1
02446 02029000  C
02446 02030000  C ITS HARD TO SEE HOW THIS COULD HAPPEN, BUT...
02446 02031000      IF (NIN.GT.LVNCH) R E T U R N
02460 02032000  C

```

C-2

PAGE 0057 NUNCLU

```

02460 02033000 C ACTUALLY PERFORM THE ELIMINATION NOW.
02460 02034000      00 62 I = MIN.LVNGH
02462 02033000      62 NUN(SAVE(I)) = 0
02472 02036000 C
02472 02037000 C AGAIN COLLAPSE THE ARRAY TO ALLOW MORE EFFICIENT SEARCH.
02472 02038000      CALL COLAPS(LIVING,NEAR,NUN,ND)
02501 02039000      R E T U R N
02502 02040000      007 NFCLUS = LIVING
02504 02041000      R E T U R N
02505 02042000      END
    
```

SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
CMK10		SUBROUTINE		CLASS	INTEGER	ARRAY	0-215 .I
COLAPS		SUBROUTINE		COUNT	INTEGER	ARRAY	0-214 .I
CSAVE	INTEGER	ARRAY	0-210 .I	DIANTR		SUBROUTINE	
DUN	INTEGER	ARRAY	0-211 .I	ERROR	INTEGER	ARRAY	0-213 .I
I	INTEGER	SIMPLE VAR	0+26	I1	INTEGER	SIMPLE VAR	0+227
I2	INTEGER	SIMPLE VAR	0+231	I3	INTEGER	SIMPLE VAR	0+232
I4	INTEGER	SIMPLE VAR	0+233	I5	INTEGER	SIMPLE VAR	0+234
IC	INTEGER	SIMPLE VAR	0+27	ID	INTEGER	SIMPLE VAR	0+210
IDIAN	INTEGER	SIMPLE VAR	0+236	IDIANP	INTEGER	SIMPLE VAR	0+215
IE	INTEGER	SIMPLE VAR	0+211	IN	INTEGER	SIMPLE VAR	0+216
IND	INTEGER	ARRAY	0-26 .I	IPAC	INTEGER	SIMPLE VAR	0+241
IPRINT	INTEGER	SIMPLE VAR	0+237	IT	INTEGER	SIMPLE VAR	0+223
J	INTEGER	SIMPLE VAR	0+221	K	INTEGER	SIMPLE VAR	0+235
LIVING	INTEGER	SIMPLE VAR	0+230	LVNGH	INTEGER	SIMPLE VAR	0+226
MAXCLN	INTEGER	SIMPLE VAR	0-24 .I	NEAR	INTEGER	ARRAY	0-224 .I
MIN	INTEGER	SIMPLE VAR	0+240	NINCL	INTEGER	SIMPLE VAR	0+217
NJ	INTEGER	SIMPLE VAR	0+224	NP	INTEGER	SIMPLE VAR	0+225
ND	INTEGER	SIMPLE VAR	0-223 .I	NER2	INTEGER	SIMPLE VAR	0+222
NFCLUS	INTEGER	SIMPLE VAR	0-217 .I	NI	INTEGER	SIMPLE VAR	0+214
NINCLN	INTEGER	SIMPLE VAR	0-216 .I	NP	INTEGER	SIMPLE VAR	0-221 .I
NPS	INTEGER	SIMPLE VAR	0-222 .I	NPA	INTEGER	SIMPLE VAR	0+212
NPA2	INTEGER	SIMPLE VAR	0+213	NUN	INTEGER	ARRAY	0-25 .I
NUNCLU		SUBROUTINE		PRINTED	INTEGER	ARRAY	0+24 .I
PRINTST	CHARACTER	SIMPLE VAR	0+25 .I	PRINTP		SUBROUTINE	
SAVE	INTEGER	ARRAY	0-212 .I	SEEK	LOGICAL	SIMPLE VAR	0+220
TSPXL	INTEGER	ARRAY	0-220 .I	UICD	INTEGER	ARRAY	0-27 .I
UNCLC		SUBROUTINE					

PROGRAM UNIT NUNCLU COMPILED

Parent: CLASSIFY

PERPIXEL

Calls: REJECTH

PERPIXEL(PIXELS,CLUSTERS,LABELS,ND,NC,NFCLUS,REJECT,MAXCLUS,MASK,NX)

Performs a per pixel classification of a line.

Method: This subroutine performs a checked per pixel classification of multidimensional data in PIXELS to classes in CLUSTERS. Nearest neighbor (Euclidean distance) classification is employed. However, when the distance is too great, the classification is rejected, and (generally) a new cluster center is added. On that event, the cluster reject thresholds are recomputed and this line classification is restarted. Labels (i.e. class numbers) are written in line LABELS.

The program is straightforward with the possible exception of the biased distances and the modification of rejection thresholds in loop 120. All distances are biased by -32768; the rejection thresholds are also biased by -32768 in REJECTH. We wish to make the test less severe than was applied to pure pixels (the test pixels), so we logically multiply the thresholds by 2 (which, in terms of distances, amounts to deciding reject on  $0.7 * \text{distance between competing classes}$  rather than  $0.5 * \text{distance between}$ ). Thus, if  $r$  is the biased and  $R$  the logical threshold, we have  $r = R - 32768$ , so that

$$\begin{aligned} r' &= 2R - 32768 = 2(R+32768) - 32768 \\ &= 2R + 32768 . \end{aligned}$$

Program Variables

AO                    INTEGER\*4   Long integer to carry out the threshold arithmetic intermediate steps.

CLUSTERS(ND,MAXCLUS)    INTEGER ARRAY   The clusters.

I,J,K                    INTEGER   DO loop index.

ICL                      INTEGER   Class number of nearest cluster.

IM                    INTEGER   Distance to nearest cluster

IS                    INTEGER   Accumulator for computing distance.

IT                    INTEGER   Temporary for computing distance.

JR                    INTEGER   Reject threshold of current class.

LABELS(NC)            INTEGER ARRAY   The classification.

MASK                  LOGICAL   If .TRUE., a value 0 in channel 1 is used  
as a mask (i.e. not data), and the label 99 is  
stored in LABELS.

MAXCLUS              INTEGER   Maximum number of clusters (current value,  
set in MAIN, is 98).

NC                    INTEGER   Number of samples per line.

ND                    INTEGER   Dimensionality.

NFCLUS                INTEGER   Current number of clusters.

NX                    INTEGER   Actual number of samples per line (approximately  
NC, but often a little less). NC is used to dimension  
things; NX may vary from one call to the next.

PIXELS(NC,ND)        INTEGER ARRAY   One line of data.

REJECT(MAXCLUS)      INTEGER ARRAY   The reject thresholds.

REJECTH               SUBROUTINE   Calculates REJECT.

PAGE 0000 HEWLETT-PACKARD 32102D.01.03 FORTRAN/3000 TUE, OCT 13, 1981, 9:36 AM

```

00011 00316000 $CONTROL SEGMENT=ANDBASEG
00011 00317000     SUBROUTINE PERPIXEL(PIXELS,CLUSTERS,LABELS,ND,NC,NFCLUS,
00011 00318000     * REJECT,MAXCLUS,NAK,NX)
00011 00319000     LOGICAL MASK
00011 00320000     INTEGER*4 AO
00011 00321000     INTEGER*2 PIXELS(NC,ND),CLUSTERS(ND,MAXCLUS),LABELS(NC),
00011 00322000     * REJECT(MAXCLUS)
00011 00323000 130 DO 10 I = 1,NX
00017 00324000     IF (.NOT MASK) GO TO 30
00022 00325000     IF (PIXELS(I,1).NE.0) GO TO 30
00022 00326000     ICL = 99
00024 00327000     GO TO 10
00027 00328000     30 CONTINUE
00029 00329000     IN = 32767
00027 00330000     ICL = 0
00041 00331000     DO 30 J = 1,NFCLUS
00046 00332000     IS = -32768
00052 00333000     JR = REJECT(J)
00060 00334000     DO 20 K = 1,ND
00065 00335000     IF (IS GE JR) GO TO 30
00072 00336000     IT = PIXELS(I,K)-CLUSTERS(K,J)
00112 00337000     IS = IS+IT*IT
00116 00338000     IF (IS GE JR) GO TO 30
00122 00339000     20 CONTINUE
00124 00340000     IF (IN LE IS) GO TO 30
00126 00341000     IN = IS
00122 00342000     ICL = J
00124 00343000     30 CONTINUE
00135 00344000     IF (ICL.EQ.0) GO TO 100
00141 00345000     10 LABELS(I) = ICL
00150 00346000     R E T U R N
00151 00347000 100 IF (NFCLUS GE MAXCLUS) GO TO 10
00153 00348000     NFCLUS = NFCLUS+1
00156 00349000     ICL = NFCLUS
00160 00350000     DO 110 K = 1,ND
00162 00351000     110 CLUSTERS(K,ICL) = PIXELS(I,K)
00202 00352000     CALL REJECTH(CLUSTERS,ND,REJECT,NFCLUS,MAXCLUS)
00212 00353000     DO 120 K = 1,NFCLUS
00217 00354000     AO = REJECT(K)
00224 00355000     AO = AO+32768
00224 00356000     IF (AO GT 16000) AO = 16000
00246 00357000     120 REJECT(K) = AO
00257 00358000     GO TO 130
00263 00359000     END
    
```

SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
AO	INTEGER*4	SIMPLE VAR	0-214	CLUSTERS	INTEGER	ARRAY	0-214 .I
I	INTEGER	SIMPLE VAR	0-214	ICL	INTEGER	SIMPLE VAR	0-212
IF	INTEGER	SIMPLE VAR	0-216	IS	INTEGER	SIMPLE VAR	0-210
IT	INTEGER	SIMPLE VAR	0-211	J	INTEGER	SIMPLE VAR	0-217
JR	INTEGER	SIMPLE VAR	0-215	K	INTEGER	SIMPLE VAR	0-213
LABELS	INTEGER	ARRAY	0-213 .I	MASK	LOGICAL	SIMPLE VAR	0-215 .I
MAXCLUS	INTEGER	SIMPLE VAR	0-216 .I	NC	INTEGER	SIMPLE VAR	0-211 .I
NC	INTEGER	SIMPLE VAR	0-212 .I	NFCLUS	INTEGER	SIMPLE VAR	0-210 .I
ND	INTEGER	SIMPLE VAR	0-214 .I	PERPIXEL		SUBROUTINE	
PIXELS	INTEGER	ARRAY	0-210 .I	REJECT	INTEGER	ARRAY	0-217 .I
REJECTH		SUBROUTINE					

PROGRAM UNIT PERPIXEL COMPILED

Parents: MOREQUES, PERPIXEL

REJECTH

REJECTH(CLUSTERS,ND,REJECT,NFCLUS,MAXCLUS)

This subroutine determines the cluster-dependent classification rejection thresholds.

Method: Using a bias of -32768, the inter-cluster squared distance from cluster I to cluster J is maximized. The maximum over J, divided by four, is the classification rejection threshold for cluster I, and is stored in REJECT(I). To minimize round-off error, the division by four is carried out in two stages; the second accounts for the bias:  $R = D - 32768$ , so  $D/2 - 32768 = R/2 - 16384$  should be stored as the twice halved square of distance.

Program Variables

CLUSTERS(ND,MAXCLUS)	Integer ARRAY	The cluster centers.
I,J,K,	INTEGER	DO loop index.
IM,IS,IT	INTEGER	Used in calculation and maximization of distances.
MAXCLUS	INTEGER	Maximum number of clusters.
ND	INTEGER	Dimensionality
NFCLUS	INTEGER	Actual number of clusters.
REJECT(MAXCLUS)	INTEGER ARRAY	Rejection thresholds.

PRECEDING PAGE BLANK NOT FILMED

PAGE 0025 HEWLETT-PACKARD J21020.01.03 FORTRAN/3000 TUE, OCT 13, 1961, 9:37 AM

```

00006 00752000 $CONTROL SEGMT=ANOBASEC
00006 00753000     SUBROUTINE REJECTN(CLUSTERS,ND,REJECT,NFCLUS,MAXCLUS)
00006 00754000     INTEGER*2 CLUSTERS(ND,MAXCLUS),REJECT(MAXCLUS)
00006 00755000     DO 10 I = 1,NFCLUS
00013 00756000 C
00013 00757000 C FIND THE CRITER THE GREATST DISTANCE AWAY
00013 00758000 C
00013 00759000     IN = -32768
00022 00760000     DO 20 J = 1,NFCLUS
00027 00761000     IF (I.EQ.J) GO TO 20
00035 00762000 C
00035 00763000 C FIND DIST(I,J)*0.2
00035 00764000 C
00035 00765000     IS = -32768
00044 00766000     DO 30 K = 1,ND
00051 00767000     IT = (CLUSTERS(K,J)-CLUSTERS(K,I))/2
00067 00768000     30 IS = MIN(16383,IS+IT*IT)
00100 00769000     IF (IS.LE.IN) GO TO 20
00105 00770000     IN = IS
00107 00771000     20 CONTINUE
00110 00772000     REJECT(I) = IN/2-16384
00115 00773000     10 CONTINUE
00116 00774000     R E T U R N
00120 00775000     E N D

```

## SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
CLUSTERS	INTEGER	ARRAY	0-210 ,I	I	INTEGER	SIMPLE VAR	0+23
IN	INTEGER	SIMPLE VAR	0+24	IS	INTEGER	SIMPLE VAR	0+26
IT	INTEGER	SIMPLE VAR	0+27	J	INTEGER	SIMPLE VAR	0+25
K	INTEGER	SIMPLE VAR	0+210	MAXCLUS	INTEGER	SIMPLE VAR	0-24 ,I
ND	INTEGER	SIMPLE VAR	0-27 ,I	NFCLUS	INTEGER	SIMPLE VAR	0-25 ,I
REJECT	INTEGER	ARRAY	0-26 ,I	REJECTN		SUBROUTINE	

PROGRAM UNIT REJECTN COMPILED

PRECEDING PAGE BLANK NOT FILMED

Parent: MAIN

SETSYM

SETSYM(SYMBOL)

Sets SYMBOL to the appropriate character set for display in MAPP.

Method: Because of the restrictions of HP FORTRAN-SEGMENTER and of IDIMS, this usually straightforward task is 14 lines long. The result is to set SYMBOL(1) equal to a blank and SYMBOL(2) to SYMBOL(59) (printable) ASCII characters between 33 and 90 (decimal).

Program Variables

CC(2)	CHARACTER*1 ARRAY Equivalenced to IC.
IC	INTEGER Used to transfer ASCII binary to characters.
J	INTEGER DO loop index.
NOX	INTEGER Index into SYMBOL.
SYMBOL(59)	CHARACTER*1 ARRAY The symbol set.

PRECEDING PAGE BLANK NOT FILMED

PAGE 0014 HEWLETT-PACKARD 32102B.01.03 FORTRAN/3000 TUE, OCT 13, 1981, 9:37 AM

```

00017 00464000 #CONTROL SEGMENT=AMOEBASEG
00017 00465000     SUBROUTINE SETSYN(SYMBOL)
00017 00466000     CHARACTER*(*) CC(2),SYMBOL(1)
00017 00467000     EQUIVALENCE (CC(1),IC)
00017 00468000     NOX = 2
00017 00469000     IC = 32
00021 00470000     SYMBOL(1) = CC(2)
00021 00471000 10 DO 20 J = 33,90
00021 00472000     IC = J
00021 00473000     SYMBOL(NOX) = CC(2)
00021 00474000     NOX = NOX+1
00021 00475000     IF (NOX.GT.59) R E T U R N
00021 00476000 20 CONTINUE
00021 00477000     GO TO 10
00021 00478000     END

```

## SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
CC	CHARACTER	ARRAY	0+32 .1	IC	INTEGER	SIMPLE VAR	0+31 .1
J	INTEGER	SIMPLE VAR	0+34	NOX	INTEGER	SIMPLE VAR	0+33
SETSYN		SUBROUTINE		SYMBOL	CHARACTER	ARRAY	0+34 .1

PROGRAM UNIT SETSYN COMPILED

PRECEDING PAGE BLANK NOT FILMED

Parent: MSORT, SORT

SHELL

SHELL(KEY,INDEX,NUMBER)

Sorts KEY into increasing order; the elements of KEY are not interchanged. Rather, the array INDEX is permuted so that  $KEY(INDEX(I)) \leq KEY(INDEX(J))$  when  $I < J$ .

Method: See Shell, D. L., A high speed sorting program, Comm. ACM 2 (1959), 30-32.

Program Variables

INDEX(NUMBER)      INTEGER ARRAY    On entry, INDEX(I) = I; on exit, INDEX is permuted.

KEY(NUMBER)        INTEGER ARRAY    The list to be sorted.

NUMBER             INTEGER          The number of items in the list.

I,IM,IT,J,K,M      INTEGER          Internal variables

PRECEDING PAGE BLANK NOT FILLED

PAGE 0012 HEULETT-PACKARD 31102B.01.03 FORTRAN/3000 TUE, OCT 13, 1981, 9:36 AM

```

00000 00412000 C CONTROL SEGMENT=ANDBASEC
00000 00413000 C SUBROUTINE SNELL(KEY,INDEX,NUMBER)
00000 00414000 C
00000 00415000 C DIMENSION PROGRAM( SORT
00000 00416000 C INTEGER*2 KEY(NUMBER),INDEX(NUMBER)
00000 00417000 C
00000 00418000 C FAST SORT ROUTINE: IT IS ASSUMED THAT INDEX(K) = K ON ENTRY
00000 00419000 C ON EXIT, KEY(INDEX(I)) LE KEY(INDEX(I+1)), I = 1,NUMBER-1.
00000 00420000 C
00000 00421000 C REFERENCE:
00000 00422000 C
00000 00423000 C SNELL, D.L., A HIGH SPEED SORTING PROCEDURE, COMM
00000 00424000 C ACH, (1959),30-32.
00000 00425000 C
00000 00426000 C N = NUMBER
00000 00427000 C 10 N = N/2
00000 00428000 C IF (N EQ 0) R E T U R N
00000 00429000 C K = NUMBER-N
00000 00430000 C J = 1
00000 00431000 C 20 I = J
00000 00432000 C 30 IH = I+H
00000 00433000 C IF (KEY(INDEX(I)) LE KEY(INDEX(IH))) GO TO 40
00000 00434000 C IT = INDEX(I)
00000 00435000 C INDEX(I) = INDEX(IH)
00000 00436000 C INDEX(IH) = IT
00000 00437000 C I = I+H
00000 00438000 C IF (I GE N) GO TO 30
00000 00439000 C 40 J = J+1
00000 00440000 C IF (J GT K) GO TO 10
00000 00441000 C GO TO 20
00000 00442000 C END
    
```

SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
I	INTEGER	SIMPLE VAR	0+32	IN	INTEGER	SIMPLE VAR	0+23
INDEX	INTEGER	ARRAY	0-25 ,1	IT	INTEGER	SIMPLE VAR	0+25
J	INTEGER	SIMPLE VAR	0+24	K	INTEGER	SIMPLE VAR	0+27
KEY	INTEGER	ARRAY	0-26 ,1	N	INTEGER	SIMPLE VAR	0+26
NUMBER	INTEGER	SIMPLE VAR	0-24 ,1	SNELL		SUBROUTINE	

PROGRAM UNIT SNELL COMPILED

Parent: MAIN

SORT

Calls: SHELL

SORT(TSPXL,DUMMY,INDEX,ND,NP,NPS)

Sorts TSPXL in increasing order of the sum of odd channels of the first test pixel in each test set. Sets are kept together.

Method: First the sums of odd bands of the first test pixel in each test set are accumulated, and an index into the sets is formed so that INDEX(I) = I. Then SHELL is called. On return from SHELL, the test sets are reordered by the permutation of INDEX induced by SHELL. The actual sets are now switched in place.

Program Variables

DUMMY(NP5) INTEGER ARRAY Used to accumulate sums in odd bands of the first element of each test set, and then as a temporary vector while TSPXL is being reordered.

I INTEGER DO loop index.

INDEX(NP5) INTEGER ARRAY The pointer array, used by SHELL to indicate order of DUMMY.

J INTEGER DO loop index

K INTEGER DO loop index.

L INTEGER DO loop index; note: the index into TSPXL is  $I * 5 + L - 5$ , where I is the test set number and L is the number,  $I = 1, \dots, NP5$ ,  $L = 1, \dots, 5$ .

ND INTEGER Dimensionality of TSPXL.

NP INTEGER Number of test pixels.

NP5 INTEGER Number of test sets (= NP/5).

SHELL SUBROUTINE Sorts vector in increasing order.

TSPXL(ND,NP) INTEGER ARRAY Test pixels, organized as vectors, then sets, then count of sets, to be reordered by increasing sum of odd bands.

PRECEDING PAGE BLANK NOT FILMED

PAGE 0011 HEULETT-PACKARD 32102B 01.02 FORTRAN/3000 TUE, OCT 13, 1961, 9:36 AM

```

00010 00377000 SCONTROL SEGMENT=ANODEBASEG
00010 00378000 SUBROUTINE SORT(TSPXL,DUMMY,INDEX,ND,NP,NPS)
00010 00379000 C
00010 00380000 C FAFENT PROGRAM: MAIN
00010 00381000 C DAUGHTER PROGRAM: SHELL
00010 00382000 C
00010 00383000 C INTEGER*2 TSPXL(ND,NP),DUMMY(NPS),INDEX(NPS)
00010 00384000 C
00010 00385000 C THIS SUBROUTINE SORTS TSPXL INTO INCREASING ORDER
00010 00386000 C ON SU7 OF 060 CHANNELS ( = AVERAGE BRIGHTNESS?)
00010 00387000 C
00010 00388000 C PLACE THE AVERAGE BRIGHTNESS OF TEST PIXEL I IN DUMMY(K)
00010 00389000 C WHERE I = 1+5*(K-1) THE BRIGHTNESS IS ASSUMED TO BE IN ODD
00010 00390000 C CHANNELS A LA KAUTH-THOMAS OR OTHER.
00010 00391000 K = 1
00012 00392000 DO 1 I = 1,NP,5
00017 00393000 DUMMY(K) = 0
00022 00394000 DO 1000 J = 1,ND,2
00027 00395000 1000 DUMMY(K) = TSPXL(K)+TSPXL(J,I)
00041 00396000 INDEX(K) = K
00044 00397000 1 K = K+1
00046 00398000 C
00046 00399000 C SOFT DUMMY
00046 00400000 CALL SHELL(DUMMY,INDEX,NPS)
00054 00401000 C
00054 00402000 C NOW PERFORM ACTUAL SWITCHING.
00054 00403000 100 DO 110 K = 1,ND
00061 00404000 DO 110 L = 1,5
00065 00405000 DO 120 I = 1,NPS
00073 00406000 120 DUMMY(I) = TSPXL(K,1+5-5+L)
00110 00407000 DO 130 I = 1,NPS
00115 00408000 130 TSPXL(K,1+5-5+L) = DUMMY(I)
00121 00409000 110 CONTINUE
00127 00410000 R E T U R N
00124 00411000 END
    
```

SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
DUMMY	INTEGER	ARRAY	0-210 ,1	I	INTEGER	SIMPLE VAR	0-24
INDEX	INTEGER	ARRAY	0-27 ,1	J	INTEGER	SIMPLE VAR	0-26
K	INTEGER	SIMPLE VAR	0-27	L	INTEGER	SIMPLE VAR	0-25
ND	INTEGER	SIMPLE VAR	0-26 ,1	NP	INTEGER	SIMPLE VAR	0-25 ,1
NPS	INTEGER	SIMPLE VAR	0-24 ,1	SHELL		SUBROUTINE	
SORT		SUBROUTINE		TSPXL	INTEGER	ARRAY	0-211 ,1

PROGRAM UNIT SORT COMPILED

Parent: MAIN

START

Calls: MARKLP, MARKKUPDN, MRKIUL, CONNCT

START(INTTHR,ND,NR,NC,NZ,DATBUF,LABBUF,ISCAN,UICB,IND,IMGIN,LMGCLAS,LAB,  
MASK)

START makes the vector boundary decisions and returns a map (on disk) of the components of the complement of the boundary.

Method: The program employs Wide Image Logic to segment the image into strips. Three lines of data and three lines of labels are active at any given time. Circular Buffers using pointers I1, I2, and I3, manage this; I1 always points to the eldest and I3 to the newest. Even so, we can't phrase the description of the method in absolute terms (i.e. as if the entire image were in memory at once). The initial phase must be described first.

Initially, a labels buffer is set to all one. Then the data is scanned, and points for which the vector gradient test is failed in the left-right direction are marked (subroutine MARKLR). Then up-down boundaries are located (subroutine MARKUPDN), and one-pixel gaps in the boundary map are filled (subroutine FILLR). Then, the logical OR of the first and second line replaces the first. (These are details, but this is detailed documentation, and FORTRAN loves it.) We now have an excellent estimate of the boundary in the first two lines and a fair first cut on the third. Subroutine MRKIVL is called to mark intervals in line 1. MRKIVL replaces each interval contained in the complement of the boundary along a line with (successively) 1,2,... . If any patch slices were found in this step, then the first line is initialized with patch labels (starting at -32767).

Now the big loop is entered. Intervals in line I2 are marked. If any patches are found here, subroutine CONNCT is used to transfer old labels to new intervals and to begin new labels. Row I1 labels are written and new data is read into DATBUF(.,.,I1). New labels are marked with 1 and the circular buffer pointers are rotated. (Now I3 is the newest). We now call MARKLR (I3) and then MARKUPDN, and return to the big loop. Exit

from the loop when we run out of data (or labels). If out of data (or at the bottom of a strip), process lines I2 then I3 and write on disk. If out of labels, simply paint the rest of the image boundary. (Since 64K labels are allowed, this seems unlikely to happen on even the largest natural images.)

The initialization phase guarantees labels are not propagated from the bottom of one strip to the top of the next.

### Program Variables

CHKIO	SYSTEM SUBROUTINE
CONNCT	SUBROUTINE Transfers labels from one line to the next, and begins new labels when no connection exists.
DATBUF(NC,ND,3)	INTEGER ARRAY Three lines of data, organized as a circular buffer.
FILLR	SUBROUTINE Replaces a boundary-not boundary-boundary gap by three boundary points along a line.
FINISHED	LOGICAL Returned .TRUE. by CONNCT when no more labels exist.
I,ITM,J,K	INTEGER DO loop index.
I1,I2,I3	INTEGER Circular buffer pointers
IMGCLAS	INTEGER Image number of the (temporary) labels-boundary disk maps.
IMGIN	INTEGER Image number of the data.
IN	INTEGER The number of intervals found by MRKIVL in a line.
IND(1)	INTEGER ARRAY Error indicator
INTTHR(ND)	INTEGER ARRAY The vector thresholds obtained by THRFND.
IROW	INTEGER Line number being read.
IROWOUT	INTEGER Line of labels being written
ISCAN(1)	INTEGER ARRAY Scratch array used by CONNCT to store labels.

IT	INTEGER	Used to rotate pointers.
LAB	INTEGER	Current label number.
LABBUF(NC,3)	INTEGER ARRAY	Three lines of labels.
MARKLR	SUBROUTINE	Mark left-right boundaries depending on vector thresholds.
MARKUPDN	SUBROUTINE	Mark up-down boundaries.
MASK	LOGICAL	The Mask flag.
MRKIVL	SUBROUTINE	Mark intervals along a line in the complement of the boundary.
NC	INTEGER	Number of columns in a strip.
ND	INTEGER	Dimensionality
NW,NX,NY	INTEGER	Used to partition the image into strips (see Wide Image Logic).
NZ	INTEGER	Actual number of samples.
READP	SYSTEM SUBROUTINE	
UICB(1)	INTEGER ARRAY	User Information Control Block
WRITEP	SYSTEM SUBROUTINE	
Z	INTEGER	-32768; boundary marker.

PAGE 0032 HEULETT-PACKARD 321020 01.03 FORTRAN/3000 TUE, OCT 13, 1981, 9:38 AM

