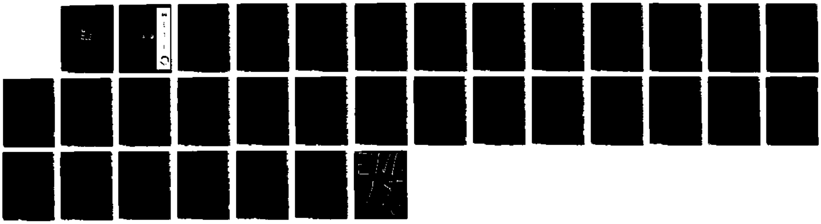
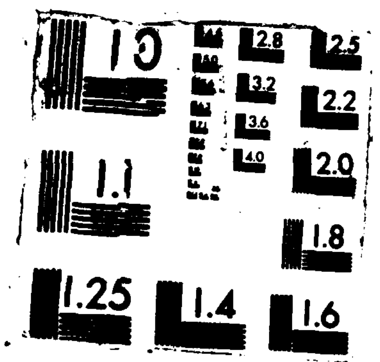
AD-A183 370 CLASSIFICATION OF SELECTED RADAR IMAGERY PATTERNS USING A BINARY TREE CLASSIFER(U) ARMY ENGINEER TOPOGRAPHIC LABS FORT BELVOIR VA N D FOX OCT 86 ETL-0442 1/1

UNCLASSIFIED F/G 17/9 NL




1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

AD-A183 370

(12)

ETL-0442

DTIC FILE CO

E T L

Classification of selected radar imagery patterns using a binary tree classifier

DTIC
ELECTE
AUG 0 6 1987
S D
D

Neil D. Fox

October 1986

Prepared for
U.S. ARMY CORP OF ENGINEERS
ENGINEER TOPOGRAPHIC LABORATORIES
FORT BELVOIR, VIRGINIA 22060-5546

US ARMY ENGINEER TOPOGRAPHIC LABORATORIES
ETL

87 8 5 011

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER ETL-0442	2. GOVT ACCESSION NO. ADA183370	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) CLASSIFICATION OF SELECTED RADAR IMAGERY PATTERNS USING A BINARY TREE CLASSIFIER		5. TYPE OF REPORT & PERIOD COVERED Research Note Dec 83 - Sep 84
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Neil D. Fox		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS U.S. Army Engineer Topographic Laboratories Fort Belvoir, Virginia 22060-5546		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Engineer Topographic Laboratories Fort Belvoir, Virginia 22060-5546		12. REPORT DATE October 1986
		13. NUMBER OF PAGES 31
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Binary Decision Tree
Bayes Classifier
Radar Imagery
Feature Selection

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report details the results of classifying radar imagery using a binary tree classifier. It was found that this classification algorithm works well with radar imagery, which would indicate a normal (Gaussian) feature vector distribution. The number of elements in each feature vector is the limiting factor, classification time is negligible once the tree structure has been created.

PREFACE

This work reported on was done under DA Project 4A161102B52C, Task B, Work Unit 0015, "Automated Radar Feature Extraction."

The work was performed during the period December 1983 to September 1984 under the supervision of Dr. Frederick W. Rohde, Team Leader, Center for Physical Sciences, and Dr. Robert D. Leighty, Director, Research Institute.

COL Alan L. Laubscher, CE, was Commander and Director, and Mr. Walter E. Boge was Technical Director of the U.S. Army Engineer Topographic Laboratories during the report preparation.

CONTENTS

ILLUSTRATIONS

CLASSIFICATION OF SELECTED RADAR IMAGERY PATTERNS USING A BINARY TREE CLASSIFIER

INTRODUCTION

By using hierarchical clustering, it is possible to improve the classification results of a single-stage classifier. The inefficiency of the single-stage classifier is in part due to the simultaneous use of all feature vector components. By using a decision tree, only the features best suited to separate the classes at a node are used.

A decision tree is a set of nodes that represent a set of feature vectors in the training set. The root node contains all the vectors in the set. This root node is then clustered into two new nodes (called "sons"), and hence, two new "subclasses" are formed. The subclasses are further clustered until only terminal nodes (those with a dominant class) remain.

An unknown sample enters the tree at the root node. A decision rule is used to classify the feature vector downward in the tree to either the left or right son. This process of sending the sample down the tree is repeated until it reaches a terminal node. It then is classified to be the same class as the dominant class of that node.