```

00012 00995000 8CONTROL SEGMENT=ANDBASEC
00012 00996000      SUBROUTINE START(INTTHR,ND,NR,NC,NZ,DATBUF,LABBUF,
00012 00997000      *ISCAN,UICB,IND,INGIN,INGCLAS,LAB,NASK)
00012 00998000      INTEGER*2 Z,INTTHR(ND),DATBUF(NC,ND,J),LABBUF(NC,J),ISCAN(1),
00012 00999000      * UICB(1),IND(1)
00012 01000000      LOGICAL FINISHED,NASK
00012 01001000      FINISHED = .FALSE.
00013 01002000      Z = -32768
00022 01003000      LAB = Z
00024 01004000  C
00024 01005000  C PROCESS IN STRIPS ABOUT NC WIDE...
00024 01006000      DO 96 NU = 1,NZ,NC
00031 01007000  C
00031 01008000  C NY IS LAST OF STRIP
00031 01009000      NY = NU*NC-1
00032 01010000      IF (NY.GT.NZ) NY = NZ
00042 01011000  C
00042 01012000  C NX IS ACTUAL NUMBER CRABBED
00042 01013000      NX = NY-NU+1
00046 01014000      IRQU = 3
00050 01015000      IRQUOT = 0
00052 01016000      I1 = 1
00054 01017000      I2 = 2
00056 01018000      I3 = 3
00060 01019000  C
00060 01020000  C INIT LABELS BUFFER
00060 01021000      DO 10 I = 1,NX
00065 01022000      DO 10 J = 1,3
00072 01023000      10 LABBUF(I,J) = 1
00103 01024000  C
00103 01025000  C READ 3 LINES DATA
00103 01026000      DO 20 J = 1,3
00110 01027000      DO 20 K = 1,ND
00115 01028000      CALL READP(UICB,IND,INGIN,DATBUF(1,K,J),2,K,J,NU,NX,K+1,
00115 01029000      * J,NU,NX)
00150 01030000      IF (INC(1).LT.0) CALL CHKBU(UICB,IND,INGIN,NU,NX,K,1019)
00170 01031000      20 CONTINUE
00172 01032000  C
00172 01033000  C MARK BOUNDARY 1-3 LR
00172 01034000      DO 30 J = 1,3
00177 01035000      30 CALL MARKLR(Z,DATBUF(1,1,J),INTTHR,NC,ND,LABBUF(1,J),NASK,NX)
00224 01036000  C
00224 01037000  C MARK BOUNDARY 2 UD
00224 01038000      CALL MARKUPD(Z,DATBUF(1,1,1),DATBUF(1,1,2),LABBUF(1,1),
00224 01039000      * LABBUF(1,2),LABBUF(1,3),NC,ND,INTTHR,NASK,NX)
00264 01040000  C
00264 01041000  C FILL IN CRACKS 2
00264 01042000      CALL FILLR(Z,LABBUF(1,2),NX)
00273 01043000  C
00273 01044000  C DUP LABELS 2 TO 1
00273 01045000      DO 70 I = 1,NX
00300 01046000      70 IF (LABBUF(1,2).EQ.Z) LABBUF(1,1) = Z
00316 01047000      CALL MKIVL(Z,LABBUF(1,1),NX,IN)
00326 01048000      DO 222 I = 1,NX
00333 01049000      IF (LABBUF(1,1) NE Z) LABBUF(1,1)=LABBUF(1,1)+Z
00353 01050000      222 CONTINUE
00356 01051000      LAB = LAB+IN

```

PRECEDING PAGE BLANK NOT FILMED

PAGE 0033 START

```

00365 01052000 1000 CONTINUE
00365 01053000 C
00365 01054000 C MARK INTERVALS I2
00365 01055000 CALL MARKVL(Z,LABBUF(I,I2),NX,IN)
00376 01056000 C
00376 01057000 C SEE IF ANYTHING TO CONNECT
00376 01058000 IF (LAB.NE.Z) GO TO 81
00402 01059000 LAB = Z+IN
00403 01060000 DO 1111 I = 1,NX
00412 01061000 IF (LABBUF(I,I1).NE.Z) LABBUF(I,I1) = LABBUF(I,I1)+Z
00427 01062000 1111 CONTINUE
00440 01063000 GO TO 82
00441 01064000 81 IF (IN.EQ.0) GO TO 82
00445 01065000 CALL CONNCT(FINISHED,Z,NX,LABBUF(I,I1),LABBUF(I,I2),ISCAN,IN,LAB)
00467 01066000 C
00467 01067000 C SEE IF FINISHED
00467 01068000 IF (FINISHED) GO TO 2000
00473 01069000 C
00473 01070000 C WRITE I1 LABELS
00473 01071000 82 IROUOUT = IROUOUT+1
00474 01072000 CALL WRITEP(UICB,IND,INGCLAS,LABBUF(I,I1),Z,I,IROUOUT,NU,NX,
00474 01073000 * 1,IROUOUT+1,NU,NX)
00526 01074000 IF (IND(I).LT.0) CALL CHKIO(UICB,IND,INGCLAS,I1,NU,0,20)
00547 01075000 C
00547 01076000 C READ NEXT
00547 01077000 IROU = IROU+1
00550 01078000 IF (IROU.GT.NR) GO TO 300
00554 01079000 DO 85 K = 1,ND
00561 01080000 CALL READP(UICB,IND,INGIN,DATBUF(I,K,I1),Z,K,IROU,NU,NX,
00561 01081000 * K+1,IROU,NU,NX)
00614 01082000 IF (IND(I).LT.0) CALL CHKIO(UICB,IND,INGIN,K,I1,NU,30)
00634 01083000 85 CONTINUE
00635 01084000 C
00635 01085000 C MARK NEW LABELS
00635 01086000 DO 200 I = 1,NX
00642 01087000 200 LABBUF(I,I1) = 1
00652 01088000 C
00652 01089000 C ROTATE BUFFERS
00652 01090000 IT = I1
00654 01091000 I1 = I2
00656 01092000 I2 = I3
00660 01093000 I3 = IT
00662 01094000 C
00662 01095000 C MARK
00662 01096000 CALL MARKLR(Z,DATBUF(I,I,13),INTTHR,NC,NU,LABBUF(I,I3),MASK,NX)
00706 01097000 CALL MARKUPDN(Z,DATBUF(I,I,11),DATBUF(I,I,12),LABBUF(I,I1),
00706 01098000 * LABBUF(I,I2)-LABBUF(I,I3),NC,ND,INTTHR,MASK,NX)
00737 01099000 C
00737 01100000 C DO IT AGAIN
00737 01101000 GO TO 1000
00737 01102000 C
00737 01103000 C FINISHED...PROCESS LAST TWO SCAN LINES
00737 01104000 300 CONTINUE
00737 01105000 C
00737 01106000 C PROPGATE I2 TO I3
00737 01107000 DO 310 I = 1,NX
00762 01108000 IF (LABBUF(I,I2).EQ.Z) LABBUF(I,I3) = Z

```

ORIGINAL PAGE IS  
OF POOR QUALITY

PAGE 0034 START

```

01001 01109000 310 CONTINUE
01002 01110000 C
01002 01111000 C SCAN LINE I2 THEN I3
01002 01112000 DO 410 ITH = 1,2
01007 01113000 CALL HRRIVL(Z,LABBUF(1,I2),NX,IN)
01020 01114000 C
01020 01115000 C JOIN I2 TO I1
01020 01116000 IF (IN.NE 0) CALL CONNCT(FINISHED,Z,NX,LABBUF(1,I1),LABBUF(1,I2),
01020 01117000 * ISCAN,IN,LAB)
01045 01118000 IF (FINISHED) GO TO 2000
01051 01119000 C
01051 01120000 C STORE I1
01051 01121000 IROUOUT = IROUOUT+1
01052 01122000 CALL WRITEP(UICD,IND,INGCLAS,LABBUF(1,I1),2,1,IROUOUT,NX,NX,
01052 01123000 * 1,IROUOUT+1,NX,NX)
01104 01124000 IF (IND(1).LT.0) CALL CNKIC(UICD,IND,INGCLAS,I1,NX,0,40)
01125 01125000 I1 = I2
01127 01126000 I2 = I3
01131 01127000 C
01131 01128000 C DO IT AGAIN (TWICE)
01131 01129000 410 CONTINUE
01132 01130000 96 CONTINUE
01136 01131000 R E T U R N
01137 01132000 2000 DO 2020 J = 1,NX
01144 01133000 2020 LABBUF(J,1) = Z
01153 01134000 2010 IROUOUT = IROUOUT+1
01154 01135000 DO 2040 I = IROUOUT,NR
01161 01136000 CALL WRITEP(UICD,IND,INGCLAS,LABBUF(1,I),2,1,I,NX,NX,
01161 01137000 * 1,I+1,NX,NX)
01212 01138000 IF (IND(1).LT.0) CALL CNKIC(UICD,IND,INGCLAS,I,NX,Z,750)
01232 01139000 2040 CONTINUE
01233 01140000 C
01233 01141000 C SEE IF FINISHED
01233 01142000 IF (NY.GE.NZ) R E T U R N
01240 01143000 NY = NY+1
01243 01144000 NY = NY+NC
01246 01145000 IF (NY.GT.NZ) NY = NZ
01253 01146000 NX = NY-NV+1
01257 01147000 IROUOUT = 0
01261 01148000 GO TO 2010
01262 01149000 END

```

## SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
CNKIC		SUBROUTINE		CONNCT		SUBROUTINE	
DATBUF	INTEGER	ARRAY	0-214 ,I	FILLR		SUBROUTINE	
FINISHED	LOGICAL	SIMPLE VAR	0-211	I	INTEGER	SIMPLE VAR	0-23
I1	INTEGER	SIMPLE VAR	0-216	I2	INTEGER	SIMPLE VAR	0-217
I3	INTEGER	SIMPLE VAR	0-220	INGCLAS	INTEGER	SIMPLE VAR	0-26 ,I
INGIN	INTEGER	SIMPLE VAR	0-27 ,I	IN	INTEGER	SIMPLE VAR	0-27
IND	INTEGER	ARRAY	0-210 ,I	INTNR	INTEGER	ARRAY	0-221 ,I
IROU	INTEGER	SIMPLE VAR	0-26	IROUOUT	INTEGER	SIMPLE VAR	0-24
ISCAN	INTEGER	ARRAY	0-212 ,I	IT	INTEGER	SIMPLE VAR	0-212
ITH	INTEGER	SIMPLE VAR	0-25	J	INTEGER	SIMPLE VAR	0-210
K	INTEGER	SIMPLE VAR	0-221	LAB	INTEGER	SIMPLE VAR	0-29 ,I
LABBUF	INTEGER	ARRAY	0-213 ,I	LABR		SUBROUTINE	
LABR		SUBROUTINE		LABR		SUBROUTINE	
LABRUPON		SUBROUTINE		LABR	LOGICAL	SIMPLE VAR	0-24 ,I
LABRIVL		SUBROUTINE		NC	INTEGER	SIMPLE VAR	0-216 ,I
NR	INTEGER	SIMPLE VAR	0-220 ,I	NR	INTEGER	SIMPLE VAR	0-217 ,I
NX	INTEGER	SIMPLE VAR	0-213	NX	INTEGER	SIMPLE VAR	0-214
NY	INTEGER	SIMPLE VAR	0-215	NZ	INTEGER	SIMPLE VAR	0-215 ,I
REASP		SUBROUTINE		START		SUBROUTINE	
UICD	INTEGER	ARRAY	0-211 ,I	WRITEP		SUBROUTINE	
Z	INTEGER	SIMPLE VAR	0-222				

PROGRAM UNIT START COMPILED

Parent: MAIN

THINTSTM

Calls: GETN25

THINTSTM(MP,TSP,TTP,CLASS,COUNT,N25,N60,N288,N140,N388,N428,ND,NTSI,FILENO,  
UICB,IND)

Subroutine ASELECT creates a disk file of samples taken 5 at a time from the patches (the components of the complement of the bouncary). There will be many of these sets in a large image. The starting conditions for the clustering part of AMOEBA require relatively few. Further, the starting cluster centers have not been formed yet. Subroutine THINTSTM forms the starting cluster centers and "thins" the test sets.

Method: In MAIN, program variables N288, N388, and N428 are selected as large as possible depending on available memory (i.e. depending on ND). (If ND is 4, then N288 = 288, N388 = 388, and N428 = 428). If we start with NTS test sets (stored on disk by ASELECT), and if  $NTS < N428$ , then THINTSTM simply forms NTS means as starting cluster centers and returns  $N140 = NTS$  and  $N488 = NTS$ . Otherwise there are many test sets, and a complex procedure is followed to prevent the number of means and the number of test sets from growing too large.

1. If  $NTS < N25$ , go to step 7 (the finish). Otherwise read 25 test sets into the temporary buffer TTP.
2. Each test set has 5 test points; classify the 25 first elements in the class of nearest last element centers.
3. For each of the test sets classified correctly (i.e. in which the first element was nearer its last than the last of another of the 25), form the mean and add this mean to the mean pool MP, indicate the main pool is occupied with COUNT, and add the test set to the test set pool TSP, indicating it is occupied by CLASS. Count these events in NMP (the means) and NTSP (the test sets).
4. If there are more than 100 vectors in the mean pool, proceed to step 5. If there are more than N388 test sets, proceed to step 6. Otherwise go to step 1 and get more.

5. More than 100 vectors are in the mean pool. Classify all center test pixels in the classes of the mean pool and count the number of times a mean is hit. Eliminate each mean which was not hit. If NMP is greater than N60 ( $=60$ ) after this operation, eliminate each with 1, 2, ... classification until  $NMP \leq N60$ . At each elimination, reclassify center test pixels which were assigned to an eliminated mean and test if  $NMP < N60$ . When N60 is reached, test whether NTSP is greater than N388. If  $NTSP \leq N388$ , return to step 1 to gather more, else proceed with step 6.
6. More than N388 test sets are in the test set pool, but fewer than 100 means are in the mean pool. Moreover, the majority of the test sets have been assigned to a class by the logic of step 5. (The ones just added haven't been, of course, so they are kept around a while longer.) Determine the mean which has the most test sets assigned. Test sets assigned to this mean are duplicates. Eliminate one (the first one found) and decrease the count. Repeat until fewer than N288 test sets are present, then go to step 1.
7. Close the disk file which contained the test sets, and compress and count the means and test sets; N140 is the number of means (starting cluster centers) and N428 is returned the number of test sets.

#### Program Variables

CLASS(N428)	INTEGER ARRAY	The class to which a center test pixel is assigned.
COUNT(N140)	INTEGER ARRAY	The number of center test pixels assigned to a mean.
FCLOSE	SYSTEM SUBROUTINE	
FILENO	INTEGER	File number of scratch disk file containing test sets.
FIRST	LOGICAL	Switch used to jump around rewind of file in GETN25.

GETN25           SUBROUTINE   Reads N25 test sets from disk.

I,II,J,K,L,LS,M    INTEGER   DO loop index.

IA                INTEGER   Class of nearest last test set element to first  
                  in batch of N25.

ICL               INTEGER   Class of nearest mean to test set center.

IE                INTEGER   Count of how unpopular a mean should be to be  
                  eliminated.

IM                INTEGER   Distance to mean of a test set center.

IND(1)            INTEGER ARRAY   Error indicator.

IS                INTEGER   Distance accumulator.

IT                INTEGER   Used to convert logical to integer. Also  
                  distance temporary.

IT1               INTEGER   Distance temporary.

JS                INTEGER   The test set center reclassified after a mean  
                  is eliminated.

JT                INTEGER   A classification, tested to see if a duplicate  
                  test set has been found.

LIT               LOGICAL   Equivalenced to IT; used to convert logical to  
                  integer to compensate for the inadequacy of the Segmenter.

MAX               INTEGER   The running maximum count of classification  
                  in all means.

MP(ND,N140)       INTEGER ARRAY   The mean pool, and, on return, the means.

N140              INTEGER   On call, 140; on return, the number starting  
                  clusters.

N25               INTEGER   The number of test sets read at a time.

N288              INTEGER   Number of test sets sought after the iteration  
                  is started.

N388              INTEGER   Test set iteration trigger.

N428            INTEGER   Number of slots for test sets.

N60            INTEGER   Means sought after iteration started.

ND             INTEGER   Dimensionality.

ND5            INTEGER   ND\*5, for GETN25.

NDXM           INTEGER   Pointer to mean pool, used in search for a  
free slot.

NDXP           INTEGER   Pointer to test set pool.

NEAR           INTEGER   Used in test set to test set smallest distance  
determination.

NMP            INTEGER   Number in mean pool.

NTS            INTEGER   Number of test sets on disk remaining.

NTSI           INTEGER   Number of test sets on disk.

NTSP           INTEGER   Number in test set pool.

TSP(ND,5,N488)   INTEGER ARRAY   Test set pool.

TTP(ND,5,N25)   LOGICAL ARRAY   Test sets for GETN25 to read into.

UICE(1)        INTEGER ARRAY   User Information Control Block.

PAGE 0041 HEULETT-PACKARD 321020 01.03 FORTRAN/3000 TUE, OCT 13, 1981, 9:39 AM

```

00015 01260000 %CONTROL SEGMENT=ANCDBASEC
00015 01261000     SUBROUTINE TRINTSTK(RP, TSP, TYP, CLASS, COUNT, N25, N60,
00015 01262000     * N200, N140, N300, N420, ND, N151, FILENO, UIC(7), IND)
00015 01263000     SYSTEM INTRINSIC FCLOSE
00015 01264000 C SUBROUTINE TO REDUCE A LARGE NUMBER OF TEST SETS
00015 01265000 C TO ABOUT N300, AND TO FORM ABOUT N200 NEAR VECTORS
00015 01266000 C TO START NUMCLU
00015 01267000 C
00015 01268000 C PARAMETERS:
00015 01269000 C
00015 01270000 C     RP -- NEAR POOL
00015 01271000 C     TSP -- TEST SET POOL
00015 01272000 C     TYP -- TEMPORARY TEST SET POOL
00015 01273000 C     CLASS -- CLASSIFICATION OF TSP IN RP
00015 01274000 C     COUNT -- COUNT OF RP CLASSIFICATIONS
00015 01275000 C     ND -- DIMENSIONALITY
00015 01276000 C     NTSI -- NUMBER OF TEST SETS INPUT
00015 01277000 C     NTS -- NUMBER OF TEST SETS
00015 01278000 C     NTSF -- NUMBER IN TEST SET POOL
00015 01279000 C     NRP -- NUMBER IN NEAR POOL
00015 01280000 C     N25 -- NUMBER OF TEST SETS GRABBED AT A TIME
00015 01281000 C     N60 -- NEARS BOUGHT AFTER ITERATION STARTED
00015 01282000 C     N200 -- NEAR ITERATION TRIGGER, AND
00015 01283000 C     -- TEST SETS BOUGHT AFTER ITERATION STARTED
00015 01284000 C     N140 -- ABSOLUTE MAX NUMBER NEARS
00015 01285000 C     ON RETURN, THIS PARAMETER IS THE ACTUAL NUMBER
00015 01286000 C     N300 -- TEST SET TRIGGER
00015 01287000 C     N420 -- ABSOLUTE MAX NUMBER OF TEST SETS
00015 01288000 C     ON RETURN, THIS PARAMETER IS THE ACTUAL NUMBER
00015 01289000 C     ND*H -- SEARCH INDEX IN RP LOOKING FOR NEU
00015 01290000 C     ND*P -- DITTO FOR TSP
00015 01291000 C
00015 01292000     LOGICAL LIT, TYP(ND,5,N25), FIRST
00015 01293000     EQUIVALENCE (LIT, LIT)
00015 01294000     INTEGER*2 RP(ND, N140), TSP(ND,5, N420),
00015 01295000     * CLASS(N420), COUNT(N140), FILENO, UIC(1), IND(1)
00015 01296000 C INITIALIZE
00015 01297000     FIRST = TRUE
00015 01298000     ICL = 1
00015 01299000     JS = 1
00015 01300000     DO 10 I = 1, N140
00015 01301000     10 COUNT(I) = -1
00015 01302000     DO 20 J = 1, N420
00015 01303000     20 CLASS(J) = -1
00015 01304000     NRP = 0
00015 01305000     NTSF = 0
00015 01306000     NTS = NTSI
00015 01307000     ND*H = 0
00015 01308000     ND*P = 0
00015 01309000     NDS = ND*5
00015 01310000 C
00015 01311000 C     INITIALLY NEAR POOL AND TEST SET POOL ARE EMPTY
00015 01312000 C
00015 01313000 C     REFERENCE POINT FOR NEXT N25 TEST SETS
00015 01314000 C     GET N25 TEST SETS
00015 01315000 C     (RETURN IF NOT THAT MANY)
00015 01316000 C     IF (NTR GE N425) GO TO 2000

```

PAGE 0042 THIRTYVM

```

00066 01317000      N25 = NTS
00070 01318000      N140 = NTS
00072 01319000      N428 = NTS
00074 01320000      CALL GETN25(FIRST, TTP, N25, N05, FILENO, UICB, IND)
00107 01321000      CALL FCLOSE(FILENO, 4, 0)
00112 01322000      DO 1 I = 1, NTS
00120 01323000      DO 2 K = 1, N0
00122 01324000      IS = 2
00127 01325000      DO 3 LS = 1, 5
00134 01326000      LIT = TTP(K, LS, I)
00144 01327000      IS = IS+IT
00151 01328000      3 TSPCK(LS, I) = IT
00164 01329000      2 NPCK(I) = IS/5
00173 01330000      1 CONTINUE
00174 01331000      R E T U R N
00177 01332000      2000 IF (NTS LT N25) GO TO 6000
00204 01333000      CALL GETN25(FIRST, TTP, N25, N05, FILENO, UICB, IND)
00217 01334000      NTS = NTS-N25
00222 01335000      C
00222 01336000      C CLASSIFY FIRST PIXEL IN EACH TEST SET IN NEAREST
00222 01337000      C LAST PIXEL CLASS
00222 01338000      DO 30 L = 1, N25
00227 01339000      NEAR = 16000
00231 01340000      C
00231 01341000      C FIND CLOSEST TO L-TH CRITER
00231 01342000      DO 40 M = 1, N25
00236 01343000      IS = -32768
00243 01344000      DO 50 K = 1, N0
00252 01345000      IF (IS GT NEAR) GO TO 40
00262 01346000      LIT = TTP(K, I, L)
00273 01347000      IT1 = IT
00275 01348000      LIT = TTP(K, S, M)
00306 01349000      IT = IT-IT1
00311 01350000      50 IS = IS+IT*IT
00316 01351000      IF (IS GE NEAR) GO TO 40
00322 01352000      IA = M
00324 01353000      NEAR = IS
00326 01354000      40 CONTINUE
00327 01355000      C
00327 01356000      C ZAP ANY SUCH THAT CLASS(K)>K
00327 01357000      IF (IA NE L) GO TO 30
00332 01358000      C
00332 01359000      C FIND A SLOT FOR NEAR AND TEST PIXELS
00335 01360000      70 NDXM = NDXM+1
00334 01361000      IF (NDXM GT N140) NDXM = 1
00341 01362000      IF (COUNT(NDXM) GE 0) GO TO 70
00346 01363000      90 NDXP = NDXP+1
00347 01364000      IF (NDXP GT N428) NDXP = 1
00354 01365000      IF (CLASS(NDXP) GE 0) GO TO 80
00361 01366000      C
00361 01367000      C FORM HEANS, ADD TO NEAR POOL
00361 01368000      C ADD TEST SETS TO TEST SET POOL
00361 01369000      DO 90 A = 1, N0
00366 01370000      IS = 2
00370 01371000      DO 100 LS = 1, 5
00373 01372000      LIT = TTP(K, LS, L)
00407 01373000      IS = IS+IT

```

PAGE 0043 THIRTYTH

```

00412 01374000 100 TSP(K,LS,NDXP) = IT
00423 01375000 90 NP(K,NDXR) = IS/3
00436 01376000 C
00436 01377000 C SHOW COVERED AND COUNT
00436 01378000 COUNT(NDXR) = 0
00441 01379000 CLASS(NDXP) = 0
00444 01380000 NRP = NRP+1
00445 01381000 NTSP = NTSP+1
00446 01382000 30 CONTINUE
00447 01383000 C
00447 01384000 C DOES THE NEAR POOL CONTAIN MORE THAN 100 NEARS?
00447 01385000 IF (NRP.GE.100) GO TO 1000
00453 01386000 C*** YES: GO ELIMINATE (1000)
00453 01387000 C
00453 01388000 C*** NO:
00453 01389000 C ARE THERE MORE THAN N200 TEST SETS?
00453 01390000 3000 IF (NTSP.LT.N200) GO TO 2000
00457 01391000 C*** NO: GO ITERATE (2000)
00457 01392000 C
00457 01393000 C*** YES: ZAP SOME
00457 01394000 MAX = -1
00461 01395000 DO 200 I = 1,N140
00466 01396000 MAX = MAX(MAX,COUNT(I))
00475 01397000 200 CONTINUE
00476 01398000 3000 DO 210 J = 1,N429
00503 01399000 JT = CLASS(J)
00506 01400000 IF (JT.LE.0) GO TO 210
00512 01401000 IF (COUNT(JT).LT.MAX) GO TO 210
00517 01402000 CLASS(J) = -1
00522 01403000 COUNT(JT) = COUNT(JT)-1
00526 01404000 NTSP = NTSP-1
00527 01405000 IF (NTSP.LT.N200) GO TO 2000
00529 01406000 210 CONTINUE
00539 01407000 MAX = MAX-1
00536 01408000 GO TO 3000
00537 01409000 C CLASSIFY ALL CENTER OF TEST SETS AND COUNT
00537 01410000 C
00537 01411000 1000 NP = 3
00541 01412000 DO 399 J = 1,N140
00546 01413000 IF (COUNT(J).GE.0)COUNT(J) = 0
00555 01414000 399 CONTINUE
00556 01415000 C NOW CLASSIFY EM
00556 01416000 DO 300 J = 1,N428
00563 01417000 IF (CLASS(J).LT.0) GO TO 300
00570 01418000 IN = 16000
00572 01419000 DO 310 I = 1,N140
00577 01420000 IF (COUNT(I).LT.0) GO TO 310
00604 01421000 IS = -32760
00613 01422000 DO 320 K = 1,N0
00620 01423000 IF (IS.GE.IN) GO TO 310
00625 01424000 IT = TSP(K,NP,J)-NP(K,I)
00645 01425000 320 IS = IS+IT+IT
00652 01426000 IF (IS.GT.IN) GO TO 310
00656 01427000 IN = IS
00660 01428000 ICL = I
00662 01429000 310 CONTINUE
00663 01430000 CLASS(J) = ICL

```

PAGE 0044 THIRTYTH

```

00666 01431000      COUNT(ICL) = COUNT(ICL)+1
00672 01432000      300 CONTINUE
00673 01433000      C
00673 01434000      C ALL CLASSIFIED AND COUNTED. NOW SEE IF ANY MISSES
00673 01435000      DO 330 I = 1,N140
00700 01436000      IF (COUNT(I).NE.0) GO TO 330
00709 01437000      COUNT(I) = -1
00710 01438000      NRP = NRP-1
00711 01439000      330 CONTINUE
00712 01440000      C
00712 01441000      C SEE IF WE'RE DOWN TO N60
00712 01442000      IF (NRP.LE.N60) GO TO 3000
00716 01443000      C NO. SO ZAP SOME WITH 1,2,...CLASSIFICATIONS
00716 01444000      C
00716 01445000      IE = 1
00720 01446000      400 DO 440 II = 1,N140
00723 01447000      IC = COUNT(II)
00739 01448000      IF (IC.NE.IE) GO TO 440
00734 01449000      NRP = NRP-1
00737 01450000      COUNT(II) = -1
00740 01451000      C REMOVED. NOW RECLASSIFY ITEMS ASSIGNED TO CLASS I
00740 01452000      C
00740 01453000      DO 450 J = 1,N428
00742 01454000      IF (CLASS(J).NE.IE) GO TO 450
00751 01455000      IN = 16000
00754 01456000      DO 460 I = 1,N140
00761 01457000      IF (COUNT(I).LT.0) GO TO 460
00767 01458000      IS = -32768
00778 01459000      DO 470 K = 1,N0
01003 01460000      IF (IS.GE.IN) GO TO 460
01012 01461000      IT = TSP(K,3,J)-NPK(I)
01031 01462000      470 IS = IS+IT
01036 01463000      IF (IS.GT.IN) GO TO 460
01042 01464000      JS = J
01044 01465000      IN = IS
01046 01466000      ICL = I
01070 01467000      460 CONTINUE
01091 01468000      CLASS(JS) = ICL
01094 01469000      COUNT(ICL) = COUNT(ICL)+1
01080 01470000      450 CONTINUE
01081 01471000      IF (NRP.LT.N60) GO TO 3000
01087 01472000      440 CONTINUE
01070 01473000      IE = IE+1
01071 01474000      GO TO 4000
01072 01475000      6000 CALL FCLOSE(FILENO,4,0)
01076 01476000      L = 0
01100 01477000      C NOW PUT EVERYTHING IN THE FIRST SLOTS OF NP AND TSP
01100 01478000      C
01100 01479000      DO 7000 I = 1,N428
01109 01480000      IF (CLASS(I).EQ.-1) GO TO 7000
01112 01481000      L = L+1
01113 01482000      DO 7010 J = 1,N0
01120 01483000      DO 7010 J = 1,N3
01123 01484000      7010 TSP(K,J,L) = TSP(K,J,I)
01131 01485000      7000 CONTINUE
01132 01486000      N425 = L
01134 01487000      L = 0

```

PAGE 0045 THINTSTR

```

01174 01488000      DO 7020 I = 1,N100
01175 01489000      IF (COUNT(I).EQ.-1) GO TO 7020
01176 01490000      L = L+1
01177 01491000      DO 7030 K = 1,ND
01178 01492000 7020 RPER(L) = RPA(L,I)
01217 01493000 7020 CONTINUE
01218 01494000      N100 = L
01219 01495000      R E T U R N
01217 01496000      ENB

```

## SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
CLASS	INTEGER	ARRAY	0-120 .I	COUNT	INTEGER	ARRAY	0-217 .I
PCLOSE		SUBROUTINE		FILENO	INTEGER	SIMPLE VAR	0-26 .I
FIPST	LOGICAL	SIMPLE VAR	0-136	GETN23		SUBROUTINE	
I	INTEGER	SIMPLE VAR	0-25	IA	INTEGER	SIMPLE VAR	0-27
IC	INTEGER	SIMPLE VAR	0-211	ICL	INTEGER	SIMPLE VAR	0-231
IE	INTEGER	SIMPLE VAR	0-213	II	INTEGER	SIMPLE VAR	0-218
IP	INTEGER	SIMPLE VAR	0-221	IND	INTEGER	ARRAY	0-24 .I
IS	INTEGER	SIMPLE VAR	0-223	IT	INTEGER	SIMPLE VAR	0-230
ITL	INTEGER	SIMPLE VAR	0-232	J	INTEGER	SIMPLE VAR	0-222
JS	INTEGER	SIMPLE VAR	0-216	JT	INTEGER	SIMPLE VAR	0-217
K	INTEGER	SIMPLE VAR	0-233	L	INTEGER	SIMPLE VAR	0-214
LIT	LOGICAL	SIMPLE VAR	0-230	LS	INTEGER	SIMPLE VAR	0-237
R	INTEGER	SIMPLE VAR	0-227	MAX	INTEGER	SIMPLE VAR	0-226
RP	INTEGER	ARRAY	0-223 .I	N100	INTEGER	SIMPLE VAR	0-213 .I
N23	INTEGER	SIMPLE VAR	0-216 .I	N200	INTEGER	SIMPLE VAR	0-214 .I
N300	INTEGER	SIMPLE VAR	0-212 .I	N420	INTEGER	SIMPLE VAR	0-211 .I
N69	INTEGER	SIMPLE VAR	0-219 .I	ND	INTEGER	SIMPLE VAR	0-210 .I
NDS	INTEGER	SIMPLE VAR	0-224	NDXN	INTEGER	SIMPLE VAR	0-26
NDXP	INTEGER	SIMPLE VAR	0-210	NEAR	INTEGER	SIMPLE VAR	0-228
NPP	INTEGER	SIMPLE VAR	0-212	NP	INTEGER	SIMPLE VAR	0-225
NTS	INTEGER	SIMPLE VAR	0-220	NTSI	INTEGER	SIMPLE VAR	0-27 .I
NTSP	INTEGER	SIMPLE VAR	0-234	THINTSTR		SUBROUTINE	
TSP	INTEGER	ARRAY	0-222 .I	TTP	LOGICAL	ARRAY	0-221 .I
UIC0	INTEGER	ARRAY	0-25 .I				

PROGRAM UNIT THINTSTR COMPILED

Parent: MAIN

THRFD

THRFD(NFL,INTTHR,SCANLINE,UICB,IND,KOUNT,NR,NC,NB,MASK,IMGIN)

Subroutine THRFD finds vector thresholds INTTHR so that about NFL percent of the scene is in patches. The thresholds are used in subroutine START to decide boundary; a boundary decision is made when any channel differs from its neighbor by more than that channel's INTTHR.

Method: Initially the data is sparsely sampled to estimate the variability. The initial thresholds are, for each channel K,  $.15 \times (NFL+5) \times A(K)/\sqrt{ND}$ , where  $A(K)$  is the average estimated in that channel of the difference of a point from its left-hand neighbor.

These initial thresholds are updated in an adaptive program which scans the data checking when a boundary decision would be made. A count is kept of the decisions non boundary (i.e. pure). When too few pure points are being found, the thresholds are increased (making it harder to be a boundary point and thus easier to be a pure point). When too many are found, the thresholds are decreased. Too many or few is decided by comparing the count NFND with the target TARGET.

Only one threshold is increased or decreased at a time. A count is kept of the number of boundary decisions which have been made per channel. When it is necessary to decrease the thresholds, that threshold which has the highest count is decreased. Dually, when the thresholds must be increased, that threshold with the lowest count is incremented. The effect is that the algorithm expects all channels to contribute equally in boundary-finding.

Flags UP and DOWN, initially .FALSE., control exit. When the thresholds are increased, UP is set .TRUE. When they are decreased, DOWN is set .TRUE. Another flag, BOTH (initially false) is tested after each outer loop (the data is scanned in a pattern which spreads the sparse sample), and then, after the test is set to UP.AND.DOWN. Thus one more scan of the data is taken after both flags have been set.

The scanning strategy has two phases: the initial phase sets NCS = the larger of 4 or  $NR*NC/557039$ . Then data is sampled by every eleventh line and every NCS sample along a line to estimate variability. When the sum exceeds 32245 in any channel or when the loop falls through, the initial INTTHR estimates are calculated and the second sampline phase is entered. Let

$$NSEL = (NR/100)*(NC/100)/2 + 1$$

$$N20 = 20 ; \text{ if } N20 \geq NC, N20 = NC/2 .$$

Then the loop structure is the following (in FORTRAN):

```

DO 665 NN = 2, N20, NSEL
  DO 65 N = 1,20,5
    IS = N+NN
    IF(IS.GE.NR)IS = NR/2
      DO 100 I + IS, NR, 17
        read data for line I
        DO 60 J = NN, NC, 5
          count pure decisions and boundary
          decisions per channel.
60          CONTINUE
          see if too many (→80), just right (→100)
          or too few (here) (increase and go to 63)
80          too many : decrease
63          reset counters and target
100         CONTINUE
65         CONTINUE
            test BOTH ; if .TRUE., RETURN
            set BOTH to UP.AND.DOWN
665        CONTINUE
            RETURN

```

As can be seen, the sample is not sparse: in particular, the 65 loop must be executed at least twice, first with  $NN = 2$  and then  $NN = 2+NSEL$  (in a  $512 \times 512$  image,  $NSEL = 13$ ). Thus, the rows sampled are (with  $NSEL = 13$ )

3	,	20	,	37	,	54	,	71	,	88	,	...
8	,	25	,	42	,	59	,	76	,	93	,	...
13	,	30	,	47	,	64	,	81	,	98	,	...
18	,	35	,	52	,	69	,	86	,	103	,	...
16	,	33	,	50	,	67	,	84	,	101	,	...
21	,	38	,	55	,	72	,	89	,	106	,	...
26	,	43	,	60	,	77	,	94	,	111	,	...
31	,	48	,	65	,	82	,	99	,	116	,	...

During the first four, every fifth line sample pair is sampled beginning at sample 2. During the second four, the starting sample is 15. This sampling strategy avoids staying in any place too long and is fast.

#### Program Variables

BOTH	LOGICAL	A flag which, when true, means the thresholds have been increased and decreased.
CHKIO	SYSTEM SUBROUTINE	
DELTGT	REAL	$(NFL+5)/100$ , used to increment TARGET when a test is made.
DOWN	LOGICAL	A flag used to tell when the thresholds have been decreased.
FLAG	LOGICAL	Used to tell when a pure point has been detected.
I,J,K,N,NN	INTEGER	DO loop index.
IMGIN	INTEGER	Input image number.
IND(1)	INTEGER ARRAY	Error indicator.
INTTHR(ND)	INTEGER ARRAY	The integer thresholds.
IS	INTEGER	Starting row in loop 100.
IT	INTEGER	Used to accumulate initial estimate of variability.
JM	INTEGER	J-1; points to sample to left along a line.
KI	INTEGER	Index of threshold to be increased or decreased.

KOUNT(ND) INTEGER ARRAY Used to count the number of times a boundary decision would have been made in a channel if all were tested.

MASK LOGICAL If .TRUE., a value of 0 in channel 1 is regarded as a mask (i.e. not image data).

MAX,MIN INTEGER Used to find max or min of KOUNT to determine which channel threshold to adjust.

N20 INTEGER Loop 665 parameter: usually 20.

NC INTEGER Number of samples.

NCS INTEGER  $\text{MAX}(NR*NC/557039,4)$  used to sparsely sample on the initial estimate.

ND INTEGER Dimensionality.

NFL INTEGER Input parameter: percent of scene which user believes to be inside patches.

NFLD INTEGER  $\text{NFL}+5$ ; interval parameter which allows for "crack" fill in logic in start which adds boundary points not based on thresholding. This has been found to amount to about 5 percent.

NFND INTEGER Running number of pure points found; tested against TARGET to decide if too many, too few, or about right during a pass through the data.

NR INTEGER Number of lines.

NSEL INTEGER  $(NR/100)*(NC/100)/2+1$ ; used as loop 665 parameter.

NUM INTEGER Counter during variability estimation phase.

OONUM REAL Used to form initial threshold estimates.

READP SYSTEM SUBROUTINE

SCANLINE(NC,ND) INTEGER ARRAY One line of data.

TARGET REAL Running count of the target percent pure points.

UICB(1)

INTEGER ARRAY User Information Control Block

UP

LOGICAL A flag which is set when the thresholds have been increased.

PAGE 0047 HEULETT-PACKARD 32102B.01.03 FORTRAN/3000 TUE, OCT 12, 1981, 9:40 AM

```

00012 01531000 SCNTROL SECMNT=ANOEASEC
00013 01532000 SUBROUTINE THRFND(NFL,INTTHR,SCANLINE,UICB,IND,KOUNT,HR,NC,ND,
00015 01533000 * MASK,INGIN)
00015 01534000 INTEGER*2 SCANLINE(NC,ND),INTTHR(ND),KOUNT(ND),UICB(1),IND(4),
00015 01535000 * BUFFER(NC)
00015 01536000 LOGICAL MASK,FLAG,UP,DOWN,BOTH
00017 01537000 NSEL = (NR/100)*(NC/100)/2+1
00023 01538000 NCS = IFIX(FLOAT(NR)*FLOAT(NC)/32767./17.)
00041 01539000 IF (NCS.LT.5) NCS = 4
00046 01540000 NUM = 1
00050 01541000 NFLD = MAX(MIN(NFL+5,60),25)
00063 01542000 NFND = 0
00069 01543000 TARGET = NFLD*.01
00071 01544000 DELTGT = TARGET
00072 01545000 DO 20 K = 1,ND
00100 01546000 20 INTTHR(K) = 0
00104 01547000 IR = 1
00106 01548000 UP = .FALSE.
00110 01549000 DOWN = UP
00112 01550000 BOTH = UP
00114 01551000 C
00114 01552000 C SAMPLE DATA TO GET INITIAL ESTIMATE
00114 01553000 C
00114 01554000 DO 30 I = 2,NR,11
00121 01555000 C
00121 01556000 C EVERY 11 SCAN LINES
00121 01557000 C
00121 01558000 N20 = 20
00122 01559000 IF (N20.GE.NC) N20 = NC/2
00131 01560000 DO 25 K = 1,ND
00136 01561000 CALL READP(UICB,IND,INGIN,SCANLINE(I,K),2,K,I,1,NC,K+1,I,1,NC)
00179 01562000 IF (IND(I).LT.0) CALL CHKID(UICB,IND,INGIN,IR,NC,I,50)
00210 01563000 25 CONTINUE
00211 01564000 DO 30 J = 2,NC,NCS
00216 01565000 JM = J-1
00221 01566000 IF (.NOT.MASK) GO TO 1
00232 01567000 IF (SCANLINE(J,1) EQ.0.OR.SCANLINE(JM,1)EQ.0) GO TO 30
00251 01568000 1 DO 40 K = 1,ND
00256 01569000 IT = INTTHR(K)+IABS(SCANLINE(J,K)-SCANLINE(JM,K))
00277 01570000 IF (IT.GE.32245) GO TO 660
00305 01571000 40 INTTHR(K) = IT
00311 01572000 NUM = NUM+1
00312 01573000 30 CONTINUE
00314 01574000 660 OONUM = IS.*DELTGT/SORT(FLOAT(ND))/FLOAT(NUM)
00320 01575000 DO 50 K = 1,ND
00333 01576000 50 INTTHR(K) = IFIX(FLOAT(INTTHR(K))*OONUM)
00347 01577000 DO 66 K = 1,ND
00354 01578000 66 KOUNT(K) = 0
00360 01579000 DO 665 NN = 2,N20,NSEL
00365 01580000 DO 65 N = 1,20,5
00372 01581000 IS = N*NN
00375 01582000 IF (IS GE.NR) IS = NR/2
00403 01583000 DO 100 I = IS,NR,17
00410 01584000 DO 25 I = 1,NC
00415 01585000 CALL READP(UICB,IND,INGIN,SCANLINE(I,I),2,K,I,1,NC,K+1,I,1,NC)
00447 01586000 IF (IND(I).LT.0) CALL CHKID(UICB,IND,INGIN,IF,NC,I,60)
00467 01587000 65 CONTINUE

```

PAGE 0048 THRFND

```

00470 01398000      DO 60 J = NN,NC,3
00475 01399000      JM = J-1
00500 01390000      IF (.NOT.HASK) GO TO 2
00505 01391000      IF (SCANLINE(J,1) EQ. 0. OR. SCANLINE(JR,1) EQ. 0) GO TO 60
00524 01392000      2 FLAG = .TRUE.
00526 01393000      DO 70 K = 1,ND
00533 01394000      IF (ABS(SCANLINE(J,K)-SCANLINE(JR,K)).LT. INTTHR(K)) GO TO 70
00539 01395000      KOUNT(K) = KOUNT(K)+1
00561 01396000      FLAG = .FALSE.
00562 01397000      70 CONTINUE
00564 01398000      IF (FLAG) MFND = MFND+1
00567 01399000      TARGET = TARGET+DELTGT
00573 01600000      60 CONTINUE
00574 01601000      IT = IFIX(TARGET)
00601 01602000      IF (MFND GT. IT+3) GO TO 80
00607 01603000      IF (MFND GE. IT-3) GO TO 100
00615 01604000      C 100 FEW--INCREASE
00615 01605000      MAX = KOUNT(1)
00620 01606000      KI = 1
00622 01607000      UP = .TRUE.
00624 01608000      DO 61 K = 2,ND
00631 01609000      IF (KOUNT(K).LE.MAX) GO TO 61
00636 01610000      MAX = KOUNT(K)
00641 01611000      KI = K
00643 01612000      61 CONTINUE
00644 01613000      INTTHR(KI) = INTTHR(KI)+1
00650 01614000      GO TO 63
00651 01615000      C 100 MANY--DECREASE
00651 01616000      80 MIN = KOUNT(1)
00654 01617000      KI = 1
00656 01618000      DOWN = .TRUE.
00660 01619000      DO 81 K = 2,ND
00665 01620000      IF (KOUNT(K).GE.MIN) GO TO 81
00672 01621000      MIN = KOUNT(K)
00675 01622000      KI = K
00677 01623000      81 CONTINUE
00700 01624000      INTTHR(KI) = INTTHR(KI)-1
00704 01625000      C RESTORE COUNTERS
00704 01626000      63 DO 83 K = 1,ND
00711 01627000      83 KOUNT(K) = 0
00715 01628000      MFND = 0
00717 01629000      TARGET = DELTGT
00721 01630000      100 CONTINUE
00722 01631000      65 CONTINUE
00723 01632000      IF (BOTH) GO TO 666
00727 01633000      BOTH = UP.AND.DOWN
00733 01634000      665 CONTINUE
00734 01635000      666 RETURN
00735 01636000      END

```

SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
BOTH	LOGICAL	SIMPLE VAR	0+221	BUFFER	INTEGER	ARRAY	0+223 .1
CMKID		SUBROUTINE		DELST	REAL	SIMPLE VAR	0+226
DOUN	LOGICAL	SIMPLE VAR	0+226	FLAG	LOGICAL	SIMPLE VAR	0+222
I	INTEGER	SIMPLE VAR	0+24	INGIN	INTEGER	SIMPLE VAR	0+24 .1
IND	INTEGER	ARRAY	0-212 .1	INTNR	INTEGER	ARRAY	0-215 .1
IR	INTEGER	SIMPLE VAR	0+212	IS	INTEGER	SIMPLE VAR	0+214
IT	INTEGER	SIMPLE VAR	0+216	J	INTEGER	SIMPLE VAR	0+213
JM	INTEGER	SIMPLE VAR	0+26	K	INTEGER	SIMPLE VAR	0+227
KI	INTEGER	SIMPLE VAR	0+230	KOUNT	INTEGER	ARRAY	0-211 .1
NASF	LOGICAL	SIMPLE VAR	0-25 .1	MAX	INTEGER	SIMPLE VAR	0+223
NIN	INTEGER	SIMPLE VAR	0+231	N	INTEGER	SIMPLE VAR	0+25
N20	INTEGER	SIMPLE VAR	0+211	NC	INTEGER	SIMPLE VAR	0-27 .1
NCS	INTEGER	SIMPLE VAR	0+225	ND	INTEGER	SIMPLE VAR	0-26 .1
NFL	INTEGER	SIMPLE VAR	0-216 .1	NFLD	INTEGER	SIMPLE VAR	0+215
NFND	INTEGER	SIMPLE VAR	0+210	NH	INTEGER	SIMPLE VAR	0+217
NR	INTEGER	SIMPLE VAR	0-210 .1	NSEL	INTEGER	SIMPLE VAR	0+220
NUM	INTEGER	SIMPLE VAR	0+224	ODUN	REAL	SIMPLE VAR	0+234
READP		SUBROUTINE		SCANLINE	INTEGER	ARRAY	0-214 .1
SBRT	REAL	FUNCTION		TARGET	REAL	SIMPLE VAR	0+232
THRFD		SUBROUTINE		UICB	INTEGER	ARRAY	0-213 .1
UP	LOGICAL	SIMPLE VAR	0+27				

PROGRAM UNIT THRFD COMPILED

Parent: NUMCLU

UNCLE

UNCLE(TSPXL,MEAN,ND,MAX,NEAR,SUM,ID)

This program finds the cluster from MEAN nearest TSPXL. Only clusters with index I such that  $SUM(I) \neq 0$  are considered. The distance is returned as ID, the index as NEAR.

Method: Self-documenting.Program Variables

ID	INTEGER	The squared distance (biased by -32768) from TSPXL to the nearest cluster in MEAN.
IS,IT	INTEGER	Used to compute the distance.
K,M	INTEGER	DO loop index.
MAX	INTEGER	The largest possible number of means. Inactive means have $SUM(.) = 0$ .
MEAN(ND,MAX)	INTEGER ARRAY	The cluster centers.
ND	INTEGER	Dimensionality
NEAR	INTEGER	Returned index of nearest cluster to TSPXL.
SUM(MAX)	INTEGER ARRAY	An indicator that a cluster has been eliminated. If $SUM(I) = 0$ , then cluster MEAN(.,I) is gone.
TSPXL(ND)	INTEGER ARRAY	The point to be classified.

**PRECEDING PAGE BLANK NOT FILMED**

PAGE 0030 HEULETT-PACKARD J21020.01.03 FORTRAN/3000 TUE. OCT 13, 1901, 9:41 AM

```

00007 02043000 $CONTROL SECHENT=ANODEBASEC
00007 02044000 SUBROUTINE UNCLE(TSPXL,MEAN,NO,MAX,NEAR,SUN,IS)
00007 02045000 C
00007 02046000 C PARENT PROGRAM: HUNCLO
00007 02047000 C
00007 02048000 C SUBROUTINE UNCLE FINDS THE CLUSTER NEAR FROM MEAN
00007 02049000 C WHICH IS CLOSEST TO TSPXL WITHOUT REGARD FOR HOW FAR
00007 02050000 C THIS MIGHT BE. ONLY CLUSTERS K WITH SUN(K) NOT ZERO
00007 02051000 C ARE CONSIDERED.
00007 02052000 INTEGER*2 TSPXL(NO),MEAN(NO,MAX),SUN(MAX)
00007 02053000 ID = 16000
00011 02054000 DO 1 K = 1,MAX
00016 02055000 IF (SUN(K).EQ.0) GO TO 1
00024 02056000 IS = -32768
00033 02057000 DO 2 N = 1,NO
00040 02058000 IF (IS.GT.ID) GO TO 1
00047 02059000 IT = MEAN(N,K)-TSPXL(N)
00060 02060000 2 IS = IS+IT*IT
00069 02061000 IF (IS.GT.ID) GO TO 1
00071 02062000 ID = IS
00073 02063000 NEAR = K
00073 02064000 1 CONTINUE
00076 02065000 R E T U R N
00077 02066000 END
    