Feature selection is done at each node. Ideally, all combinations of feature vector components should be tested, but this becomes impractical owing to time constraints when the number of feature vector components become large. Fortunately, the number of feature vector components necessary to classify radar imagery can be small.

Since an assumption of hierarchical classes is made, a Bayes classifier was chosen to take advantage of the normal distribution one expects in such a system. Also, only the mean vector and covariance matrix need be computed for each node.[1]

DESIGN

The feature vector components chosen for the tree are

1. Covariance.
2. Skewness.
3. The number of lines detected by a Hough transform.
4. The average pixel value.
5. The number of pixels over a threshold value.

The clustering was done with a K-means algorithm, (k=2).

The decision rule[2] is

$$\underline{X} \in \omega_i \text{ iff } d_1(\underline{X}) > d_2(\underline{X}),\ i=1,2 \text{ and } j=2,1$$

$$d_i(\underline{X}) = \ln p(\omega_i) - \tfrac{1}{2}\ln|\Sigma_i| - \tfrac{1}{2}(\underline{X}-\mu_i)^T \Sigma_i^{-1}(\underline{X}-\mu_i)$$

Where ω_i is class i,

$\underline{X}$ is the unknown sample,

Σ_i is the covariance matrix of the i^{th} class, and

μ_i is the mean vector of the i^{th} class,

$p(\omega_i)$ is the a priori probability of $\underline{X} \in \omega_i$.

---

[1] Jack K. Mui and King-Sun Fu, "Automated Classification of Nucleated Blood Cells Using a Binary Tree Classifier," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 5, Sep. pp. 429-442, 1980.

[2] J. T. Tou and R. C. Gonzales, "Pattern Recognition Principles", Addison-Wesley Publishing Company, Reading, Massachusetts, 1974.