```

SYMBOL MAP

NAME	TYPE	STRUCTURE	ADDRESS	NAME	TYPE	STRUCTURE	ADDRESS
ID	INTEGER	SIMPLE VAR	0-24 ,I	IS	INTEGER	SIMPLE VAR	0+23
IT	INTEGER	SIMPLE VAR	0+24	K	INTEGER	SIMPLE VAR	0+26
N	INTEGER	SIMPLE VAR	0+25	MAX	INTEGER	SIMPLE VAR	0-27 ,I
MEAN	INTEGER	ARRAY	0-211 ,I	NO	INTEGER	SIMPLE VAR	0-210 ,I
NEAR	INTEGER	SIMPLE VAR	0-26 ,I	SUN	INTEGER	ARRAY	0-23 ,I
TSPXL	INTEGER	ARRAY	0-212 ,I	UNCLE		SUBROUTINE	

PROGRAM UNIT UNCLE COMPILED

PRECEDING PAGE BLANK NOT FILMED

## APPENDIX A

## THE THEORETICAL FOUNDATION OF AMOEBA

The Assumptions. The clustering technique AMOEBA is based on three groups of theoretical statements. The first group concerns the relationship between the spatial and spectral behavior of the data. Roughly speaking, it is assumed that spectrally homogeneous groups of pixels found in spatially connected blobs represent the real classes. The second group contains a definition of a new concept (the pair probability of misclustering) and specifies that in clustering it is desired to minimize this probability. The third group concerns the problem of handling the classification of pixels which are on the spatial boundary between two real classes.

Group A

A1 Real classes exist and can be distinguished using digital multi-imagery.

Discussion: While it might not be questionable that real classes exist, it is certainly not clear that this is the case for their representation in multi-image measurements. Assumption A1 may fail in clustering if too much is hoped for in the identification of clusters in the data with real world classes. On the other hand, the assumption must certainly be held, at least implicitly, by all who would cluster the data looking for associations homologous to real classes.

It can also be observed that digital multi-image data consists of pixels at the atomic level. A consequence of assumption A1 is that, at least for some pixels, it is meaningful to ask what real class a pixel belongs to. It is clearly not possible to ask this of all pixels. Pixels on spatial boundaries display erratic statistical fluxuations which are generally difficult to model. This is particularly true of data which is one or more of

- (a) data sampled in a particular scan line direction and subjected to significant band-width limited processing after sampling;

- (b) pictorial imagery with sampling at a density comparable to the point-spread of the lens system;
- (c) multi-temporal imagery in which imperfect registration has been performed: that is, multi-imagery in which spatial adjustments have been made so that pixels from various single images are samples from approximately the same spatial point.

(It can be noted that Landsat multi-temporal multi-spectral data enjoys all of these properties.) For digital multi-imagery, we can distinguish, at least in principle, between mixture pixels and pure pixels (the rest). Mixture pixels arise as a consequence of finite bandwidth ((a) or (b)) or imperfect registration (c). If the data is sampled efficiently and real classes are found in small groups then many pixels will be mixtures. On the other hand, if it is to be believed that an adequate spatial sample is available, then each real class must be represented in at least some pure pixel associations. In order to make this more precise, we introduce some terminology.

Let  $I$  denote the digital image. The next three assumptions concern the existence of a set  $P \subset I$  of pixels such that neighboring pixels in the set are unusually like one another in measurement space. Call two pixels with spatial coordinates  $(i,j)$  and  $(n,m)$  neighbors if  $|i - n| + |j - m| = 1$ . (A pixel inside the image has four neighbors.) A path is an ordered sequence  $p_1, \dots, p_n$  such that  $p_{k-1}$  is a neighbor of  $p_k$  for  $k = 2, \dots, n$ . A set  $Q$  is said to be connected if for each pair  $p, q$  in  $Q$  there is a path  $p_1, \dots, p_n$  in  $Q$  with  $p_1 = p$  and  $p_n = q$ . It is easy to see that any non-void  $P \subset I$  is a union of non-void maximal connected sets  $Q_i$ ,  $i = 1, \dots, k$  with  $Q_i \cap Q_j = \emptyset$  for  $i \neq j$ ; the components  $Q_i$  are uniquely determined.

We assume that:

A2 A subset  $P$  of  $I$  has the property that each pixel  $p \in P$  is a pure measurement from a real class.

Call the components of  $P$  patches; consequences of the purity assumption and the discussion above on sampling are the following two statements.

Rather than formalize the sampling discussion, we simply assume:

- A3. All pixels from a given patch are measurements from the same real class.
- A4. Each real class has at least one measurement pixel in  $P$ .

### Group B

Consider a clustering  $C = \{C_0, \dots, C_m\}$  of the data. (It is convenient to include a cluster  $C_0$  which one might call "unknown"; most classification rules allow a threshold in which a pixel is not assigned to any cluster.) In what follows, we call a clustering of a pixel  $p$  in  $C_i$  meaningful if and only if  $i > 0$ . Consider also the unknown real partition of data into pure real classes  $\{R_1, \dots, R_k\}$  plus a mixture class  $R_0$ . These are not simply unknown: they depend on the observer, and so are unknowable. In clustering, one might hope to minimize the "probability of misclassification." Unfortunately, since the clusters are not labeled and, indeed, no labels independent of an external observer exist, this concept is meaningless.

One observation we can make right away is the following: it is clearly an error if  $p \in R_i$  for  $i \neq 0$  and  $p \in C_0$ . This is actually a restriction on the "rejection thresholds", and is used in AMOEBA to determine when the clustering is going astray. Here we simply assume  $R_i \cap C_0 = \emptyset$  when  $i \neq 0$ .

Consider a pair  $\{p, q\}$  of pure pixels. Let  $r(s)$  denote the real class a pixel  $s$  is in and  $c(s)$  the cluster. Since  $c(p) \neq 0$  and  $c(q) \neq 0$ ,  $p$  and  $q$  are clustered in meaningful clusters, and there are four cases:

- (i)  $r(p) = r(q)$  and  $c(p) = c(q)$  ;
- (ii)  $r(p) \neq r(q)$  and  $c(p) \neq c(q)$  ;
- (iii)  $r(p) = r(q)$  and  $c(p) \neq c(q)$  ;
- (iv)  $r(p) \neq r(q)$  and  $c(p) = c(q)$  .

The last two cases are errors.

B1 Definition. The probability that two pure pixels are in the same real class and are clustered differently plus the probability that two pure pixels are in distinct real classes and are clustered alike is called the pair probability of misclustering (PPMC).

B2 Objective. In clustering pure pixels, it is desired that  
 (a) each pure pixel be assigned to a meaningful cluster, and  
 (b) the PPMC is minimal.

Before examining just how these assumptions are developed into a clustering program, we consider the problem of handling mixture pixels.

#### Group C

The underlying classification rule used in AMOEBA is a nearest neighbor (Euclidean distance) to cluster center. A clustering program based on the model discussed above furnishes cluster centers. The pixels are tentatively classified by nearest cluster center and this classification is checked spatially. We assume:

- C.1.a. The nearest cluster center classification is generally accurate.
- b. Each pixel with two, three or four spatial neighbors in the same class is acceptably classified.
- c. Each pixel with no neighbor in the same class is not correctly classified.

Assumption C.1 allows us to locate and mark pixels with one or no neighbor in the same class for examination. Most of these pixels are boundary pixels. To model this situation, we assume:

- C.2 Each pixel on a spatial boundary is, as a measurement vector, a convex combination of the cluster centers in which two of its four neighbors are classified.

Although this model ignores both contaminated boundaries and registration errors, it leads to a method for reclassifying apparent errors and, unexpectedly, to a cluster-dependent rejection threshold. Note that, if  $b = \alpha p + (1 - \alpha)q$  is a convex combination of vectors  $p$  and  $q$ ,

then  $\text{dist}(b, \text{nearer of } p, q) \leq \frac{1}{2} \text{dist}(p, q)$ . Suppose  $p$  is a pixel which was marked as having no neighbor in the same class. Let  $q_1, q_2, q_3$  and  $q_4$  be the cluster centers of the classes of the four neighbors of  $p$  which are acceptably classified. (Usually at most two are distinct, and often only one neighbor is in a valid cluster at this point.) For each cluster  $i$ , let  $r(i) = \frac{1}{2} \max_j \text{dist}(c_i, c_j)$  denote half the distance from cluster center  $c_i$  to the other furthestest away. Reclassify pixel  $p$  in class  $q_i$  provided  $\text{dist}(p, q_i) < r(i)$  and  $\text{dist}(p, q_i) \leq \min_{i \neq j} \{d : d = \text{dist}(p, q_j) \text{ and } d < r(j)\}$ .

The IDIMS function AMOEBA represents one attempt to follow this model as far as it can take us. We only make two concessions to reality: First, the boundary estimation program is good but not perfect, so we do not actually classify patches as one unit. Second, registration errors blow the mixture  $\sqrt{2}$  high (that is, in registration-error pixels, the  $\alpha$  depends on the band, but, even so, the distance to the closest is more than  $\sqrt{2} \cdot 1/2 d(p, q) \approx .7 \text{dist}(p, q)$ ).

## APPENDIX B

## SYSTEM SUBROUTINES

* ATHWDS	Authorize work data sets (temporary images)
* CHKIO	Check for errors after an I/O is performed
* CLOSEP	Close a picture file
* DELWDS	Deauthorize work data sets
** FCHECK	Check for I/O errors
** FCLOSE	Close a file
** FOPEN	Open a file
** FWRITE	Write a record
* OPENPI	Open the input image disk file
* OPENPO	Open the output image disk file
* Params	Prompt for user parameters
* PRINTP	Print a message
* READP	Read a portion of an image
* WRITEP	Write a portion of an image

\* Supplied by ESL. Reference "IDIMS Applications Programmers Guide"  
ESL-TM1047

\*\* Supplied by Hewlett-Packard. Reference "MPE Intrinsic Reference  
Manual" Part No. 3000-90010.

**PRECEDING PAGE BLANK NOT FILMED**

175

APPENDIX C

IDIMS USER DOCUMENTATION

PRECEDING PAGE BLANK NOT FILMED

## AMOEBA

A. PURPOSE

Performs completely unsupervised clustering and classification of a multispectral image using a spatial-spectral clustering algorithm.

B. INPUT AND OUTPUT

The input image must contain between 2 and 16 bands, and must contain no measurement less than zero or greater than 127. The output image is of type BYTE.

C. PARAMETERS

There are 7 optional parameters. They are:

STATFILE = Statistics file name (alphanumeric character) that output statistics data is to be stored in.

PCTFLDS = The user's estimate of the percent of the image contained in "fields"--spatially connected spectrally homogeneous areas; (integer)

Default = 45

CHANIMAP\* = Shall a map of band one of the image be sent to the user? (character)

Default = 'N'

LABELMAP\* = Shall a map of labels be sent to the user? (character)

Default = 'N'

CLASSMAP\* = Shall a classification map be sent to the user? (character)

Default = 'N'

MASK = Shall a value of 0 in band 1 be taken as a mask (i.e., not part of the image)? (character)

Default = 'Y'

MINCLUS = User's desired minimum number of clusters. If negative, exactly -MINCLUS clusters will be sought. If positive, at least MINCLUS clusters will be sought (integer)

Default = 10

MAXCLN = User's desired maximum number of clusters. May not exceed 98.

Default = 98

D. EXAMPLE

INIMAGE > AMOEBA > OUTCLUST

PRECEDING PAGE BLANK NOT FILMED

\*These are primarily debugging aids to give the user a quick look at the data and follow the progress of the function. Additional parameters PRINTSL (starting line), PRINT NL (number of lines), and PRINTSS (starting sample) are required.

After prompting for parameters, messages to the user will appear as follows: (assume the image is NB bands and all default options are taken).

INTTHR = (list of boundary detection thresholds)

#LABELS = (number of distinct "fields" found; a field is defined as a connected area of non-boundary)

#TSTSTS = (number of "test sets" found. A test set is a set of 5 pixels collected from the same field).

Minimum number of clusters sought: 10 start with nn clusters, mm test points. kk clusters have non void assignments.

Square of diameters of starting clusters: sss

Number of clusters: cc

Estimate of Pair PMC: pp percent.

Final number of clusters = ff

(number in l) ("center" of l)

(number in ff) ("center" of ff)

There are uu unclassified.

The mask contains aa points.

End function--AMOEBa

The meaning of most of these outputs is explained in the algorithm documentation (F). The principal user output is the list of clusters "centers" (attractors is probably a better term) and the number of image elements assigned to that center.

#### E. DIAGNOSTIC MESSAGES

There are five messages AMOEBA may return:

1. Your image contains a value over 127. Please use MAP to put into the range 0-127. FUNCTION DOES NOT SUPPORT INPUT DATA TYPE

A value over 127 was encountered.

2. NUMBER OF BANDS SPECIFIED NOT ALLOWED BY FUNCTION

Number of bands must be at least 2 and at most 16.

3. EXTERNAL FILE COULD NOT BE ACCESSED

The STATFILE name is already in use.

4. SPECIFIED NON-IMAGE FILE PREFIX INVALID

The STATFILE name contains more than 8 characters, or is otherwise invalid.

## 5. INPUT IMAGE SIZE EXCEEDS FUNCTION CAPABILITIES

The input image is too wide for even one buffer on input and one on output. Use MOSAIC to segment the image into strips. (The number of bands and number of lines do not matter, only the number of samples).

Additional messages may be returned if I/O problems are encountered during operation of the function.

## F. ALGORITHM

The clustering and classification function AMOEBA is based on a simple model for image data. In the model, the concepts of boundary, field, and classification are defined, and assumptions are made about the accuracy of a clustering in terms of the classification. The classification is based on a spatially modified nearest neighbor classifier (Euclidean distance); to train such a classifier, one needs to know only the number of classes and the class "centers". The actual function proceeds in steps as follows; subroutine names are given in parenthesis:

### Find boundary detection vector thresholds (THRFND):

Boundary detection is based on vector gradient thresholds (rather than a norm threshold or other one-dimensional decision rule). Thus one threshold is determined for each band. Initially the image is sparsely sampled to estimate with each band contributing about the same number of boundary decision estimates. These thresholds are passed to the next step.

### Find connected sets of non-boundary (START):

The data is scanned three lines at a time (in a circular buffer), and a circular buffer of labels is created. A boundary decision results when a point and its neighbor differ by more than the threshold in any band. "Cracks" are filled in, and intervals are located on each new labels line. These are then connected to the previous line of labels, and the previous line written to disk. The resulting intermediate image contains -32768 (the smallest 16 bit two's complement integer) marking boundary, or n (which starts at -32767 and is incremented) labeling connected sets of non-boundary. #LABELS = n is printed.

### Extract test sets and store on disk scratch file (ASELECT)

The labels map and data are scanned, and as large a buffer as there is memory for is allocated to accumulate samples bearing the same label. When the buffer fills (or at end of the data), each same-label batch is sampled, taking every fifth point (from batches with at least 5), and the sample sets are stored on disk. There can be as many as 64K-1 such sets. These are passed to the next step. Their number is printed (#TSTSTS). The temporary labels map is deleted from disk.

### Thin test sets and accumulate starting clusters (THINTSTM)

The starting clusters are to be means of samples taken from the same component of the complement of the boundary. Accordingly, they should be spectrally purer, since this tends to minimize registration errors and reduce noise. However, it is out of the question to classify 10,000

things in 2,000 classes, and then may be even more than this many test sets (i.e., more than 2,000) in a large image. We therefore reduce the number by (a) removing apparent duplicate means, (b) removing test sets in which the first sample is relatively unlike the last, and (c) removing apparent duplicate test sets. The final number of test sets is printed in the next step. The temporary test set file is deleted from disk.

#### Find the clusters and their number (NUMCLU)

First sort the test sets in increasing order based on the sum of the odd channel values. Samples from the same test set are in the same "field", and therefore are from the same real class (on the model assumptions). Samples spread out in this order tend to be from different real classes. Errors are made by the classifier when a center attracts points from different real classes or participates in the splitting of a pair from the same test set. Therefore, centers which make errors are eliminated. A running estimate is kept of the probability that a pair from different classes is clustered alike, plus the probability that a pair from the same real class is clustered differently. The minimum value of this estimate of the Pair PMC is used to determine the number of clusters and exactly what they are.

#### Classify and count (CLASSIFY)

A spatially modified nearest neighbor classification is now performed. Initially, each point is classified by nearest neighbor. Then this classification is checked for accuracy by looking at the classification of the four nearest neighbors. Points with one neighbor in the same class are deemed OK!. Not-OK points are examined with the view of reclassification in the class of OK - neighbors, provided this can be done consistently with the anticipated spectral appearance of a mixture pixel. No reclassification based on spatial content alone is performed. That is, all reclassification must fit the mixture and registration error model. Circular buffers are managed (similarly to START), and the checked nearest neighbor classification is written to disk as a type BYTE image.

#### Finish (AMSTATS)

The optional STATFILE is written.

#### G. COMMENT

The user is advised to be cautious about interpreting any clustering of image data. Many images, indeed, are not suitable for clustering. If AMOFBA is selected, the output parameter Pair PMC is a good indication of accuracy: more than 25 percent and the area was probably poorly clustered. Under 20 and clustering was at least self-consistent.

Users wanting to learn more about the method or the underlying model should consult the reference.

#### H. REFERENCE

Jack Bryant, "On the clustering of multidimensional pictorial data", Pattern Recognition 11, pp. 115-125 (1979).

APPENDIX D

SAMPLE INTERACTIVE SESSION





## APPENDIX E

## SAMPLE BATCH JOB

1. Create, using the Editor, and store the following file:

```
!JOB  jobcard
!IDIMS
inputimage>AMOEBA(parameterlist)>outputimage
>END
!EOJ
```

2. Exit the Editor and enter  
: STREAM filename.
3. Return later for results.

In some systems, it may be mandatory to store the input image (or perhaps both); someone should be around to respond to requests to hang tapes.

N83 12496

D2

RICE SCENE RADIATION

RESEARCH PLAN

James Heilman  
Remote Sensing Center  
Texas A&M University  
College Station, Texas 77843

December, 1981

## RICE SCENE RADIATION RESEARCH PLAN

The goal of rice scene radiation research is to develop an understanding of the functional relationships between rice and its spectral characteristics. These functional relationships will be integrated into spectral - agrometeorological models for use in crop identification, development stage estimation, and condition assessment.

### CROP IDENTIFICATION

#### Introduction

Knowledge of the cultural and biophysical characteristics of crops and their relationships to spectral response are important inputs to the pattern recognition research effort. For crop identification, this research will provide information on what crops can and cannot be separated using the current and planned sensor technologies, what additional kinds of measurements are needed, and the important times and frequency of observations needed to enable crop discrimination and identification. For the sampling and estimation research effort, knowledge of the cultural and biophysical characteristics of crops which significantly affect spectral response is needed in order to account for the agronomic factors of importance in an advanced dynamic sampling and estimation approach.

#### Technical Issues

Scene radiation applied research issues in crop identification which have been defined are:

1. What are the key cultural and biophysical characteristics of crops (which are potentially observable from remotely sensed data) that

permit separability between crops and identifiability of crop types at harvest and earlier in the season?

2. How are the cultural and biophysical characteristics related to crop type manifested in the spectral response observed by existing and planned sensors such as MSS, TM, and other advanced sensors?
3. What new kinds of observations are needed to improve the estimates of key crop characteristics shown to permit separability between crops and identifiability of crop types?

### Technical Approach

Crop identification research for rice involves: field research, canopy modeling, and Landsat data analysis. Field research data will include intensive agronomic and spectral measurements. Canopy geometry measurements and available information on leaf reflectance and transmittance will enable modeling of canopies and thus increase the range of canopy conditions which can be studied with confidence. Use of field measurements and canopy modeling will enable extension of that knowledge to the relationship with Landsat MSS and TM data.

### Task Descriptions

#### Identify Cultural and Biophysical Characteristics

#### Related to Crop Identification

The first technical issue has the following specific objectives:

- \* Determine the key difference between rice and its confusion crops in the timing and duration of key physiological and cultural events.
- \* Determine the key differences in canopy geometry among rice varieties and the optical properties of canopy components.

- \* Determine the cultural, regional, and environmental differences among rice and its confusion crops.
- \* Represent the distribution of the key crop characteristics by functional forms.

A literature review will be conducted to identify planting dates, regional crop calendars, soil surveys and other descriptions of management practices. The periodic observations acquired on sample segments in the United States will also be used to help detail management practices (e.g. row width, planting dates) and provide more localized information on planting date and development stage.

When the data have been compiled, the crop characteristics will be related to geographic region and to other scene characteristics and management practices. Functional forms will be found which describe the distributions within and across the geographic regions.

#### Relate Agronomic Characteristics to Spectral Response

The second objective is to determine how cultural and biophysical characteristics affect in the spectral response of rice observed by existing and planned sensors such as MSS, TM, and other advanced sensors, and has two sub-objectives.

- \* Determine the relationships among key crop characteristics, remote sensing observables (band means and transformations with current and planned sensor systems), and background effects (e.g., soil and water background, atmospheric haze, view angle, and sun angle).
- \* Determine which remote sensing observables are predominantly a function of the crop characteristics of interest and are least sensitive to background effects.

A set of remote sensing observables including MSS bands, tasseled cap components, MSS band ratios (e.g., 7/5), TM bands, transformations of TM bands, and bands of other sensors will be examined. For all sensors, research into appropriate bands or transformations for estimation of particular canopy characteristics will be conducted.

The relationships of these remote sensing observables to crop characteristics and to scene characteristics which are not of interest will be examined. To do this, both field-acquired data and data modeled using canopy geometry and leaf reflectance and transmittance measurements will be used. Correlations, regressions, and linear and nonlinear models will be used as appropriate to describe the relationship of the remote sensing observables to and the amount of variability due to: green leaf area index, percent soil cover, green biomass, development stage, and temporal trajectories.

After the functional relationships have been determined, sensitivity analyses will be conducted on the variables of interest to determine the change in spectral response given a certain change in the canopy variable. This will enable determination of which canopy and background variables are important in determining spectral response.

Finally, the "best" bands or transformations for each of the sensors will be determined to be those which maximize sensitivity to various individual crop characteristics and minimize sensitivity to undesired effects. The crop discrimination power of these sets of remote sensing observables will be tested using multivariate analysis on data from one or more intensive test sites.

### Investigate Potential Improvements Due to New Data Types

The third objective is to determine what new kinds of observations are needed to improve the estimates of key crop characteristics shown to permit separability between crops and identifiability of crop types. The specific objectives addressing this issue are:

- \* Determine the functional relationships among key crop characteristics, background effects, and spectral response observable in other spectral regions or with other types of measurements.
- \* Identify new data types which improve the relationships with key crop characteristics used for crop discrimination while minimizing background effects.

To address this issue, spectral measurements must be acquired in the field and over test sites with sensors other than the current and planned sensors. Helicopter spectrometer and/or aircraft scanner data covering other visible and near-to-middle IR regions, thermal measurements, microwave measurements, and illumination/view angle measurements are required. The approach for addressing this issue will parallel that of the second issue except for the measurements utilized.

#### Data Requirements

The selection of treatments consists of first identifying the major sources of variation in the growth, development, and spectral response of rice. These factors include: planting date, variety, plant population/row spacing, soil conditions, and weather. The levels of each factor will be selected to sample the range of expected conditions in commercial fields.

Spectral measurements will be made in controlled plots using the EXOTECH 100A radiometer and the Barnes multiband radiometer system (having the TM bands). Detailed agronomic measurements of the crop canopies including crop development stage, vegetation measurements, crop condition, soil background condition, and grain yield will be collected.

## DEVELOPMENT STAGE ESTIMATION

### Introduction

Crop development stage is important for crop identification and yield modeling. There are three approaches for estimating crop development stages: (1) average crop calendars based on accumulation of days between stages, (2) meteorological methods based on accumulation of thermal or photo-thermal units between stages, and (3) spectral methods based on changes in spectral response as a function of development stage. The goals of this task are to investigate the use of spectral measurements to determine crop development stage and to develop a meteorologically-driven stage of development model that will accept spectral inputs.

### Technical Issues

Research issues for rice development stage estimation are:

1. What are the key biophysical characteristics of crops (which are potentially observable using remotely sensed data) that permit their development stage to be determined?
  - a. What are the critical development stages of crops with respect to crop identification, condition assessment and yield prediction?

- b. What are the key differences in timing and duration of key developmental events?
  - c. What are the key differences in canopy geometry and composition related to development stage?
  - d. How do these differences depend on cultural, environmental and geographical factors?
  - e. What are reasonably representative functional forms for the distribution functions of the key crop characteristics?
2. What are the functional relationships between development stage and the radiometric characteristics of crop canopies?
    - a. MSS bands
    - b. TM bands
    - c. Transformations of MSS and TM data
    - d. Other sensors
  3. How are the functional relationships affected by cultural, environmental and geographic factors (e.g. variety, row width, soil type, moisture stress)?
  4. How can spectrally derived development stage information best be utilized?
    - a. Development of models which, given spectral plus weather data, predict development stage
    - b. Development of models which, given development stage, agromet conditions, and canopy geometry, predict spectral response
  5. What is the improvement in performance of large area crop growth and yield models by using spectrally derived inputs (i.e., evaluation of models in the context of a large area crop yield model).

## Technical Approach

The general technical approach for addressing the research objectives in the crop development stage area will involve estimation theory. The radiometric characteristics of rice canopies will be modeled to determine functional relationships with development stage and/or time. This development will rely on both ground and satellite measured spectral data in the MSS and TM bands, and meteorological data. The trajectories of the development stage of crops in spectral space will be analyzed to identify variables with superior properties for estimating development stages of rice. Various estimators will be examined individually and together to determine their predictive abilities.

## Task Description

Four research tasks must be completed in the area of estimating crop development stage. Agronomic information of rice must be obtained describing the key biophysical characteristics that permit development stage to be determined. The necessary information will be obtained from technical literature, Texas A&M agronomists and field measurements. The key biophysical characteristics are needed to gain a physical understanding of problems associated with using spectral measurements to estimate stage of development.

The second research task involves an analysis of multiyear agronomic, spectral, and meteorological data. The development stage trajectories of rice will be examined in spectral/agronomic space to identify worthwhile spectral estimators of crop development stage.

The third research task involves development of an agrometeorological stage of development model that accepts spectral inputs. The model will be developed using agronomic and meteorological data obtained from rice experiment stations in three different climatic regions. The model will be developed to obtain a high degree of accuracy for the three maturity classes of rice.

The fourth task will consist of development of a framework for merging the spectral estimators of stage of development with the agrometeorological model so that the spectral estimates of growth stage can be used to "correct" the model estimates, if necessary.

#### Data Requirements

To identify the form of relationship between crop development stage and spectral variables and to develop initial models, reflectance measurements and observations of development stage at all growth-development stages for a representative set of cropping practices and soils are required.

The specific data requirements are:

- Reflectance measurements in the Landsat MSS bands and TM bands.
- Rice development stage observations.
- All growth and development stages from pre-planting to post-harvest sampled.
- Frequency of observations at 5-7 day intervals.
- Representative treatments above sampled (i.e., several soil types, varieties and planting dates)
- Daily meteorological data, temperature, relative humidity, solar radiation, precipitation, etc.
- Atmospheric measurements on days spectral data are acquired.

After initial model forms have been developed, a larger data set is required to test and evaluate the models. This data set should be acquired over 3-5 additional domestic and international experiment stations at locations having difference soils, weather, and cropping practices.

## CROP CONDITION ASSESSMENT

### Introduction

Potentially, multispectral data contains additional information about crops other than identification. Relatively little research has been conducted on developing and exploiting the capability of multispectral data to provide information about crop condition and yield. For example, the ratio of near infrared to red reflectances and the greenness transformation have highly significant relationships with leaf area index (LAI). Agronomic variables, such as LAI and percent soil cover, are frequently required inputs to crop models which depict limitations imposed on crop growth and yields by weather. Additionally, measures of the presence and degree of stress, such as moisture and nutrient deficits and disease and insect infestations, are potentially important inputs to crop growth and yield models. Thus, if agronomic variables related to yield could be reliably estimated from multispectral satellite data, then physiologically-based crop growth and yield models can be implemented for large areas.

### Technical Issues

The research and development program to assess crop condition and provide inputs to yield models will address the following issues:

- What are the important biophysical variables related to the condition and yield of rice? Which of these variables can be potentially estimated using remotely sensed data?
- How are the functional relationships between spectral variables and biophysical parameters of rice affected by variation in soil background, crop production practices, and environmental conditions?

### Technical Approach

Field measurements which include stress (temperature and moisture) treatments will be used to determine the effect of stress on biophysical factors. The Suits reflectance model will be used to predict changes in reflectance due to the changes in the biophysical characteristics of rice. This information will be used to support crop identification and to develop techniques for using spectral estimates of crop condition in agrometeorological yield models.

### Task Description

The two research tasks in condition assessment are to determine the key biophysical description of crop condition that can be observed by remotely sensed data, and two determine how these characteristics can be observed using remotely sensed data. These tasks will be addressed using literature reviews, historical data analysis, and field measurements.

### Data Requirements

Data requirements are the same as in Development Stage Estimation.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle THE POSSIBLE MODIFICATIONS OF THE HISSE MODEL FOR PURE LANDSAT AGRICULTURAL DATA		5. Report Date FEBRUARY 10, 1981
		6. Performing Organization Code SR-H1-04037
7. Author(s)  CHARLES PETERS	8. Performing Organization Report No. 77	
9. Performing Organization Name and Address  UNIVERSITY OF HOUSTON DEPARTMENT OF MATHEMATICS HOUSTON, TX. 77004		10. Work Unit No.
		11. Contract or Grant No. NAS9-14689
12. Sponsoring Agency Name and Address NATIONAL AERONAUTICS AND SPACE ADMINISTRATION LYNDON B. JOHNSON SPACE CENTER HOUSTON, TX. 77058 TASK MONITOR: DALE BROWNE		13. Type of Report and Period Covered TECHNICAL REPORT
14. Sponsoring Agency Code		
15. Supplementary Notes		
16. Abstract		

ORIGINAL PAGE IS  
OF POOR QUALITY

17. Key Words (Suggested by Author(s)) Maximum likelihood estimation of normal parameters, sufficient statistics, invariance.		18. Distribution Statement	
19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*

POSSIBLE MODIFICATIONS OF  
THE HISSE MODEL FOR PURE  
LANDSAT AGRICULTURAL DATA

by

Charles Peters  
Department of Mathematics  
University of Houston  
Houston, Texas

SUMMARY.

This report explores an idea, due to A. Feiveson, for relaxing the assumption of class conditional independence of LANDSAT spectral measurements within the same patch (field). Theoretical arguments are given which show that any significant refinement of the model beyond Feiveson's proposal will not allow the reduction, essential to HISSE, of the pure data to patch summary statistics. A slight alteration of the new model is shown to be a reasonable approximation to the model which describes pure data elements from the same patch as jointly gaussian with a covariance function which exhibits exponential decay with respect to spatial separation.

PRECEDING PAGE BLANK NOT FILMED

1. The Basic HISSE Model and its Modifications.

The original mathematical assumptions underlying HISSE are fully described in [7]. Briefly, they are:

- a) The sampled pure pixels are organized into  $p$  patches (fields) and corresponding to each patch  $j$ , there is a set of spectral data measurements  $X_j = (X_{j1}, \dots, X_{jN_j})$ , where  $X_{jk}$  is the (perhaps multitemporal) vector of spectral data from the  $k$ th pixel in the  $j$ th patch. For each patch  $j$ , there is also an unknown class designation  $\theta_j \in \{1, \dots, m\}$ , where  $m$  is known.
- b) The  $\{(X_j, \theta_j)\}_{j=1}^p$  are treated as independent random variables. The  $\theta_j$  have a common unknown discrete distribution  $\text{Prob}[\theta_j = \ell] = \alpha_\ell > 0$ , where  $\sum_{\ell=1}^m \alpha_\ell = 1$ .
- c) Given that  $\theta_j = \ell$ ,  $X_{j1}, \dots, X_{jN_j}$  are independently normally distributed with unknown mean  $\mu_\ell$  and unknown variance-covariance matrix  $\Omega_\ell$ .

A proposed modification due to A. Feiveson [3], introduces one additional matrix parameter for each class. Assumption (c) is changed to

- c') Given that  $\theta_j = \ell$ ,  $X_{jk} = \mu_\ell + e_j + d_{jk}$ , where  $E(e_j) = E(d_{jk}) = 0$ ,  $\text{var}(e_j) = \Sigma_\ell$ ,  $\text{var}(d_{jk}) = \psi_\ell$  and the  $e_j$ 's and  $d_{jk}$ 's are independent normal random variables. Thus the elements  $X_{j1}, \dots, X_{jN_j}$  of  $X_j$  are jointly normal with marginal distributions  $X_{jk} \sim N(\mu_\ell, \Sigma_\ell + \psi_\ell)$ , and constant within-patch covariance  $\text{cov}(X_{jk}, X_{ji}) = \Sigma_\ell$ , for  $k \neq i$ .

Notice that the original assumption (c) is a limiting case of (c') obtained by allowing  $\Sigma_{\ell} = 0$ .

For reasons discussed later, we will alter (c') to

(c'') The constant within-patch covariance for elements of the

$$j\text{th patch is } \text{cov}(X_{jk}, X_{ji}) = \frac{1}{N_j} \Sigma_{\ell}.$$

The effect of (c'') is that data elements from large patches are considered more weakly correlated than those from small patches. Assumption (c') is perhaps more appropriate if the correlation between pixels of the same patch is really independent of their spatial separation, while (c'') is better if the correlation falls off rapidly with spatial separation, on account of the preponderance of spatially distant pairs in larger patches. Calculations are presented in Section 4 to suggest that (c'') is a reasonable approximation to the average covariance between pairs when the correlation decreases exponentially with spatial separation. In Section 3 theoretical arguments are given which severely restrict the covariance models for which the patch mean vector and scatter matrix are sufficient statistics without, however, eliminating (c') and (c''). This is an important consideration, since procedures like HISSE are feasible only if the spectral information in patches can be summarized in a small number of statistics.

## 2. Numerical Procedures for the Alternative Covariance Models.

The likelihood function and iterative procedure for the current version of HISSE are given in [7] and will not be repeated here. For covariance models (c') and (c''). The likelihood functions is

$$L = \prod_{j=1}^P \log \prod_{\ell=1}^m \alpha_{\ell} f_{\ell}(X_j) = \prod_{j=1}^P \log f(X_j)$$

where the model (c')

$$f_{\ell}(X_j) = |\psi_{\ell}|^{-\frac{N_j-1}{2}} |\psi_{\ell} + N_j \Sigma_{\ell}|^{-\frac{1}{2}} \exp[-\frac{1}{2} Q'_{\ell}(X_j)]$$

and  $Q'_{\ell}(X_j) = \text{tr} \psi_{\ell}^{-1} S_j + N_j (m_j - \mu_{\ell})^T (\psi_{\ell} + N_j \Sigma_{\ell})^{-1} (m_j - \mu_{\ell}),$

while for model (c'')

$$f_{\ell}(X_j) = |\psi_{\ell}|^{-\frac{N_j-1}{2}} |\psi_{\ell} + \Sigma_{\ell}|^{-\frac{1}{2}} \exp[-\frac{1}{2} Q''_{\ell}(X_j)]$$

and  $Q''_{\ell}(X_j) = \text{tr} \psi_{\ell}^{-1} S_j + N_j (m_j - \mu_{\ell})^T (\psi_{\ell} + \Sigma_{\ell})^{-1} (m_j - \mu_{\ell}).$

In both these expressions  $m_j$  and  $S_j$  are, respectively the patch mean and scatter

$$m_j = \frac{1}{N_j} \sum_{k=1}^{N_j} X_{jk}$$

$$S_j = \sum_{k=1}^{N_j} (X_{jk} - m_j)(X_{jk} - m_j)^T.$$

Thus for both of these covariance models the patch mean and scatter are jointly sufficient.

The unconstrained likelihood equations for model (c'') have the form

$$(1.1) \quad \alpha_{\ell} = \frac{1}{p} \sum_{j=1}^p \frac{\alpha_{\ell} f_{\ell}(X_j)}{f(X_j)}$$

$$(1.2) \quad \mu_{\ell} = \sum_{j=1}^p N_j \frac{f_{\ell}(X_j)}{f(X_j)} m_j / \sum_{j=1}^p N_j \frac{f_{\ell}(X_j)}{f(X_j)}$$