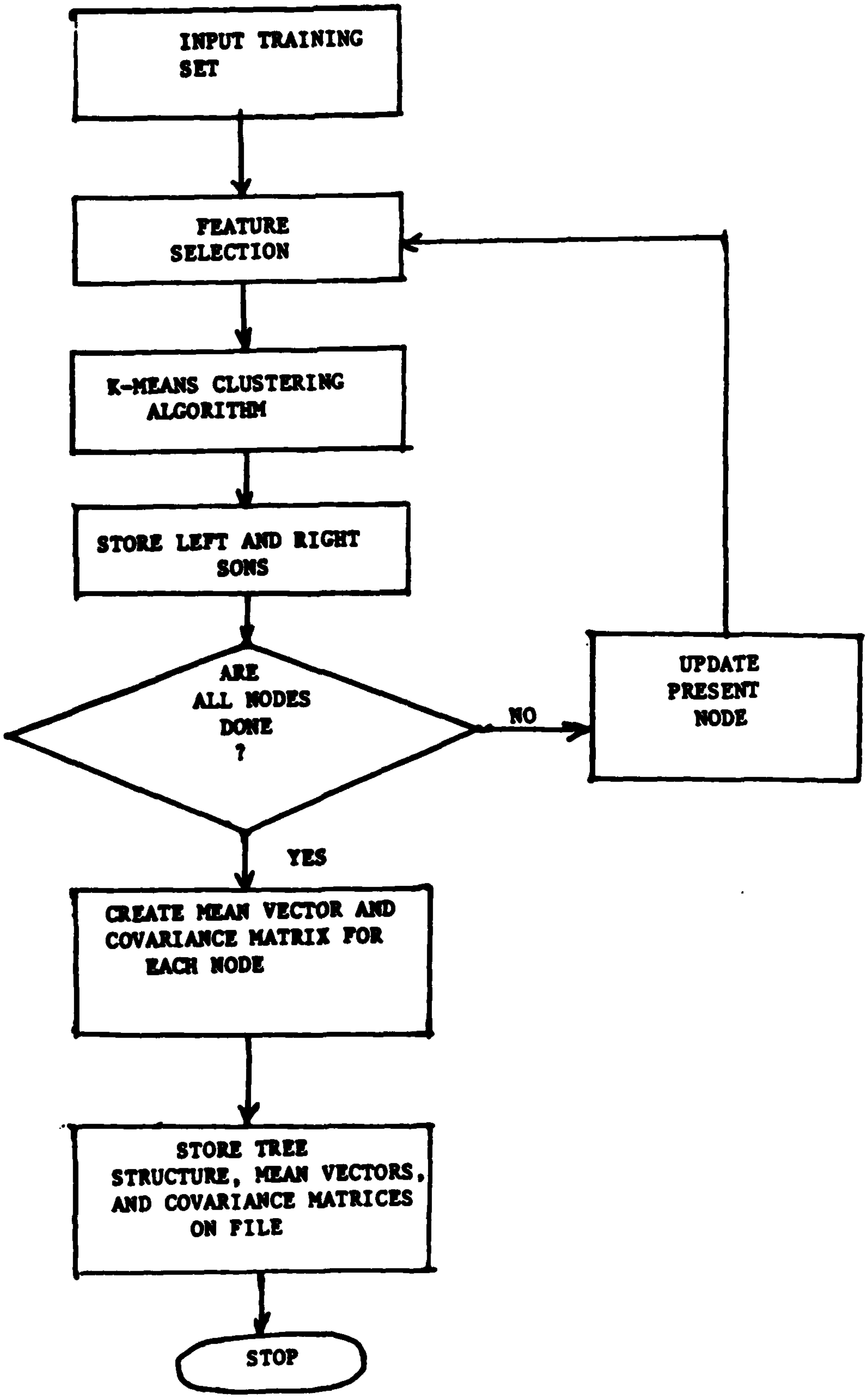


FIGURE 1. Creating the Binary Decision tree.