**ORIGINAL PAGE IS  
OF POOR QUALITY**

$$(1.3) \quad \psi_{\ell} = \frac{\sum_{j=1}^p \frac{f_{\ell}(X_j)}{f(X_j)} S_j}{\sum_{j=1}^p (N_j - 1)} \frac{f_{\ell}(X_j)}{f(X_j)}$$

$$(1.4) \quad \Omega_{\ell} = \frac{\sum_{j=1}^p \frac{f_{\ell}(X_j)}{f(X_j)} N_j (m_j - \mu_{\ell})(m_j - \mu_{\ell})^T}{\sum_{j=1}^p \frac{f_{\ell}(X_j)}{f(X_j)}}$$

where the new parameter  $\Omega_{\ell}$  is defined as  $\Sigma_{\ell} + \psi_{\ell}$ .

The expressions on the right of equations (1.1) - (1.4) are appealing in that they are averages of quantities whose expectations, given  $\theta_j = \ell$ , are the parameters on the left. In addition, the successive substitutions scheme suggested by equations (1.1) - (1.4) is a slight variation of the generalized E-M procedure of Dempster, Laird, and Rubin [2]. For covariance model (c'), the likelihood equations do not suggest a natural iterative procedure and it appears that the generalized E-M procedure has no simple formulation.

To be consistent with the original interpretation of the parameter  $\Sigma_{\ell}$  as a variance-covariance matrix, it is necessary to maximize the likelihood subject to the additional inequality constraint  $\Omega_{\ell} \geq \psi_{\ell}$ . Since a solution of equations (1.1) - (1.4) need not satisfy this constraint, maximizing the likelihood subject to  $\Omega_{\ell} \geq \psi_{\ell}$  requires a much more complicated numerical procedure. The condition  $\Omega_{\ell} \geq \psi_{\ell}$  is equivalent to a set of scalar inequality and nonlinear equality constraints, and numerical procedures for such problems are generally very slow to converge. The unconstrained maximum likelihood procedure is appropriate if as in (c'') we merely assume that  $\text{cov}(X_{j1}, X_{jk})$  is the same for all  $i$  and  $k$ , without introducing random variables  $e_j$  and  $d_{jk}$ .

### 3. Covariance Models for which patch mean and scatter are sufficient.

Let  $X = (X_1 | \dots | X_N)_{n \times N}$  be a matrix whose columns are jointly normally

ORIGINAL PAGE IS  
OF POOR QUALITY

distributed  $n$ -vectors. We are interested in characterizing those families of distributions of  $X$  for which the statistic  $(m, S)$  is sufficient, where  $m = X_1 + \dots + X_N$  and  $S = X_1 X_1^T + \dots + X_N X_N^T$ . We begin by recalling the following definitions [4, p. 32].

Definition: Let  $G$  be a group of homeomorphisms on  $\mathbb{R}^n$ . A function  $T$  defined on  $\mathbb{R}^n$  is invariant under  $G$  if  $T(gx) = T(x)$  for all  $x \in \mathbb{R}^n$ ,  $g \in G$ .  $T$  is a maximal invariant of  $G$  if  $T$  is invariant and  $T(x) = T(y)$  implies that there is a  $g \in G$  such that  $y = gx$ . A measure  $\lambda$  is invariant under  $G$  if  $\lambda g = \lambda$  for all  $g \in G$ , where  $\lambda g(E) = \lambda(g(E))$ .

Lemma 1: Let elements of  $\mathbb{R}^{nN}$  be represented as  $x = (x_1 | \dots | x_N)$  and let  $e^T = (1, 1, \dots, 1)_{1 \times N}$ . For each  $N \times N$  real orthogonal matrix  $u$  satisfying  $ue = e$ , let  $g_u(x) = xu$ . Then  $T(x) = (m, S) = (xe, xx^T)$  is a maximal invariant of the group  $G = \{g_u\}$ .

Proof:  $T(g_u x) = (xue, xu(xu)^T) = (xe, xx^T) = T(x)$ . Thus  $T$  is invariant. Suppose that  $T(x) = T(y)$  so that  $xe = ye$  and  $xx^T = yy^T$ . If  $x^{(i)}$  and  $y^{(i)}$  denote the  $i$ th rows of  $x$  and  $y$  then  $x^{(i)} x^{(j)T} = y^{(i)} y^{(j)T}$  and  $x^{(i)} e = y^{(i)} e$  for all  $i$  and  $j$ . This implies that corresponding rows of  $x$  and  $y$  have the same Euclidean norm and form the same angle with the vector  $e^T$ . In addition, the rows of  $x$  describe the same set of angles in  $\mathbb{R}^N$  as do the corresponding rows of  $y$ . Thus, by carrying out parallel Gram-Schmidt procedures on  $\{e^T, x^{(1)}, \dots, x^{(n)}\}$  and  $\{e^T, y^{(1)}, \dots, y^{(n)}\}$ , it is easy to construct an orthogonal matrix  $u$  such that  $e^T u = e^T$  and  $x^{(i)} u = y^{(i)}$  for each  $i$ ; that is, such that  $y = g_u x$ . Therefore  $T$  is a maximal invariant.

Example: Any linear function  $T$  defined on  $\mathbb{R}^n$  is a maximal invariant under the group of translations by elements of the kernel of  $T$ . In fact, most of the results in [6] characterizing linear sufficient statistics depend only on this aspect of linearity.

If  $T$  is a maximal invariant then any invariant function on  $\mathbb{R}^n$  is a function of  $T(x)$ . Moreover, a function  $h \circ T$  on  $\mathbb{R}^n$  is a maximal invariant if and only if  $h$  is one to one on the range of  $T$ . In the theorems which follow we shall require that  $T$  be a continuous open mapping, in addition to being a maximal invariant. The following lemma shows that to some extent  $T$  may be chosen for convenience, with affecting the property of openness.

Lemma 2: Let  $V$  be an open subset of  $\mathbb{R}^n$ , let  $G$  be a group of homeomorphisms from  $V$  to  $V$  and let  $T_1$  and  $T_2$  be continuous maximal invariants of  $G$  defined on  $V$  with values in  $\mathbb{R}^m$ . If  $T_1$  is an open mapping then so is  $T_2$ .

Proof: Since  $T_2$  and  $T_1$  are maximal invariants, there is a one to one function  $h: T_1(V) \rightarrow T_2(V)$  such that  $T_2 = hT_1$ . Since  $h^{-1} = T_1T_2^{-1}$  on  $T_2(V)$ ,  $T_2$  is continuous and  $T_1$  is open,  $h$  is continuous. By the Brouwer invariance of domain theorem [8, p. 3]  $h$  is an open mapping. Therefore,  $T_2$  is also open.

Theorem 1: Let  $V$  be an open subset of  $\mathbb{R}^n$ , let  $\mathcal{M}$  be a homogeneous collection of finite Borel measures on  $\mathbb{R}^n$ , and let  $\lambda$  be a fixed element of  $\mathcal{M}$ . Suppose that  $\lambda(V^c) = 0$  and  $\lambda(U) > 0$  for each nonempty open subset  $U$  of  $V$ . Let  $G$  be a group of homeomorphisms from  $V$  to  $V$  such that  $\lambda(gB) = 0$  whenever  $\lambda(B) = 0$  and  $g \in G$ . Suppose that  $f_\mu$  is a continuous representative of  $\frac{d\mu}{d\lambda}$  for each  $\mu \in \mathcal{M}$  and that  $T: V \rightarrow \mathbb{R}^m$  is a continuous open maximal invariant of

G. Then  $T$  is a sufficient statistic for  $\mathcal{M}$  if and only if each  $f_\mu$  is invariant under  $G$ .

Proof: Suppose that  $T$  is sufficient. Then for each  $\mu \in \mathcal{M}$  there exists a Borel measurable function  $k_\mu$  such that  $k_\mu \cdot T$  is a version of  $d\mu/d\lambda$ , [1].

Let  $\mu \in \mathcal{M}$  and  $g \in G$  be fixed. The set

$$U = \{x \in V \mid f_\mu(x) \neq f_\mu(gx)\}$$

is an open subset of  $B \cup g^{-1}(B)$ , where

$$B = \{x \in V \mid f_\mu(x) \neq k_\mu(T(x))\}.$$

Since  $\lambda(B) = 0$ ,  $\lambda(g^{-1}(B)) = 0$  and  $\lambda(U) = 0$ . Therefore,  $U$  is empty and it follows that  $f_\mu$  is invariant. Conversely, if each  $f_\mu$  is invariant, then for each  $\mu \in \mathcal{M}$  there exists a function  $h_\mu$  such that  $f_\mu = h_\mu \cdot T$ . Since  $f_\mu$  is continuous and  $T$  is open,  $h_\mu$  is continuous on  $T(V)$ . Therefore, by [1, Corollary 6.1]  $T$  is sufficient.

Corollary 1.1: Given the hypotheses of Theorem 1, if  $\lambda$  is invariant then  $T$  is sufficient if and only if each  $\mu \in \mathcal{M}$  is invariant.

Proof: In general, a density with respect to  $\lambda$  of  $\mu g$  is  $f_{\mu g} = (f_\mu \circ g)h$ , where  $h$  is a version of  $d\lambda g/d\lambda$ . If  $\lambda$  is invariant, then we can take  $h = 1$  to obtain  $f_{\mu g} = f_\mu \circ g$  as a unique continuous density of  $\mu g$ , for each  $\mu, g$ . By Theorem 1,  $T$  is sufficient if and only if  $f_{\mu g} = f_\mu$ , which is equivalent to  $\mu g = \mu$ .

Suppose that  $\mu g \in \mathcal{M}$  for each  $\mu \in \mathcal{M}$ ,  $g \in G$  and that  $\theta$  is an  $r$ -dimensional parameterization of  $\mathcal{M}$ ; i.e., a one to one function from  $\mathcal{M}$  onto  $\Omega = \theta(\mathcal{M}) \subset \mathbb{R}^r$ . Then there is a homomorphism  $g \rightarrow \bar{g}$  from  $G$  onto a group  $\bar{G}$  of transformations on  $\Omega$  defined by  $\bar{g}(\theta_0) = \theta(\theta^{-1}(\theta_0)g)$ . The following corollary is clear.

Corollary 1.2. Given the hypothesis of Theorem 1, if  $\lambda$  is invariant then  $T$  is sufficient iff  $\bar{G}$  is the trivial group consisting only of the identity mapping on  $\Omega$ .

To apply these results to the characterization problem at hand, let  $X = (X_1 | \cdots | X_N)$  be a random  $n \times N$  matrix having one of a given family of normal distributions and let  $X^{(i)}$  denote the  $i$ th row of  $X$ . We think of  $X_1, \dots, X_N$  as being the observed random vector, but at various times wish to consider the parameters

$$\mu_i = E(X_i)$$

$$\mu^{(i)} = E(X^{(i)})$$

$$C_{ij} = \text{cov}(X_i, X_j)$$

$$R^{(ij)} = \text{cov}(X^{(i)}, X^{(j)}).$$

For the open set  $V$  of Theorem 1, we take the set of regular points of  $T(x) = (xe, xx^T)$ ; that is, the set of points  $x$  at which  $T'(x)$  is surjective.  $T'(x)$  is surjective if the matrix  $\begin{pmatrix} e \\ x \end{pmatrix}$  has rank  $n + 1$ , which is almost certainly true for any of the probabilities under consideration as soon as  $N \geq n + 1$ . Clearly any of the mappings  $g_u$  of lemma 1 is a homeomorphism from  $V$  onto itself and  $T$  is a continuous open mapping on  $V$ .  $\mathcal{M}$  will be the given set of  $nN$ -variate

normal probability measures. The invariant measure  $\lambda$  of Corollary 1.2 will be that given by  $\mu_i = 0$ ,  $C_{ij} = 0$  if  $i \neq j$ ,  $C_{ii} = I_{n \times n}$ . If  $\lambda$  is not already a member of  $\mathcal{M}$ , it may be added without affecting the sufficiency of  $T$  for  $\mathcal{M}$ . According to Corollary 1.2, and lemma 1,  $T$  is sufficient for  $\mathcal{M}$  if and only if

$$(2.1) \quad \mu^{(i)}_u = \mu^{(i)}$$

and

$$(2.2) \quad u R^{(i,j)}_u^T = R^{(i,j)}$$

for all  $i, j$  and  $u \in U = \{N \times N \text{ orthogonal matrices } u \text{ such that } ue = e\}$ .

Now, (2.1) holds if and only if each  $\mu^{(i)} = \lambda_i e^T$  for some scalar  $\lambda_i$ , which is equivalent to  $\mu_1 = \dots = \mu_N$ . In (2.2)  $U$  may be replaced by the larger set  $U' = \{N \times N \text{ orthogonal matrices such that } ue = \pm e\}$ . Let  $P = \frac{1}{N} ee^T$  and  $Q = I - P$ . Then  $U'$  is the set of all orthogonal matrices which commute with  $P$ , and (2.2) states that each  $R^{(i,j)}$  commutes with each  $u \in U'$ . Let  $w$  be an orthogonal matrix such that

$$wPw^T = \left[ \begin{array}{c|c} 1 & 0_{1 \times (N-1)} \\ \hline 0_{(N-1) \times 1} & 0_{(N-1) \times (N-1)} \end{array} \right].$$

Then  $U'$  is the set of all orthogonal matrices  $u$  such that  $wuw^T$  commutes with  $wPw^T$  and (2.2) holds iff  $wR^{(i,j)}w^T$  commutes with  $wuw^T$  for each  $u \in U'$ . Elementary calculations show that  $wuw^T$  must be of the form

$$wuw^T = \left[ \begin{array}{c|c} 1 & 0 \\ \hline 0 & v \end{array} \right]$$

where  $v$  is  $(N-1) \times (N-1)$  orthogonal, and that for some scalars  $\lambda_1^{(i,j)}$ ,  $\lambda_2^{(i,j)}$ .

$$wR^{(i,j)}w^T = \left[ \begin{array}{c|c} \lambda_1^{(i,j)} & 0 \\ \hline 0 & \lambda_2^{(i,j)} I \end{array} \right].$$

It follows that (2.2) is true iff each  $R^{(i,j)}$  is a linear combination of  $P$  and  $Q$ . Therefore, (2.2) holds if and only if each  $R^{(i,j)}$  has constant diagonal elements and constant off diagonal elements, which may depend on  $i$  and  $j$ . Thus, there are matrices  $A = (a_{ij})$  and  $B = (b_{ij})$  such that

$$\text{cov}(X_{ik}, X_{j\ell}) = \begin{cases} a_{ij} & \text{if } k = \ell \\ b_{ij} & \text{if } k \neq \ell \end{cases}$$

That is,

$$\text{var}(X_k) = A \quad \text{for all } k$$

and  $\text{cov}(X_k, X_\ell) = B \quad \text{if } k \neq \ell.$

Consequently,  $A$  and  $B$  are symmetric and we have established

Theorem 2: Let  $X_1, \dots, X_N$  be jointly normally distributed  $n$ -vectors whose joint distribution is a member of a family  $\mathcal{M}$ . Then the mean and scatter matrix of the  $X_i$ 's are sufficient for  $\mathcal{M}$  if and only if for each member of  $\mathcal{M}$ , (a) the  $X_i$ 's are identically distributed, and (b)  $\text{cov}(X_i, X_j)$  is independent of  $i$  and  $j$ .

#### 4. Conclusion:

As we mentioned in Section 1 if one thinks of a patch as an approximation to a field then it is difficult to understand how the within-patch covariance of

ORIGINAL PAGE IS  
OF POOR QUALITY

spectral measurements from a given patch could be constant but dependent on the patch size as in (c"). According to the results of Section 3, there is no more sophisticated covariance model whose parameters can be estimated with optimum efficiency using only the patch means and scatters; however, there may be more realistic covariance models which are well approximated by (c') or (c"). For example, suppose that a patch is rectangular in shape with multidimensional spectral information  $\{X_{ij} | i = 1 \dots r; j = 1 \dots c\}$  where  $i$  and  $j$  denote the spatial line and column number of the pixel producing  $X_{ij}$ . Suppose further that the correlation of two observations  $X_{ij}$  and  $X_{kl}$  decays exponentially with their spatial separation; that is,

$$\text{cov}(X_{ij}, X_{kl}) = \Omega^{1/2} A^{|i-k|} B^{|j-l|} \Omega^{1/2},$$

where  $\Omega$  is their common variance matrix and  $A$  and  $B$  are symmetric commuting matrices of spectral radius less than 1. Let  $\Sigma$  be the average covariance over all pairs of distinct pixels. Then a simple calculation shows that for large  $r$  and  $s$  (large patch size)  $r s \Sigma$  is nearly  $4 \Omega^{1/2} A(I-A)^{-1} B(I-B)^{-1} \Omega^{1/2}$ , so that  $\Sigma$  is nearly inversely proportional to the patch size, as is required by (c"). If  $A$  and  $B$  are positive semidefinite, so that  $z^T X_{ij}$  is always positively correlated with  $z^T X_{kl}$  for any  $z$ , then the expression just given is an upper bound for the average within-patch covariance for any patch size. Therefore, the effect of approximating the exponential covariance model with the constant covariance model (c") may be predictable, and not serious.

## References

1. Bahadur, R.R. (1954); Sufficiency and statistical decision functions. Ann. Math. Statist. 25, 423-463.
2. Dempster, A.P., Laird, N.M., and Rubin, D.B. (1977); Maximum likelihood from incomplete data via the EM algorithm. J. Roy. Statist. Soc. Ser. B 39, 1-38.
3. Feiveson, A.: private communication.
4. Giri, N.C. (1977); Multivariate Statistical Inference, Academic Press, New York.
5. Hall, W.J., Wijsman, R.A., and Ghosh, J.K. (1965); The relationship between sufficiency and invariance with application in sequential analysis, Ann. Math. Statist. 36, 575-614.
6. Peters, B.C., Decell, H.P., and Redner, R.A. (1978); Characterizations of linear sufficient statistics. Sankhya, Ser. A. 40, 303-309.
7. Peters, B.C., and Kampe, F. (1980); Numerical trials of HISSE. Report No. 75, Department of Mathematics, University of Houston, Contract NAS9-14689, August 1980.
8. Spivak, M. (1979); Differential Geometry, Vol. I, 2nd Ed., Publish or Perish Inc., Berkeley, Ca.

N88 12498

D4

1. Report No. SR-T1-04207		2. Government Accession No.		3. Recipient's Catalog No.	
4. Title and Subtitle The Akaike Information Criterion and its Application to Mixture Proportion Estimation				5. Report Date November 1981	
				6. Performing Organization Code	
7. Author(s) Richard A. Redner, Genshiro Kitagawa, William A. Coberly				8. Performing Organization Report No.	
9. Performing Organization Name and Address Division of Mathematical Sciences The University of Tulsa Tulsa, OK 74104				10. Work Unit No.	
				11. Contract or Grant No. NAS 9-14689-95	
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Lyndon B. Johnson Space Center Houston, TX				13. Type of Report and Period Covered	
				14. Sponsoring Agency Code	
15. Supplementary Notes					
16. Abstract  This report discusses the Akaike Information Criterion (AIC) with special emphasis on the application of the AIC to mixture models. The theory and applications of the AIC are discussed. Mixture model simulations and the application of the AIC to a large portion of a Landsat segment are presented.  ORIGINAL PAGE IS OF POOR QUALITY.					
17. Key Words (Suggested by Author(s)) Mixture estimation, Proportion Estimation, AIC, Clustering			18. Distribution Statement		
19. Security Classif. (of this report) Unclassified		20. Security Classif. (of this page) Unclassified		21. No. of Pages 54	22. Price*

\*For sale by the National Technical Information Service, Springfield, Virginia 22161

THE AKAIKE INFORMATION CRITERION AND ITS APPLICATION  
TO MIXTURE PROPORTION ESTIMATION

Richard A. Redner  
Genshiro Kitagawa\*

and

William A. Coberly

Division of Mathematical Sciences  
The University of Tulsa  
Tulsa, Oklahoma 74104

Report # SR-T1-04207

Prepared For

Earth Observation Division  
NASA/Johnson Space Center  
Houston, Texas  
Contract NAS-9-14689-95

September, 1981

\*Currently at the Bureau of the Census, Washington, D.C.

## Abstract

This report discusses the Akaike Information Criterion (AIC) with special emphasis on the application of the AIC to mixture models. The theory and applications of the AIC are discussed. Mixture model simulations and the application of the AIC to a large portion of a Landsat segment are presented.

## Table of Contents

Section	Page
1. Introduction . . . . .	1
2. Description of the AIC . . . . .	1
3. Derivation of the AIC . . . . .	6
4. Applications . . . . .	8
5. Mixture Models . . . . .	14
6. Mixture Simulations . . . . .	17
7. Concluding Remarks . . . . .	24
8. References . . . . .	26
Appendix	
A. Proofs of the approximations provided in section 3 . . . . .	A1
B. Figures for section 6 . . . . .	B1

## 1. INTRODUCTION

Estimation of parameters in a statistical model is a familiar and well discussed topic, but a more important topic, and certainly a more difficult one, is the selection of the appropriate model. The AIC (Akaike's Information Criterion) is a useful tool in model selection. It is particularly important in selecting the order of the model or in selecting the number of free parameters in a model. This report discusses the use of the AIC in selecting the order of a model and we emphasize the use of the AIC in determining the number of components in a mixture model.

After introducing appropriate notation in section 2, we show that the AIC is an extension of the maximum likelihood principle, as well as an entropy maximization principle. Section 3 will discuss the derivation of the AIC. In section 4, we give applications of the AIC to model selection in a number of important problems and we also introduce the BIC which is discussed in Hannan [1980]. Section 5 will discuss the application of the AIC to the mixture models. In section 6, we will look at the effectiveness of the AIC in dealing with the order selection problem on some 1 and 2 dimensional mixture problems.

## 2. DESCRIPTION OF THE AIC

We want to introduce the AIC as an extension of the maximum likelihood method. Let's begin by explaining why such an extension is needed. If we consider the case where the order of a particular model is determined, then the maximum likelihood

method is an excellent method for obtaining an estimate for the unknown parameters. The maximum likelihood estimate, under weak assumptions, is a strongly consistent, asymptotically unbiased and an asymptotically minimum variance estimator of the unknown parameters (Zacks [1971]). However, in the case that the order of the model is not known, the maximum likelihood estimator no longer has all of these desirable properties. The cause of this difficulty is that the maximum likelihood estimator has a preference for models of high order. As the order of the model is increased, the value of the maximum likelihood function, evaluated at the maximum likelihood estimate for that order model, is increased. Therefore, the maximum likelihood estimator will always have too many parameters.

The use of the maximum likelihood estimator to estimate the order of the model will lead to an estimate which fits the data very well (in fact too well), but will be a very poor estimator of the true density function. In section 4, we will use histograms as a concrete example of this problem.

As a possible replacement for the maximum likelihood estimator, we wish to consider an entropy maximization principle. This approach to the AIC was introduced and developed by Akaike [1972b, 1973, 1977]. It has also been supported by a Bayesian approach in Akaike [1978, 1979, 1980].

Let  $X$  be a random variable with density function  $g(x)$ . If  $f(x)$  is any other density function, we can define a measure of similarity of  $f$  and  $g$  by

$$B(g;f) \equiv E_x \log \frac{f(x)}{g(x)}$$

which equals  $\int \left\{ \log \frac{f(x)}{g(x)} \right\} g(x) dx$ , in the case of a continuous random variable  $x$ , and which equals  $\sum_{i=1}^p \log \left( \frac{f_i}{g_i} \right) g_i$ , in the discrete case. This measure is the entropy of  $f$  and  $g$  as defined by Kullback [1968]. It is non-positive and equals zero only in the case that  $f = g$  almost everywhere.

One interpretation of this, in the discrete case, is that for a sample of size  $N$ , the quantity  $N \cdot B(g;f)$  is approximately the logarithm of the probability of obtaining the data distribution  $g(x)$  from the assumed model  $f(x)$ .

Let  $f$ , a model for the data, be defined by

$$f(x) = \begin{cases} f_i & \alpha_i < x < \alpha_{i+1} \quad i = 1, \dots, p \\ 0 & \text{otherwise} \end{cases}$$

where  $\left( \sum_{i=1}^p f_i \right) (\alpha_{p+1} - \alpha_1) = 1$ . Given that we have  $N$  independent observations  $x_1, x_2, \dots, x_N$  from  $f$  we define  $N_i$   $i = 1, \dots, p$  to be the frequency of observations in the interval  $\alpha_i < x < \alpha_{i+1}$  and define relative frequencies  $g_i$   $i = 1, \dots, p$  by  $g_i = N_i/N$ .

The probability of observing the frequencies  $\{N_1, N_2, \dots, N_p\}$  from the model  $f$  is

$$w = \frac{N!}{N_1! N_2! \dots N_p!} f_1^{N_1} f_2^{N_2} \dots f_p^{N_p}.$$

From this we see that

$$\log w = \log N! - \sum_{i=1}^p \log N_i! + \sum_{i=1}^p N_i \log f_i$$

and using the fact that  $\log N! \approx n \log n - n$ , then

$$\log w \approx N \log N - N - \sum_{i=1}^p N_i \log N_i + \sum_{i=1}^p N_i + \sum_{i=1}^p N_i \log f_i$$

$$= -N \sum_{i=1}^p \frac{N_i}{N} \log \frac{N_i}{N} + N \sum_{i=1}^p \frac{N_i}{N} \log f_i$$

$$= N \sum_{i=1}^p g_i \log \frac{f_i}{g_i}$$

$$= N \cdot B(g;f).$$

ORIGINAL PAGE IS  
OF POOR QUALITY

So,  $N \cdot B(g;f)$  is approximately the logarithm of the probability of obtaining the distribution  $g$  from the assumed model  $f$ .

From a statistical inference point of view, we wish to find a model  $f(y)$  which will maximize the expected entropy

$$E_x B(g;f(\cdot, \theta(x))) = E_x E_y \log \left\{ \frac{f(y|\theta(x))}{g(y)} \right\}$$

and this idea is well motivated by example 1.

Let  $\ell(\hat{\theta})$  denote the value of the log likelihood function, evaluated at the maximum likelihood estimate  $\hat{\theta}$ ,

$$\ell(\hat{\theta}) = \frac{1}{N} \sum_{k=1}^N \log f(x_k, \hat{\theta}(x_1, \dots, x_N))$$

and let  $k$  denote the number of free parameters in the model. We define the AIC function by

$$\text{AIC} = -2\ell(\hat{\theta}) + \frac{2k}{N}.$$

The factor of  $-2$  is introduced for convenience, since in the normal case

$$-2 \log \exp\left(\frac{-(x-u)^2}{2\sigma^2}\right) = \frac{(x-u)^2}{\sigma^2}.$$

Observe, that

$$E_y \log \left\{ \frac{f(y)}{g(y)} \right\} = E_y \log f(y) + c \quad (1)$$

where  $c$  is independent of the choice of  $f$ . Also, it can be shown in many cases, that for  $\hat{\theta}$  the maximum likelihood estimate of  $\theta$ , we have

$$E_x E_y \log f(y, \hat{\theta}) \approx (\ell(\hat{\theta}) - K/N). \quad (2)$$

By using (1) and (2), we see that an estimator which minimizes the AIC, should approximately maximize the entropy function. The AIC estimate is a choice of parameters (which includes the choice of the number of parameter) which minimizes the AIC.

There is a relationship between the AIC and certain classical hypothesis tests. Let  $\hat{\theta}_m$  and  $\hat{\theta}_{m+k}$  be the maximum likelihood estimates for the  $m$  and  $m+k$  order models. Under certain regularity conditions we have that  $\ell(\hat{\theta}_{m+k}) - \ell(\hat{\theta}_m)$  is asymptotically  $\chi^2(k)$  (Rao [1973]). In the case that this holds, one can apply the Neyman-Pearson likelihood ratio test, and for a particular level of significance this would be equivalent to use of the AIC.

The Neyman-Pearson theory is designed to handle a particular type of composite hypothesis test and is not applicable to a variety of situations. In contrast to this, the AIC has wide applicability. It can be applied in the comparison of different types of models and can be applied when the Fisher Information matrix is singular. We consider the AIC as a simplification of the usual hypothesis testing approach to model building.

A second way to think about the AIC is as a penalized likelihood estimate. There have been a number of penalized approaches. Good and Gaskins [1971] and later Tapia [1978] introduced roughness penalties for estimators in infinite dimensional spaces. Redner [1980] and Rossback and Lennington [1978] discuss penalties on mixture models.

The AIC approach to the penalty term is appealing because of the simplicity and generality of the approach. However, the approximation in (2) is not valid for mixture models and we need to investigate this problem.

### 3. DERIVATION OF THE AIC

Let  $g(x)$  denote the true density function and let  $x_1, x_2, \dots, x_N$  be an independent sample from  $g(x)$ . Let  $f(x, \theta)$  be our model for  $g(x)$  and let  $\theta_0$  be the value of the parameter  $\theta$  such that  $f(x, \theta_0) = g(x)$ . We will let  $\ell(\theta)$  denote the log likelihood function and  $\hat{\theta}$  the maximum likelihood estimate. Finally we define

$$I(\theta_0) = E_x \left\{ \frac{\partial}{\partial \theta} \log f(x, \theta_0) \cdot \frac{\partial}{\partial \theta} \log f(x, \theta_0) \right\}$$

as the Fisher information matrix and

$$J(\theta_0) = -E_x \left\{ \frac{\partial^2}{\partial \theta \partial \theta} \log f(x, \theta_0) \right\}$$

as the negative of the expected value of the Hessian of the log likelihood function.

If we assume that  $I(\theta_0)$  is a nonsingular matrix, then under certain regularity conditions we observe that  $I(\theta_0) = J(\theta_0)$  and furthermore that

$$\ell(\hat{\theta}) - \frac{1}{2N} \text{tr}(J(\theta_0) I^{-1}(\theta_0))$$

is approximately an unbiased estimator for  $E_x \ell(\theta_0)$ . It also follows that

$$\ell(\hat{\theta}) - \frac{1}{N} \text{tr}(J(\theta_0) I^{-1}(\theta_0))$$

is approximately an unbiased estimator for the entropy. (See appendix 1.)

Since  $J(\theta_0) = I(\theta_0)$ , then  $k \equiv \text{tr}(J(\theta_0) I^{-1}(\theta_0))$  is the number of parameters in the model and so

$$E_x \ell(\theta_0) = E_x \ell(\hat{\theta}) - \frac{k}{2N}$$

and

$$-2E_y E_x \log f(y, \hat{\theta}(x)) = -2 E_x \ell(\hat{\theta}) + \frac{2k}{N}.$$

We have made two critical assumptions in these calculations and they are both quite significant. First, we have assumed that the Fisher information matrix is nonsingular. In the case that it is singular, then the statements above are no longer valid. What we can say is that

$$E_x \ell(\theta_0) \approx E_x \ell(\hat{\theta}) - \frac{1}{2N} \text{rank}(I(\theta_0))$$

$$\text{and } -2E_y E_x \log f(y, \hat{\theta}(x)) \approx -2E_x \ell(\hat{\theta}) + \frac{2}{N} \text{rank}(I(\theta_0)).$$

These facts will become important when we discuss finite mixtures, since in that case, the Fisher information matrix is often singular.

Now suppose that the true distribution of the data is not in the model. We distinguish between two different cases. Observe that the true distribution cannot be modelled if we use too few parameters, but this is not a problem. The maximum likelihood estimator rarely chooses a model with too few parameters, for these models do not fit the data. The problem is with eliminating models with too many parameters. But here we are concerned with models which do not contain the true distribution for any number of parameters. This is a case in which we almost always find ourselves. The model seldom (if ever) exactly fits the data. But this does not invalidate the use of the AIC.

Let us define a parameter  $\theta_0$  to be a choice of  $\theta$  which maximizes  $E_x \log f(x|\theta)$ . When the true distribution  $g(x)$  is in the model, then this implies  $f(x, \theta_0) = g(x)$ . Here we have only that

$$E_x \log f(x, \theta_0) = \max_{\theta} E_x \log f(x|\theta).$$

We then have that

$$E_x \log f(x|\theta_0) \approx E_x \ell(\hat{\theta}) - \frac{1}{2N} \text{tr}(J^{-1}(\theta_0) I(\theta_0))$$

$$\text{and so } E_x E_y \log f(y|\hat{\theta}(x)) \approx E_x \ell(\hat{\theta}) - \frac{1}{N} \text{tr}(J^{-1}(\theta_0) I(\theta_0)).$$

If the true distribution  $g(x)$  is close to  $f(x|\theta_0)$ , then  $J(\theta_0)$  will be close to  $I(\theta_0)$ , in which case,  $\text{tr}(J^{-1}(\theta_0) I(\theta_0)) \approx k$ . In this case, we use the AIC and expect the AIC to choose a good estimator of  $\theta_0$ .

If the true distribution is not well modelled by  $f(x|\theta)$ , then we should choose a better model, rather than alter the AIC.

#### 4. APPLICATIONS

The simplicity and generality of the AIC procedure makes it particularly useful. There are many areas in which the AIC can be used. We want to discuss a very simple application of the AIC to demonstrate its use, and hopefully, in this simple environment, we can better understand how the AIC functions and perhaps judge its effectiveness.

We consider the problem of determining the bin size for a histogram of univariate data, and we recall, that for a fixed bin size, the traditional height of the bins is a maximum likelihood estimate. Given independent observations  $x_1, \dots, x_N$ , and an interval of numbers  $[a, b]$ , we want to find the choice of bin size which minimizes the AIC. If the interval is divided into  $p$  bins and  $N_i$ ,  $i = 1, \dots, p$  is the number of observations in each bin, then the AIC equals (except for an additive constant)

$$-2 \left( \sum_{i=1}^p N_i \log N_i - N \log p \right) + 2(p - 1).$$

In some examples involving normally distributed observations, we observe that the AIC chooses a conservative number of bins. In 5 examples, using 100 independent normally distributed observations for each case, the AIC chose from 5 to 8 bins. This is a reasonable, if perhaps conservative, number of bins. We can see from the examples in Figure 1, that the AIC provides a number of bins which gives a smooth histogram. A larger number of bins could be used, but not too much larger. Figure 1 is typical of other examples.

When the sample size is increased, the number of bins chosen by the AIC is also increased. In each case, the resulting histogram is a smooth histogram (see Figure 2).

That the number of bins increases with the sample size, at first appears to be a problem. How can we choose the order of the model, if the AIC chooses to use more and more parameters as the sample size increases.

To understand this situation, it is necessary to distinguish two cases. The first case is that the true density is in the model for some finite number of parameters. The histogram is an example of a second type, where the true density is not in the model. The normal density function cannot be expressed as a histogram using a finite number of bins. Therefore, for any finite sample, the AIC chooses a small number of bins relative to the sample size. As the sample is increased, the AIC can reject histograms which have only a small number of bins and uses histograms with smaller mesh size.

ORIGINAL PAGE IS  
OF POOR QUALITY

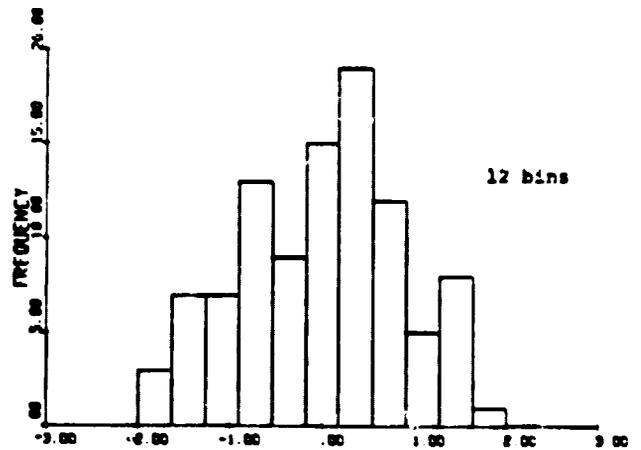
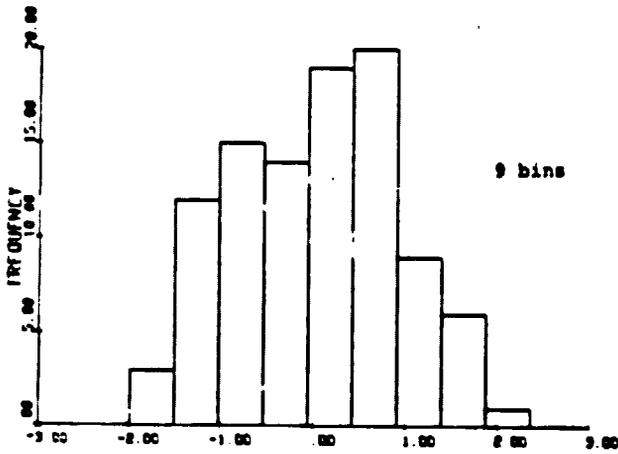
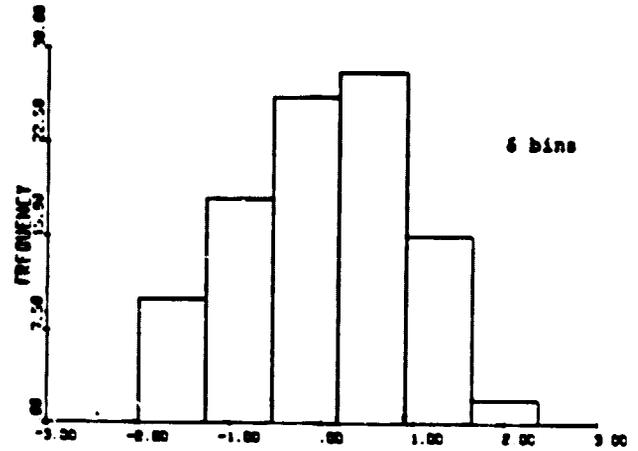
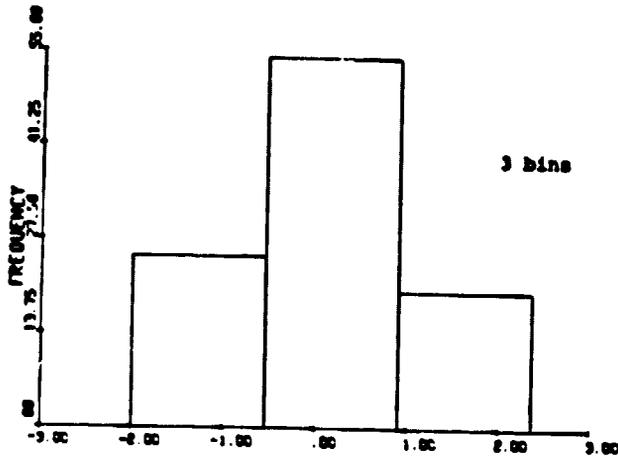


Figure 1. Graphs of histograms of normal data using different bin sizes.

ORIGINAL PAGE IS  
OF POOR QUALITY

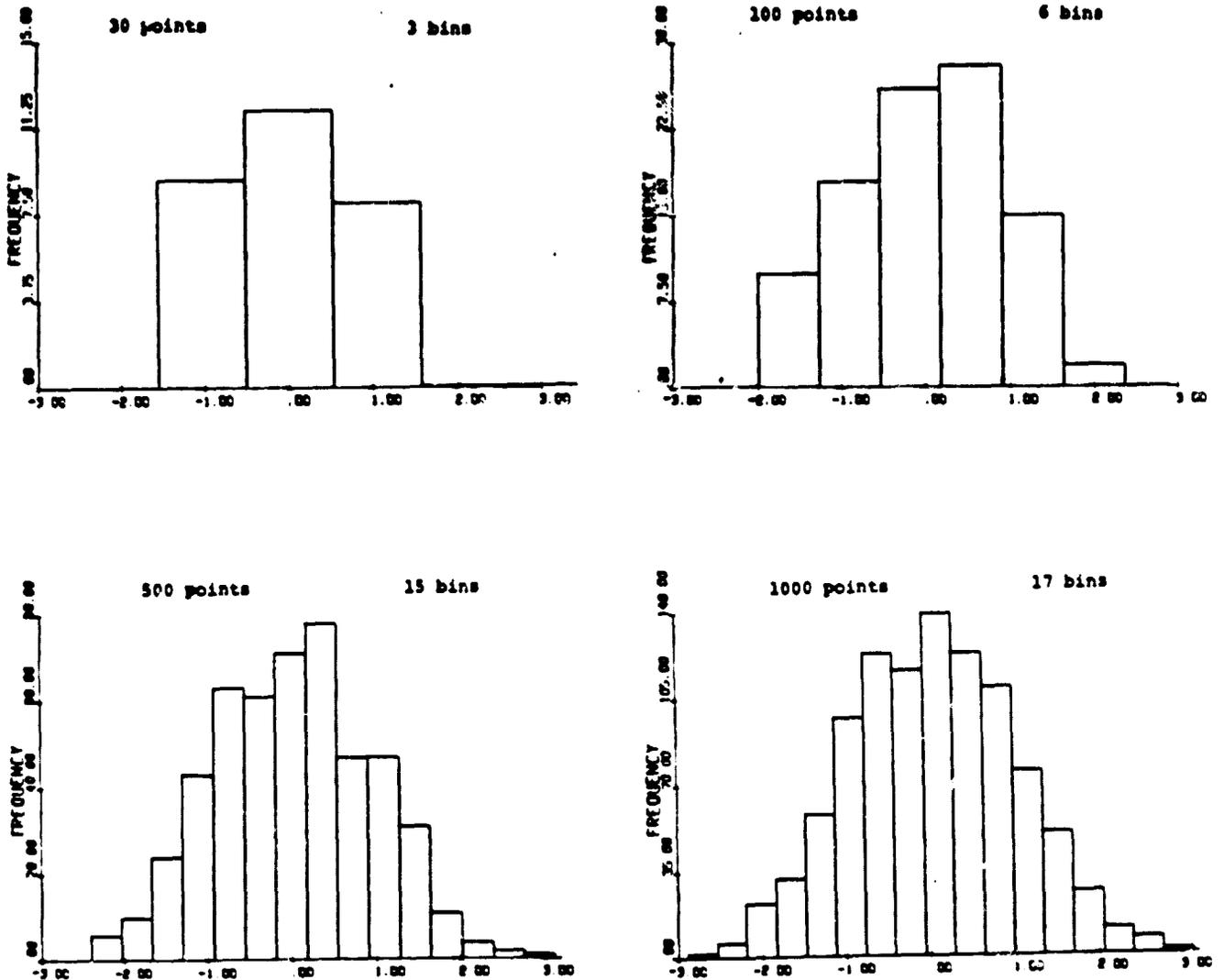


Figure 2. Graphs of histograms of normal data using AIC and different sample sizes.

The AIC tends to choose the correct number of parameters when the model matches the data. When the model does not match the data, the AIC chooses a reasonable number of parameters to fit the model, and this number increases with sample size.

We will return to this topic again when we discuss mixture models. But now let's consider other applications of the AIC. The theoretical arguments, might give the false impression that the AIC can only be used in a maximum likelihood density estimation setting, and we should dispense with that misconception at this time. While it is true that our theoretical justification of the AIC is based on the maximum likelihood density estimation situation, the AIC can be applied (with prudence) to many situations involving observations.

One of the first applications of the AIC was to time series analysis. Hipel [1981] gives a good set of references in this area and we will reproduce many of them here and add a few new entries. Autoregressive Moving Average (ARMA) process applications of the AIC have been presented by Akaike [1974], Ozaki [1977], and Lennox, McLeod and Hipel [1977a,b]. For the ARIMA process, see Ozake [1977]. Applications to the Autoregressive process were given by Akaike [1979] and Jones [1974]. Finally Kitagawa [1980] has applied the AIC to the difficult problem of modelling a time series which possesses a slowly changing spectrum.

There has been some recent work by Hannan [1980] on the estimation of the order of an ARMA process. In this paper, Hannan points out that the AIC is not a strongly consistent estimator of the order of an ARMA process. In fact it is not even a weakly

consistent estimator of the order. He defines two other criterion for estimating the number of parameters in an ARMA process. He considers the following measures. If  $p$  and  $q$  are the number of parameters in the AR and MA parts of the model, then

$$AIC = 2\ell(\hat{\theta}) + 2(p + q)$$

$$BIC = 2\ell(\hat{\theta}) + (p + q) \ln N$$

and

$$\vartheta(p, q) = 2\ell(\hat{\theta}) + (p + q) C \ln(\ln N) \quad C > 2.$$

Hannan shows that BIC and  $\vartheta(p, q)$  are strongly consistent.

While these are valuable theorems, they need to be properly understood. In the case that the model does not fit the true distribution exactly, the notion of 'correct order' becomes meaningless. What we want is a parsimonious use of parameters to obtain a reasonable fit to the data. This is our overall goal in many statistical settings. The AIC seems to perform very well in this environment and so should not be ruled out just because of these negative results. On the other hand, the AIC may provide too many parameters in large sample cases.

In Akaike [1973], the author considers factor analysis, principal component analysis, analysis of variance, and multiple regression, as other possible areas of application of the AIC. Kitagawa [1979] has used the AIC to detect outliers. Finally, choosing the order of a polynomial regression has been considered by Akaike [1972a] and Tanabe [1974].

The overall simplicity of the AIC makes it a valuable tool in model selection and helps integrate the model selection process into the estimation process. Because of these facts and the successful experience of many statistical investigators, the AIC appears to have a bright and useful future as a model selection tool.

## 5. MIXTURE MODELS

We will now focus our attention on the problem of estimating the number of classes in a mixture model. Let  $X_1, \dots, X_N$  be independent identically distributed observations from a mixture density. That is,  $X_1$  has density function

$$P(x|\theta^0) = \sum_{i=1}^{m^0} \alpha_i^0 P_i(x|\theta_i^0)$$

where  $\alpha_i^0 > 0 \quad i = 1, \dots, m^0$

$$\sum_{i=1}^m \alpha_i^0 = 1$$

and  $P_i(x|\theta_i)$  is a density function parameterized by  $\theta_i \in \Omega$ .

From the independent sample  $\{x_k\}_{k=1}^N$  on  $R^n$ , we wish to estimate the number of classes  $m$  and  $\theta = \{\alpha, \dots, \alpha_m, \theta_1, \dots, \theta_m\}$ .

Given a fixed value of  $m$ , we can obtain maximum likelihood estimates of the remaining parameters in the model using the maximum likelihood approach. We will not discuss this optimization problem here, but the reader is referred to a discussion of the EM algorithm by Redner [1980].

Once we have obtained a maximum likelihood estimate for two or more classes, we select as our estimate of the number of classes, a choice of  $m$  which minimizes

$$AIC(m) = -2\ell(\hat{\theta}_m) + 2 k_m$$

where  $k_m$  is the number of free parameters in the model with  $m$  classes.

The application of the AIC to mixture models is not as straightforward as many other applications. The difficulties arise in several ways and we will now discuss them.

Let us suppose that the true model has  $m^0$  classes. That is, the  $m^0$  component density functions are identifiable in the sense of Teicher [1963], and they have a positive probability of being observed in an independent sample. Under these conditions it is reasonable to assume that the Fisher information matrix is non-singular. Although this is not always the case, it is usually satisfied. Now let us consider the rank of the Fisher information matrix for the model with  $m^0 + 1$  classes, given that the true distribution has exactly  $m^0$  classes.

What is the rank of the Fisher information matrix? Unfortunately that is not a well defined question, for its solution depends on how the  $m^{\text{th}}$  order model is embedded into the  $m + 1^{\text{st}}$  order model.

Consider two alternative methods of embedding a one class model into a two class model and let  $\theta_1$  and  $\theta_2$  be one-dimensional parameters. The first alternative is that  $\theta_1$  equals  $\theta_2$ , and  $\alpha_1$  and  $\alpha_2$  are arbitrary. In this case, the rank of the Fisher information matrix is 1. On the other hand, if we use another embedding, say  $\alpha_2 = 0$  and  $\theta_2$  is arbitrary, then the rank of  $I(\theta_0)$  is 2. In either case, we are estimating 3 free parameters.

This problem is compounded by the fact, that in practice we do not know the true order of the model, and the possibility exists that the  $I(\theta_0)$  has full rank. In this case, it would be rank 3, if the two class model were actually correct.

Fortunately we can use the AIC as it stands and not worry about the rank of  $I(\theta_0)$ . Since its rank is not larger than the number of free parameters (since the number of free parameters is the dimension of the matrix) we use the AIC as stated.

We minimize

$$AIC(m) = -2l(\hat{\theta}) + 2k_m$$

where  $k_m$  is the number of free parameters in the  $m$  class model.

We propose to use the AIC even in the case that true distribution is not in the model. We have seen with histograms that the AIC is still an effective tool in selecting the order of a model and small changes in the data from the model should not have a strong effect on the performance of the AIC.

We will discuss some simulations that were performed using the AIC in the next section, but now we wish to consider other uses of the AIC in mixture problems. Let us consider the mixture of several multivariate normal densities. That is each  $n$ -dimensional observation has distribution

$$P(x) = \sum_{i=1}^m \alpha_i P_i(x, \mu_i, \Omega_i)$$

$$\text{where } P_i(x, \mu_i, \Omega_i) = \frac{1}{\sqrt{2\pi}^n} \frac{1}{|\Omega_i|^{1/2}} e^{-\frac{1}{2}(x - \mu_i)^t \Omega_i^{-1}(x - \mu_i)}$$

$$\text{and } \sum_{i=1}^m \alpha_i = 1 \text{ and } \alpha_i > 0 \quad i = 1, \dots, m.$$

The parameters in the model are  $\{\alpha_i, \mu_i, \Omega_i\}_{i=1}^m$  and the number of mixing components  $m$ . We may add the additional assumption that, although the covariances are unknown, they are assumed to be equal.

Choosing between the free covariance model, and the unknown but equal covariance model, is a problem in choosing the order of the model. The number of parameters to be estimated in estimating

the covariance for an n-dimensional multivariate normal density is  $\frac{n(n+1)}{2}$ . We observe that the constraint requiring the covariance to be positive definite does not affect the number of free parameters, since given any positive definite matrix, all the parameters can be varied arbitrarily by some small amount without change to the positive definiteness.

If we have a model with m classes and n dimensions, then the number of free parameters in the m covariances is  $k = \frac{m(n)(n+1)}{2}$ , unless the covariances are assumed to be equal. Then we have that  $k = \frac{n(n+1)}{2}$ .

The AIC can be used to determine which of these two models should be used on any given data set.

## 6. MIXTURE SIMULATIONS

In order to understand the application of the AIC to mixture models, we have performed several simulations. The simulations in one dimension were designed to analyze the performance of the AIC as a function of class separation. The simulations were performed with relatively small data sets and we have observed that this causes the AIC to choose a conservative number of classes. This is similar to what was discovered in the histogram application.

The data which we generated was a mixture of two normal densities. The mixing proportions were equal and the true covariances were set equal to one. The only parameters which we varied were the sample size and the mean values. All of the parameters in the model were estimated. The covariances were estimated using the assumption they were equal but unknown. The tabulated results are in Figure 3 (all of the figures for this section are in appendix B).

From these data sets, we observe that the performance of the AIC is indeed a function of the class separation. When the classes are well separated (mean values 3 standard deviations apart) the AIC unerringly chose the correct number of classes. On the other hand, the AIC always chose the one class model when the class separations were small (mean values were less than or equal to 1 standard deviation apart). The AIC performed well for 2 units of separation.

To test the limits of the AIC, we considered another simulation. This simulation was composed of 10 repetitions of the one standard deviation separation with 300 observations. The results are contained in Table 1. We can observe two things in this table. Obviously the AIC consistently chose too few classes for the model. In fact, 8 out of 10 times too few classes were chosen, and the three class model was never chosen. The histograms and estimated density functions for the first two runs are in Figure 4.

One can estimate the performance of the AIC for larger sample sizes by considering Table 1. It appears that for sample sizes in the range of 2000 data points the AIC would choose the correct model about one-half of the time.

Our final calculations in one dimension involve Landsat data from one scan line of segment 1618. We use the AIC to estimate the number of classes in the model and the resulting answer was three classes. On consulting ground truth, we see that 88 percent of the data lie in three ground truth classes, and no other ground truth class comprised more than 5 percent of the scan line. The results for channel three and the brightness component are presented in Figures 5 and 6.

Table 1. AIC values for 10 simulations.

RUN NUMBER	No. OF CLASSES		
	1	2	3
1	911.0	914.2	918.0
2	* 918.0	915.1	918.0
3	924.8	927.0	931.0
4	931.4	935.4	939.2
5	* 894.6	891.0	892.2
6	890.6	894.6	898.6
7	933.2	936.0	938.8
8	924.2	927.2	931.0
9	886.6	888.4	892.2
10	907.0	911.0	914.8

The overall effect of these data sets is to show that the AIC chooses a reasonable number of classes, considering the sample size.

As a second stage in our investigation of the application of the AIC to mixtures, we investigated mixtures of bivariate normal density functions. These investigations consisted of simulations with three and five bivariate normal densities at several different separations, and also, the estimation of the number of normal classes in a portion of Landsat segment.

We initially investigated a mixture of three normal classes at various separation. For simplicity of the example, we generated classes which had equal probabilities and which each had the identity matrix as its covariance matrix. As in the one-dimensional examples, we varied the means in the simulations and estimated the covariances

under the hypothesis that they were equal but unknown. In Figures 7, 8 and 9, the true values of the means, the values of the AIC for the different models, and a graph of the true density function are displayed. From the table of AIC values in each of the three examples, one can see that the AIC chose the correct number of classes except in Case III where the true mean separation was small and Case IC. Although the classes are well separated in Case IC the AIC chose the fourth order model. This type of error is to be expected to happen a certain percentage of the time.

We extended our investigation to mixtures of five normal classes. Again the proportions of the classes were equal and the covariance of the five classes were equal to the identity matrix. In Figures 10 and 11, we see the true mean values, the values of the AIC for the different models and graphs of the true density function. One thousand sample points were used since we must estimate 17 parameters under the assumption that the covariances are unknown but are equal. Again we see that the AIC chose the correct number of classes when the populations are well separated and chose too few classes in the case that there is considerable overlap between classes.

Let us now consider a more realistic data set. We take a Landsat segment for this purpose and we selected segment 1633 for this test. Using ground truth data, we selected the pure elements of the scene. We define a pixel to be a pure pixel, if it has neighbors which are all of the same ground truth class. The data set was reduced in dimension by the use of the Kauth transformation to the greenness and brightness plane. We used the first 20 percent of this two dimensional data set to form our working data set.

We modelled this data set, which comprises 1966 data points, by a mixture of bivariate normal density functions. During some of our calculations of the maximum likelihood estimate, we observed that the maximum likelihood iteration was proceeding to a singularity of the likelihood function. This not only causes numerical problems but completely invalidates the use of the AIC to determine the number of classes. Singularities of the likelihood function must be avoided. There are several methods for avoiding the singularities of the likelihood function. The method which we implemented was an application of a penalty term. This penalty term forces the likelihood iteration to avoid the singularities of the likelihood function (see Redner (1980)).

This type of adjustment to the natural maximum likelihood iteration is often necessary. Since the data is discrete, that is the data takes on only integer values, this is a common problem.

With this adjustment, the maximum likelihood estimate for various numbers of classes was calculated. Table 2 contains the AIC values. It was not possible to completely determine the correct number of classes using the AIC due to limits on machine time. But the reader can observe from these numbers that the number of classes is quite large. With 15 classes in the model, the maximum likelihood estimate for each parameter is based on approximately 22 data points. Furthermore it appears that the true number of classes, as determined by the AIC, might be considerably larger than 15. This is unacceptable for the type of application for which this is intended. One would expect that the number of classes chosen by the AIC for a full Landsat segment would be much larger (perhaps two to four times as large).

Table 2. AIC values for Landsat Segment

NO. OF CLASSES	AIC VALUES	BIC VALUES
7	33194	33423
8	33130	33392
9	33126	33422
10	33078	33407
11	33078	33441
15	32954	33511

This brings us back to our previous remarks concerning the consistency of the AIC and the tendency for the AIC to choose a large number of classes, when the true model is not a good approximation to the data. Because of the poor showing of the AIC in this example we extended the experiment to consider the BIC. The results of these calculations are also in Table 2. From these numbers, one can see that the trend of the AIC to choose a large number of classes, is not reflected in this table of BIC values. In fact, the choice of 8 classes by the BIC, appears to be a much better value for the types of applications for which the model is intended. Figure 12 contains the maximum likelihood estimates for the model with 8 classes and also contains the scatter plot of the 1966 data points.

In response to the poor showing of the AIC, we considered one final experiment. This estimate of the parameters and the number of classes proceeded in a three step process.

First the maximum likelihood estimate for models with different numbers of classes was calculated under the common covariance assumption. The maximization of the likelihood function with the equal

covariance hypothesis is a very stable optimization problem. It also has fewer local maxima and so it is much easier to find the global maxima.

The second phase of the procedure involved determination of the number of classes to use in the model using the AIC. In this case, the AIC chose 7 classes for the model. Table 3 contains the AIC values.

Since it appears that the common covariance assumption is not valid for Landsat data, in the final step we fixed the mean values and iterated on the proportions and the covariance. The covariances being allowed to vary independently. This provides a significant improvement to the fit to the data according to the AIC. The new AIC value is 33194.

Although the AIC has performed well in numerous applications, we observe, in the application of the AIC directly to a large portion of a Landsat segment, that it provides a model with a large number of classes. The cause of this problem is probably the unboundedness of the likelihood function. The use of the penalty term was not sufficient to completely rectify this situation. We should emphasize that it is the type of application which

Table 3. AIC values for the equal covariance model

No. OF CLASSES	AIC
6	33546
7	33276
8	33280
10	33292

we have in mind and the large amount of computation which has to be done which causes us to conclude that the answer given by the direct application of the AIC, is not suitable. In addition, the ratio of the number of parameters being estimated and the number of data points available is not particularly large. All of these considerations lead us to consider possible alternatives. The most natural alteration of the AIC is the BIC, which gives us a more desirable number of classes along with some indication that it might consistently give the correct number of classes when the model is correct and when we have large data sets. The other possibility is given by the three step application of the AIC to the mixture problem. This approach is appealing because of the stability of the natural fixed point iteration if the covariances are assumed to be unknown but equal.

## 7. CONCLUDING REMARKS

The AIC has shown to be effective in a wide range of applications. These demonstrations now include the mixture density problems. For some data sets it appears that the BIC may provide more useful results than the direct application of the AIC. The authors are optimistic about the possible uses of the AIC and BIC in determining the number of components in a mixture model and in determining which of several mixture models to use.

On the other hand, we do not consider the simulations and examples presented in this paper as sufficient proof of the applicability of the AIC or BIC in model selection for the mixture problem. In particular the AIC and BIC have not been applied to a full Landsat segment and certainly many segments must be considered before a judgement on these criterion can be made.

It is hoped that experiments along these lines will be carried out in the near future. The application of the AIC and BIC to the HISSE procedure is a particular experiment that should be carried out as well as the application of these criterion to MLE methods applied to profile data.

There are numerous other areas which need to be considered in the application of the AIC and BIC to mixture models. The mechanics of applying the AIC and BIC to mixture models needs to be considered further. Since we are dealing with expensive non-linear optimization problems to obtain the likelihood estimates, we must consider the best way to find the AIC or BIC estimates. The suggestion by Wolf (1970) may be particularly applicable to this area. Wolf suggests the use of certain non-parametric clustering schemes to assist in obtaining initial guesses of the MLE.

Finally we reiterate that the AIC and BIC can be used in a wide range of applications. We have emphasized the use of the AIC and BIC in model selection for mixture problems because that is a problem in which we have a deep interest. However, the use of the AIC and BIC in other areas should not be neglected and it is hoped that the applications which we suggested in section 4 might lead to other uses of these criterion.

## 8. REFERENCES

1. Akaike, H. (1972a) "Automatic data structure by maximum likelihood," in Proc. 5th Hawaii Int. Conf. on Sys. Sci., Western Periodicals, North Hollywood, CA, pp. 99-101.
2. Akaike, H. (1972b) "Use of an information theoretic quantity for statistical model identification," in Proc. 5th Hawaii Int. Conf. Sys Sci., Western Periodicals, North Hollywood, CA, pp. 242-250.
3. Akaike, H. (1973) "Information theory and an extension of the maximum likelihood principle," in Proc. 2nd Int. Symp. Inform. Theory, Budapest, pp. 267-281.
4. Akaike, H. (1974) "A new look at statistical model identification," IEEE Trans. Automatic Control, AC-19, pp. 716-723.
5. Akaike, H. (1977) "On entropy maximization principle," in P.R. Krishnaiah ed., Applications of Statistics, North-Holland, Amerstram.
6. Akaike, H. (1978) "A Bayesian analysis of the minimum AIC procedure, Annals of the Inst. of Stat. Math., 30A, 9-14.
7. Akaike, H. (1979) "A Bayesian extension of the minimum AIC procedure of autoregressive model fitting, Biometrika, 66, pp. 237-242.
8. Akaike, H. (1980) "Likelihood of a mode and information criterion," The Inst. of Stat. Math., Res. Memo 183.
9. Good, I.J. and Gaskins, R.A. (1971) "Nonparametric roughness penalties for probability densities," Biometrika, 58, pp. 255-277.
10. Hannan, E.J. (1980) "The estimation of the order of an ARMA process," The Annals of Statistics, 8, pp. 1071-1081.
11. Hipel, K.W. (1981) "Geophysical model discrimination using the Akaike information criterion," IEEE Trans. on Automatic Control, 26, pp. 358-378.
12. Jones, R.H. (1974) "Identification and autoregressive spectrum estimation," IEEE Trans. Automatic Control, AC-19, pp. 894-897.
13. Kitagawa, G. (1979) "On the use of AIC for detection of outliers," Technometrics, 21, pp. 193-199.
14. Kitagawa, G. (1980) "Changing spectrum estimation," The Inst. of Stat. Math., Res. Memo 194.
15. Kullback, S. (1968) "Information Theory and Statistics," New York, Dover Publications.

16. Lennox, W.C., Hipel, K.W. and McLeod, A.I. (1977a) "Advances in Box-Jenkins modelling, 1, applications," Water Resources Res., 13, pp. 567-575.
17. Lennox, W.C., Hipel, K.W. and McLeod, A.I. (1977b) "Advances in Box-Jenkins modeling, 2, applications," Water Resources Res., 19, pp. 577-586.
18. Ozaki, T. (1977) "On the order determination of ARIMA models," J Royal Stat. Soc., Series C, 26, pp. 290-301.
19. Rao, C.R. (1973) "Linear Statistical Inference and Its Applications," John Wiley and Sons, Inc., New York.
20. Redner, R.A. (1980) "Maximum likelihood estimation for mixture models," NASA Technical Report No. SR-J0-04007; JSC-16832.
21. Rossback, M.E. and Lennington, R.K. (1978) "Classy-an adaptive maximum likelihood clustering algorithm," Ninth Annual Meeting of the Classification Society.
22. Tanabe, K. (1974) "Fitting regression curves and surfaces by Akaike's information criterion," Inst. of Stat. Math, Tokyo, Res. Memo 63.
23. Tapia, R.A. and Thompson, J.R. (1978) "Nonparametric probability density estimation," John Hopkins Univ. Press, Baltimore.
24. Teicher, H. (1963) "Identifiability of finite mixtures," Annals of Math. Stat., 34, pp. 1265-1269.
25. Wolfe, J.H. (1970) "Pattern clustering by multivariate mixture analysis," Multivariate Behavioral Research, 5, pp. 329-350.
26. Zacks, S. (1971) "The Theory of Statistical Inference," John Wiley and Sons, Inc., New York.

**APPENDIX A**

**Proofs of the approximations provided  
in section 3.**

ORIGINAL PAGE IS  
OF POOR QUALITY

Let  $g(x)$  denote the true density function for a random variable  $X$  and let  $f(x, \theta)$  be a model for  $g(x)$ . Let  $\theta_0$  be a choice of parameters  $\theta$  so that

$$E_x \log f(x, \theta_0) = \max_{\theta} E_x \log f(x, \theta).$$

In the case that the true density  $g(x)$  is in the model, then this implies that  $f(x, \theta_0) = g(x)$ .

We also define

$$I(\theta_0) = E_x \left\{ \frac{\partial}{\partial \theta} \log f(x, \theta_0) \cdot \frac{\partial}{\partial \theta} \log f(x, \theta_0) \right\}$$

to be the Fisher information matrix and define

$$J(\theta_0) = - E_x \left\{ \frac{\partial^2}{\partial \theta \partial \theta'} \log f(x, \theta_0) \right\}$$

to be the negative of the expected Hessian of the log likelihood function. Let  $\hat{\theta}$  be the maximum likelihood estimate where  $\hat{\theta}$  is shorthand for  $\hat{\theta}(x_1, \dots, x_N)$ , a function of the observations. Finally we define

$$\ell(\theta) = \frac{1}{N} \sum_{k=1}^N \log f(x_k, \theta)$$

to be the log likelihood function.

Assuming the true density function  $g(x)$  to be in the model and assuming the necessary regularity conditions, we will show that

$$E_x \ell(\theta_0) \sim E_x \ell(\hat{\theta}) - \frac{1}{2N} \text{rank}(I(\theta_0)) \quad (1)$$

$$\text{and } -2 E_y E_x \log f(y, \hat{\theta}(x)) \sim -2 E_x \ell(\hat{\theta}) + \frac{2}{N} \text{rank}(I(\theta_0)) \quad (2)$$

Later on we will show that, in the case that  $g(x)$  is not in the model and if  $J(\theta_0)$  and  $I(\theta_0)$  are nonsingular, then we have that

$$E_x \ell(\theta_0) \sim E_x \ell(\hat{\theta}) - \frac{1}{2N} \text{tr} \{ J(\theta_0)^{-1} I(\theta_0) \} \quad (3)$$

$$\text{and } -2 E_y E_x \log f(y, \hat{\theta}(x)) \sim -2 E_x \ell(\hat{\theta}) + \frac{2}{N} \text{tr} \{ J(\theta_0)^{-1} I(\theta_0) \}. \quad (4)$$

ORIGINAL PAGE IS  
OF POOR QUALITY

Recall that by the law of large numbers we have

$$\frac{1}{N} \sum_{k=1}^N \frac{\partial^2}{\partial \theta \partial \theta'} \log f(x_k, \theta_0) \rightarrow -J(\theta_0). \quad (5)$$

Using the central limit theorem we can observe that

$$\frac{1}{\sqrt{N}} \sum_{k=1}^N \frac{\partial}{\partial \theta} \log f(x_k, \theta_0) \rightarrow N(0, I(\theta_0)). \quad (6)$$

From the following equation

$$\begin{aligned} \frac{\partial}{\partial \theta} \ell(\hat{\theta}) &= 0 = \frac{1}{N} \sum_{k=1}^N \frac{\partial}{\partial \theta} \log f(x_k, \hat{\theta}) \\ &\sim \frac{1}{N} \sum_{k=1}^N \frac{\partial}{\partial \theta} \log f(x_k, \theta_0) + \frac{1}{N} \sum_{k=1}^N \frac{\partial^2}{\partial \theta \partial \theta'} \log f(x_k, \theta_0) (\hat{\theta} - \theta_0) \end{aligned}$$

we calculate, using (5) and (6), that

$$\sqrt{N} J(\theta_0) (\hat{\theta} - \theta_0) \sim N(0, I(\theta_0)).$$

If  $J(\theta_0)$  (respectively  $I(\theta_0)$ ) is a singular matrix, then we have a singular normal density. In this case let  $P: R^m \rightarrow R^n$  for  $m > n$  be a matrix that  $P^T P$  is the  $n \times n$  identity matrix and  $P P^T$  is the projection onto the range of  $J(\theta_0)$ . Then

$$\sqrt{N} P J(\theta_0) (\hat{\theta} - \theta_0) \sim N(0, P I(\theta_0) P^T)$$

where  $P I(\theta_0) P^T = P J(\theta_0) P^T$  is nonsingular. From the definition of  $P$ ,

$$\sqrt{N} P J(\theta_0) P^T P (\hat{\theta} - \theta_0) \sim N(0, P I(\theta_0) P^T)$$

and so

$$\sqrt{N} P (\hat{\theta} - \theta) \sim N(0, (P J(\theta_0) P^T)^{-1} (P I(\theta_0) P^T) (P J(\theta_0) P^T)^{-1}) \quad (7)$$

Now observe that

$$\begin{aligned} \frac{1}{N} \sum_{k=1}^N \log f(x_k, \theta_0) &\sim \frac{1}{N} \sum_{k=1}^N \log f(x_k, \hat{\theta}) \\ &\quad - \frac{1}{2} (\theta_0 - \hat{\theta}) J(\theta_0) (\theta_0 - \hat{\theta}) \end{aligned}$$

and so by taking expectations

$$\begin{aligned} E_X \ell(\theta_0) &\sim E_X \ell(\hat{\theta}) - \frac{1}{2} E_X (\theta_0 - \hat{\theta})^T J(\theta_0) (\theta_0 - \hat{\theta}) \\ &= E_X \ell(\hat{\theta}) - \frac{1}{2} E_X (\theta_0 - \hat{\theta})^T P^T P J(\theta_0) P^T P (\theta_0 - \hat{\theta}) \\ &= E_X \ell(\hat{\theta}) - \frac{1}{2N} E_X \text{tr} [(PJ(\theta_0)P^T) (P(\theta_0 - \hat{\theta}) (\theta_0 - \hat{\theta})^T P^T)] \end{aligned}$$

Therefore,

$$\begin{aligned} E_X \ell(\theta_0) &\sim E_X \ell(\hat{\theta}) - \frac{1}{2N} \text{tr} [(PJ(\theta_0)P^T) (PJ(\theta_0)P^T)^{-1} \\ &\quad (PI(\theta_0)P^T) (PJ(\theta_0)P^T)^{-1}] \quad (8) \\ &= E_X \ell(\hat{\theta}) - \frac{1}{2N} \text{rank} (I(\theta_0)) \end{aligned}$$

since  $J(\theta_0) = I(\theta_0)$ .

We see that  $\ell(\hat{\theta})$  is a biased estimate of  $\ell(\theta_0)$  and this bias is on the average equal to  $\frac{1}{2N} \text{rank} I(\theta_0)$ .

We now proceed to establishing (2). First observe that

$$\begin{aligned} E_Y \log f(y, \hat{\theta}) &= \int \log f(y|\hat{\theta}) g(y) dy \\ &\sim \int \log f(y, \theta_0) g(y) dy + \int \frac{\partial}{\partial \theta} \{\log f(y, \theta_0)\} g(y) dy (\hat{\theta} - \theta_0) \\ &\quad + \frac{1}{2} (\hat{\theta} - \theta_0)^T \int \frac{\partial^2}{\partial \theta \partial \theta'} \{\log f(y, \theta_0)\} g(y) dy (\hat{\theta} - \theta_0) \\ &= E_Y \log f(y, \theta_0) - \frac{1}{2} (\hat{\theta} - \theta_0)^T J(\theta_0) (\hat{\theta} - \theta_0) \\ &= E_Y \log f(y, \theta_0) - \frac{1}{2} (\hat{\theta} - \theta_0)^T P^T P J(\theta_0) P^T P (\hat{\theta} - \theta_0) \end{aligned}$$

Taking expectations we find that

$$\begin{aligned} E_X E_Y \log f(y, \hat{\theta}) &\sim E_X E_Y \log f(y, \theta_0) \\ &\quad - \frac{1}{2N} \text{tr} ((PJ(\theta_0)P^T)^{-1} (PI(\theta_0)P^T)) \end{aligned}$$

ORIGINAL PAGE IS  
OF POOR QUALITY

$$= E_x E_y \log f(y, \theta_0) - \frac{1}{2N} \text{rank } I(\theta_0).$$

$$= E_y \log f(y, \theta_0) - \frac{1}{2N} \text{rank } I(\theta_0).$$

We observe that equation (1) is equivalent to

$$E_y \log f(y, \theta_0) \sim E_x \ell(\hat{\theta}) - \frac{1}{2N} \text{rank } I(\theta_0)$$

Putting these two estimates together

$$E_x E_y \log f(y, \hat{\theta}) \sim E_x \ell(\hat{\theta}) - \frac{1}{N} \text{rank } I(\theta_0)$$

which establishes (2).

Let us now consider the case that  $g(x)$  is not in the model and so  $\theta_0$  is a choice of parameters which maximize  $E_x \log f(x, \theta)$ . Let us also assume that the Fisher information matrix  $I(\theta_0)$  is nonsingular.

Observe that equation (6) still holds and we have immediately that

$$E_x \ell(\theta_0) \sim E_x \ell(\hat{\theta}) - \frac{1}{2N} \text{tr}\{(PJ(\theta_0)P^T)^{-1}(PI(\theta_0)P^T)\}$$

If  $J(\theta_0)$  is nonsingular then

$$E_x \ell(\theta_0) \sim E_x \ell(\hat{\theta}) - \frac{1}{2N} \text{tr}(J(\theta_0)^{-1}I(\theta_0))$$

and this gives us equation (3).

We can also observe the (7) holds, that is

$$E_x E_y \log f(y, \hat{\theta}) \sim E_x E_y \log f(y, \theta_0) \\ - \frac{1}{2N} \text{tr}((PJ(\theta_0)P^T)^{-1}(PI(\theta_0)P^T)^{-1})$$

and also

$$E_x E_y \log f(y, \theta_0) \sim E_x \ell(\hat{\theta}) - \frac{1}{2N} \text{tr}((PJ(\theta_0)P^T)^{-1}(PI(\theta_0)P^T)^{-1}).$$

Combining these facts one gets that

$$E_x E_y \log f(y, \hat{\theta}) \sim E_x \ell(\hat{\theta}) - \frac{1}{N} \text{tr} (PJ(\theta_0)P^T)^{-1} (PI(\theta_0)P^T)$$

Again if  $J$  is nonsingular, then  $P = I$  and

$$E_x E_y \log f(y, \hat{\theta}) \sim E_x \ell(\hat{\theta}) - \frac{1}{N} \text{tr} J(\theta_0)^{-1} I(\theta_0)$$

and this establishes our final result.

ORIGINAL PAGE IS  
OF POOR QUALITY

**APPENDIX B**

**Figures from section 6**

100 DATA POINTS

$\Delta = u_1 - u_2$

No. OF CLASSES	.5	1.0	2.0	3.0
1	295.0	322.2	352.0	407.2
2	295.4	321.6	340.0	402.6
3	299.0	325.4	343.0	415.6

150 DATA POINTS

$\Delta = u_1 - u_2$

No. OF CLASSES	.5	1.0	2.0	3.0
1	273.4	305.4	330.8	418.2
2	277.4	305.8	331.2	406.6
3	280.8	307.8	335.2	410.6

300 DATA POINTS

$\Delta = u_1 - u_2$

No. OF CLASSES	.5	1.0	2.0	3.0
1	871.2	886.6	1040.6	1213.2
2	875.2	885.4	1033.0	1171.8
3	878.6	892.2	1042.0	1175.8

300 DATA POINTS

$\Delta = u_1 - u_2$

No. OF CLASSES	.5	1.0	2.0	3.0
1	893.2	907.0	1052.6	1192.6
2	897.2	911.0	1034.0	1156.6
3	908.2	914.8	1038.0	1160.4

Figure 3. Tables of AIC values for 4 simulations. Each table contains AIC values for varying numbers of classes and class separation.

Run 1



Run 2

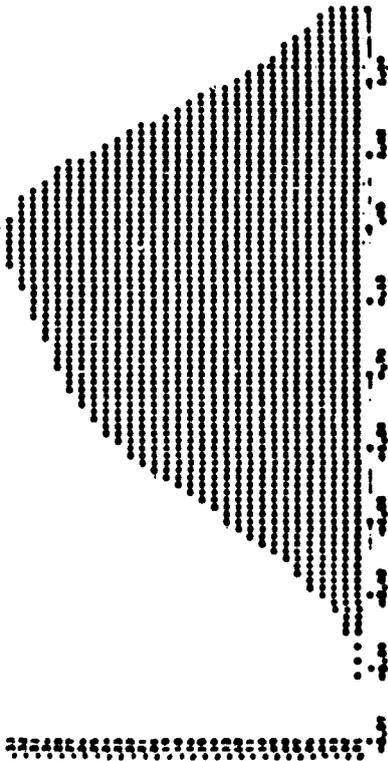
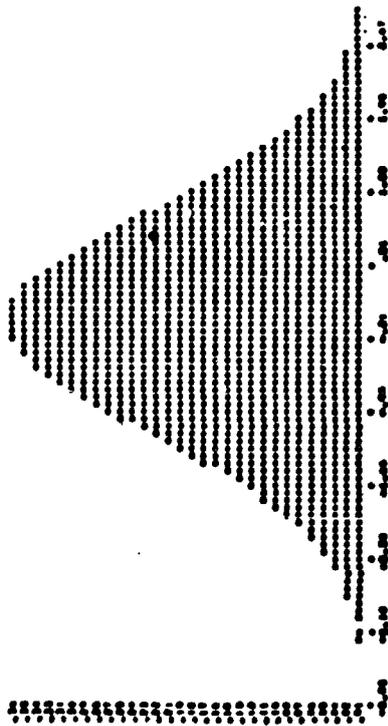
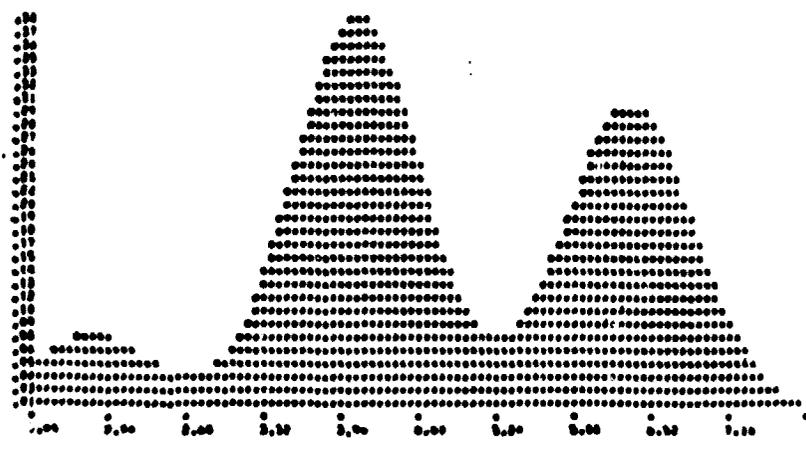


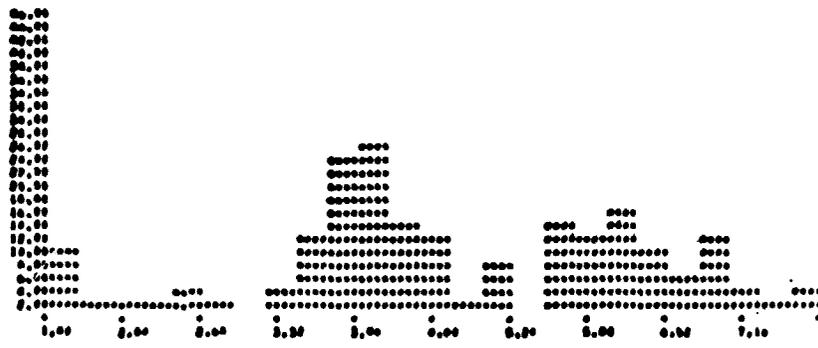
Figure 4. Graphs of histograms and estimated density functions for runs number one and two in table 1.

ORIGINAL PAGE IS  
OF POOR QUALITY

ESTIMATED DENSITY FUNCTION WITH 3 CLASSES



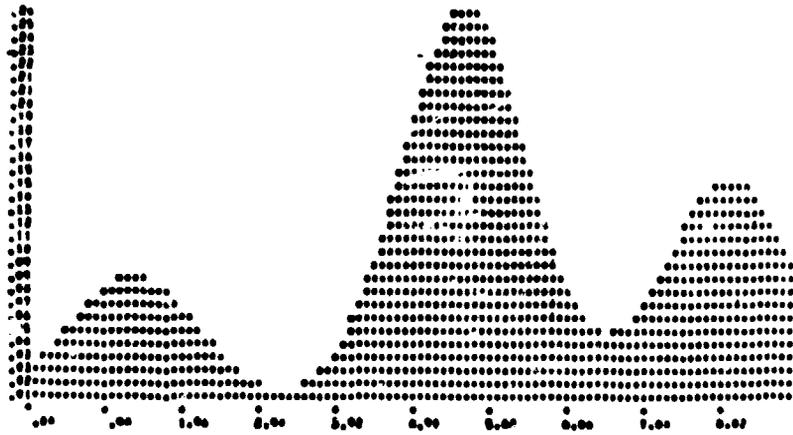
CHANNEL 3 HISTOGRAM



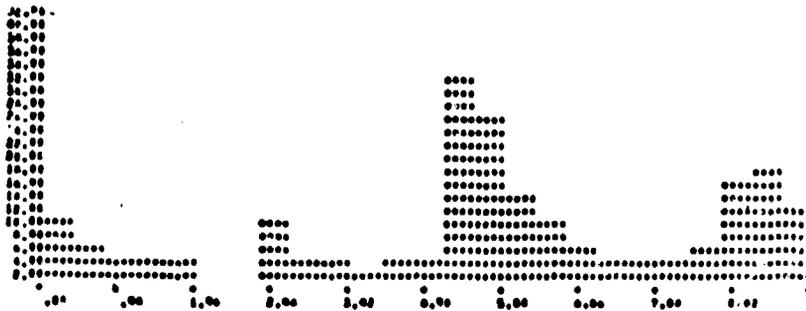
No. of CLASSES	AIC VALUES
1	721.3
2	725.3
3	662.9
4	666.9

Figure 5. Graphs and AIC values for segment 1618/235 line 62 channel 3.

ESTIMATED DENSITY FUNCTION WITH 3 CLASSES



HISTOGRAM OF BRIGHTNESS



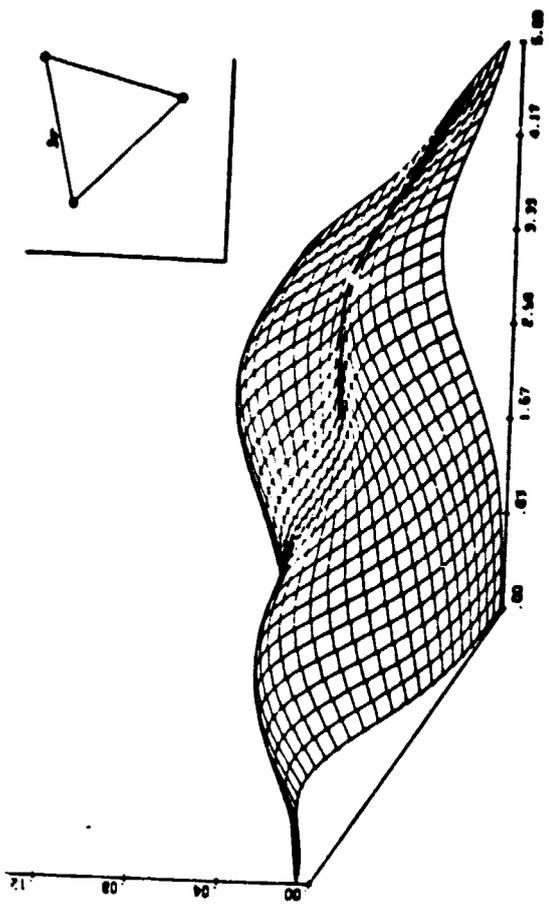
No. of CLASSES	AIC VALUES
1	941.8
2	927.5
3	857.1
4	861.2

Figure 6. Graphs and AIC values for segment  
1618/235 line 62 brightness coordinate.

~~ORIGINAL PAGE IS  
OF POOR QUALITY~~

CASE 1

TRUE MEAN VALUES	
CLASS 1	3.12, 1.00
CLASS 2	1.00, 3.12
CLASS 3	3.90, 3.90



CASE 1. MODEL MIXTURE DENSITY.

ORIGINAL PAGE IS  
OF POOR QUALITY

TABLE OF AIC VALUES (500 POINTS)

NO. OF CLASSES	AIC VALUES
1	3699.8
2	3679.8
3	3600.8
4	3600.0 *

TABLE OF AIC VALUES (500 POINTS)

NO. OF CLASSES	AIC VALUES
1	3729.1
2	3701.8
3	3642.7 *
4	3648.2

TABLE OF AIC VALUES (1000 POINTS)

NO. OF CLASSES	AIC VALUES
1	7434.3
2	7366.6
3	7219.3 *
4	7224.4

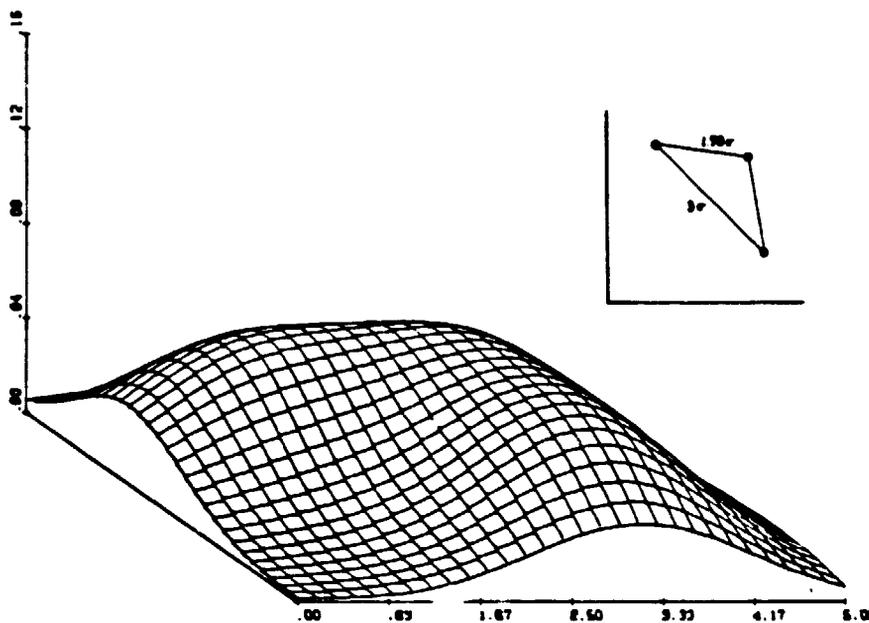
Figure 7. Mixture model density and tables of AIC values for 3 simulations ( \* denotes the minimum AIC values).

ORIGINAL PAGE IS  
OF POOR QUALITY

CASE II

TABLE OF AIC VALUES (500 POINTS)

	TRUE MEAN VALUES	No. OF CLASSES	AIC VALUES
CLASS 1	3.12, 1.00	1	3514.4
CLASS 2	1.00, 3.12	2	3471.0
CLASS 3	3.00, 3.00	3	3467.4
		4	3472.4



CASE II. MODEL MIXTURE DENSITY.

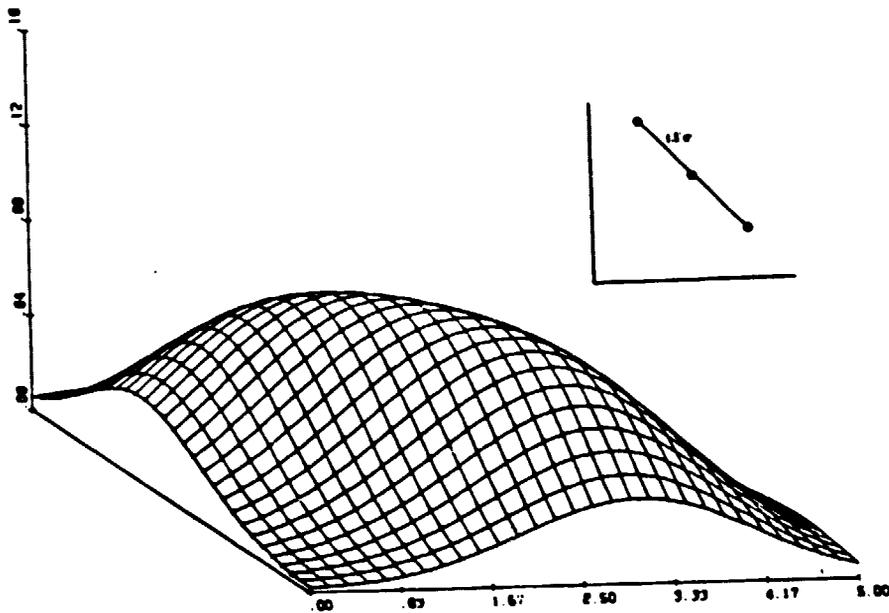
Figure 8. Mixture model density and table of AIC values.

ORIGINAL PAGE IS  
OF POOR QUALITY

CASE III

TABLE OF AIC VALUES (500 POINTS)

TRUE MEAN VALUES		NO. OF CLASSES	AIC VALUES
CLASS 1	3.12, 1.00	1	3285.8
CLASS 2	1.00, 3.12	2	3250.2 *
CLASS 3	2.06, 2.06	3	3250.8 0
		4	3256.4



CASE III. MODEL MIXTURE DENSITY.

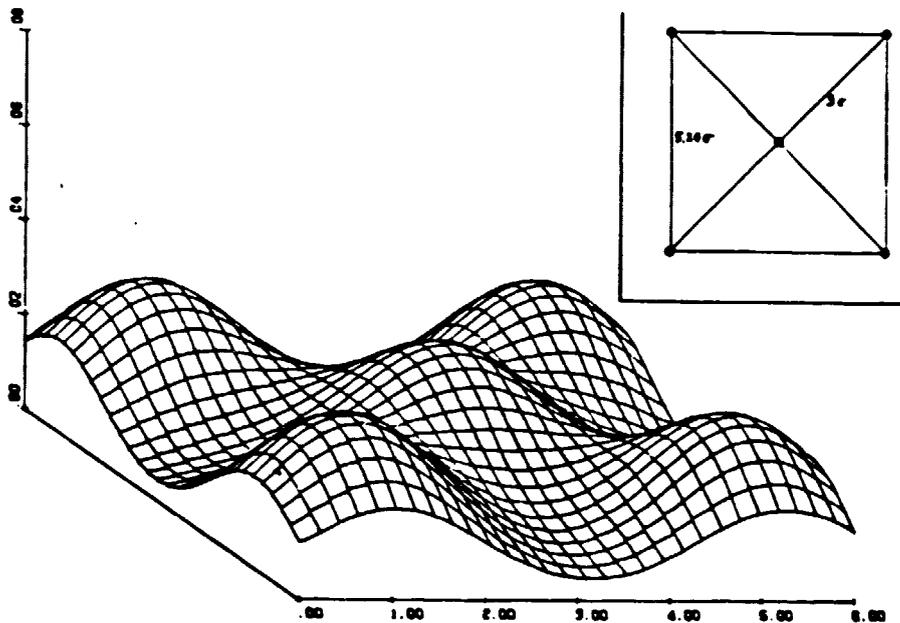
Figure 9. Mixture model density and table of AIC values.

ORIGINAL PAGE IS  
OF POOR QUALITY

CASE IV

TABLE OF AIC VALUES (1000 POINTS)

TRUE MEAN VALUES		No. OF CLASSES	AIC VALUES
CLASS 1	1.00, 1.00	1	8702.4
CLASS 2	5.24, 1.00	2	8560.6
CLASS 3	5.24, 5.24	3	8564.4
CLASS 4	1.00, 5.24	4	8411.4
CLASS 5	3.12, 3.12	5	8315.4 *
		6	8321.6



CASE IV. MODEL MIXTURE DENSITY.

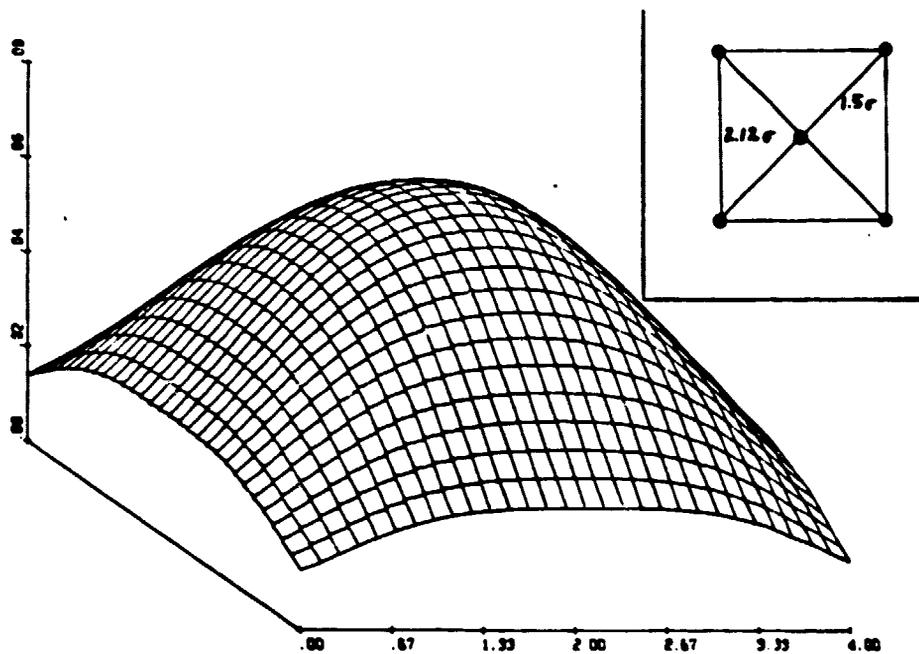
Figure 10. Mixture model density and table of AIC values.

ORIGINAL PAGE IS  
OF POOR QUALITY

CASE V

TABLE OF AIC VALUES (1000 POINTS)

	TRUE MEAN VALUES	No. OF CLASSES	AIC VALUES
CLASS 1	1.00, 1.00	1	6803.8
CLASS 2	3.12, 1.00	2	6795.6 0
CLASS 3	3.12, 3.12	3	6801.6
CLASS 4	1.00, 3.12	4	6794.2 *
CLASS 5	2.06, 2.06	5	6796.8 0
		6	6802.0



CASE V. MODEL MIXTURE DENSITY.

Figure 11. Mixture model density and table of AIC values.

ORIGINAL PAGE IS  
OF POOR QUALITY

MAXIMUM LIKELIHOOD ESTIMATES

CLASS No.	PROPORTION	MEAN VALUES
1	.34	149, 26
2	.08	88, 12
3	.16	110, 28
4	.21	102, 42
5	.04	47, 6
6	.01	104, 87
7	.07	103, 35
8	.09	99, 62

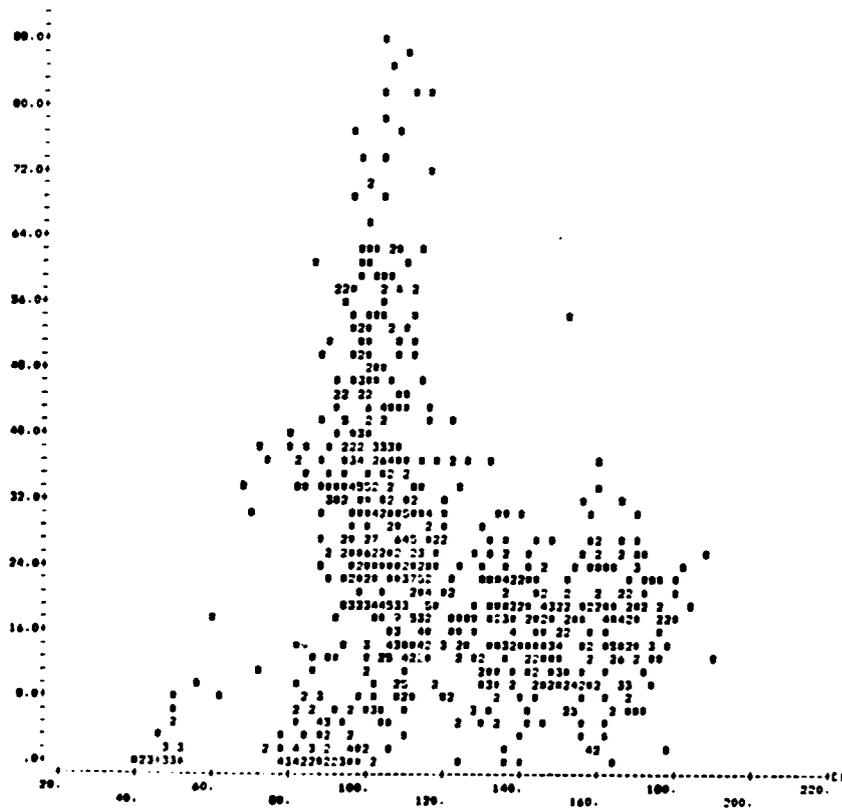


Figure 12. Maximum likelihood estimates with 8 class matter plot.