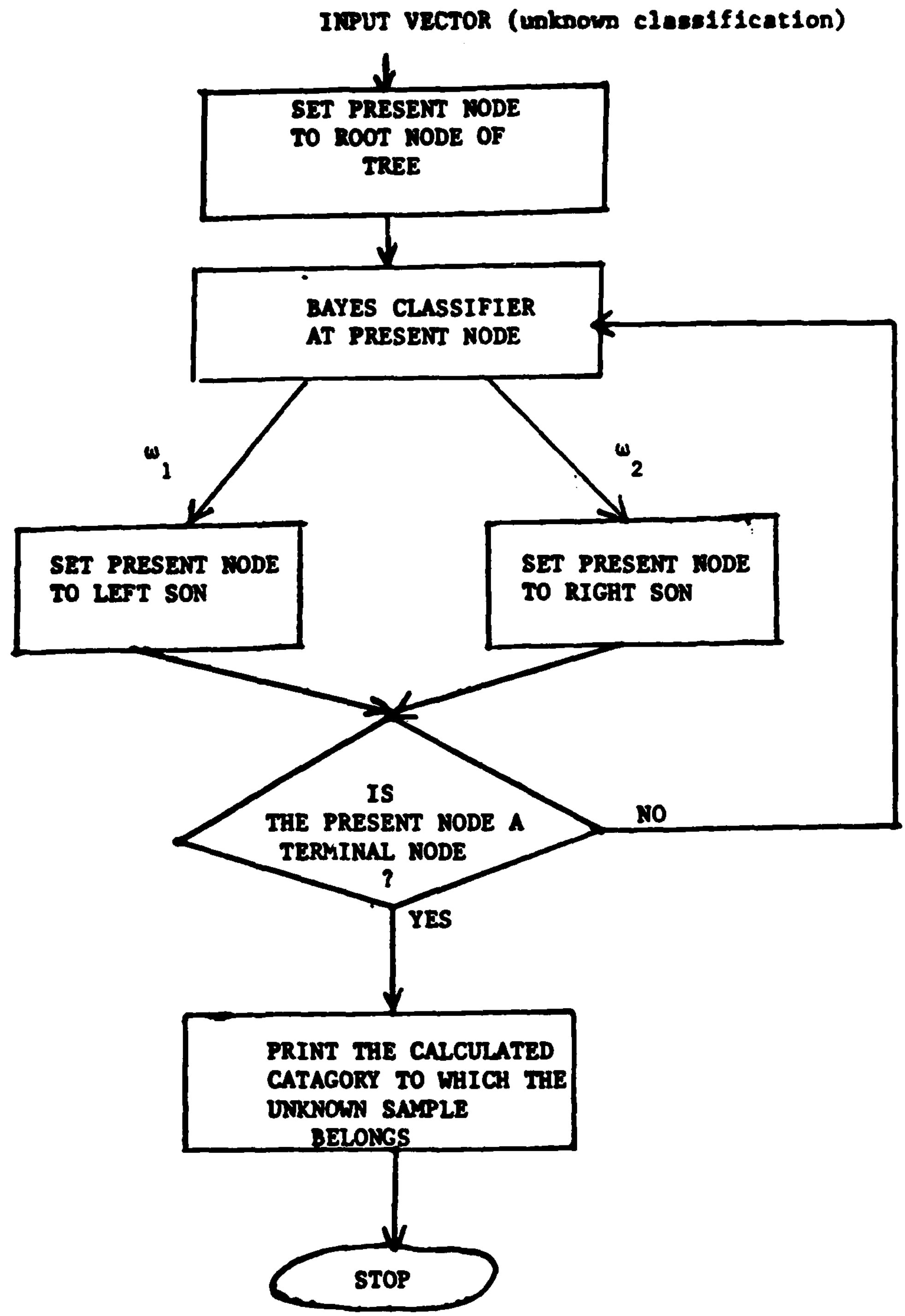


FIGURE 2. Flowchart of classifier.

The program was automated to execute all feature selection. The selection was done on the basis of how well classes were separated after clustering. The formula used is

$$\frac{I}{I+J} \sum_{n=1}^{k} I_n \log I_n + \frac{J}{I+J} \sum_{n=1}^{k} J_n \log J_n$$

where

I_n = number of vectors at left son from category n

J_n = number of vectors at right son from category n

I = number of vectors at left son

J = number of vectors at right son

k = number of categories in training set

A good separation is indicated by a large number.

The four categories of samples used for testing the tree classifier are

1. forest.
2. field.
3. water.
4. city.

The algorithms used in creating the tree and the flowchart of the tree classifier are shown in figures 1 and 2, respectively.

IMAGERY USED FOR TEST

The proposed binary tree classifier was applied to two selected sets of high resolution synthetic aperture radar imagery taken over the Huntsville, Alabama, and the Elizabeth City, North Carolina, areas with the APD-10 and the UPD-4 radar systems, respectively. Sections of the radar imagery were digitized and stored on a disk unit. A Lexidata System 3400 display processor was used to display the images on a cathode ray tube. Software was written to take 100 samples for each of four terrain classes from imagery displayed on the screen. Each image sample consisted of 32 by 32 pixels that were located within a section of one particular terrain class. The four classes considered were (1) cities (combination of commercial and residential structures, DLMS category #504 FIC 301 and #505 FIC 401), (2) fields (agriculture used primarily for crop and pasture land, DLMS category #510 FIC 950), (3) water (rivers with smooth fresh water, DLMS category #510 FIC 940 and fresh water subject to ice, lakes and reservoirs, DLMS category #510 FIC 943), and (4) forests (mixed trees, deciduous and evergreens, DLMS category #510 FIC 954). A feature vector consisting of 15 components was computed for each sample. These 15 components were made up of the first- and second-order gray level histogram statistics computed from each sample. These components were then used as the input for the binary tree classifier.

RESULTS

Two sets of radar imagery (Huntsville, Alabama and Elizabeth City, North Carolina) were used to create two binary decision trees. The resulting trees are shown in figures 3 and 4, and the results of classifying the training sets are shown in figures 5 and 6.

Often, only one or two feature vector components were used at a node, which illustrates the advantage of using a hierarchical approach rather than the traditional single-stage classifier. For example, although thresholding was an excellent feature vector component when separating water from field, forest, and city, it was a poor feature with respect to the separation of city and forest.

The overall classification accuracy for the Huntsville image samples was 99.25 percent and for the Elizabeth City samples, 98.50 percent.

CONCLUSIONS

1. A binary tree classifier can be used to classify a selected set of radar images with high accuracy as illustrated. However, a new tree hierarchy will be needed if a new set of images from a different geographical area is to be classified optimally with a reasonably high accuracy.

2. Only the most pertinent feature vector components should be used at each node. This reduces processing time and improves classification accuracy.

3. An automated classifier is practical if a small set of feature vector components is used. For vectors with a large number of features, some human interaction in feature selection may be necessary to avoid lengthy processing time. The create-tree program took approximately 11 minutes to finish a tree using five features.

4. This work represents an application of a recently developed statistical classification method. A possible limitation is the requirement to create a new tree hierarchy for training, whenever a new set of images is to be classified with a reasonably high accuracy.

REFERENCES

1. Jack K. Mui and King-Sun Fu, "Automated Classification of Nucleated Blood Cells Using a Binary Tree Classifier," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, No. 5, Sep, pp. 429-442, 1980.

2. J. T. Tou and R. C. Gonzales, *Pattern Recognition Principles*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1974.

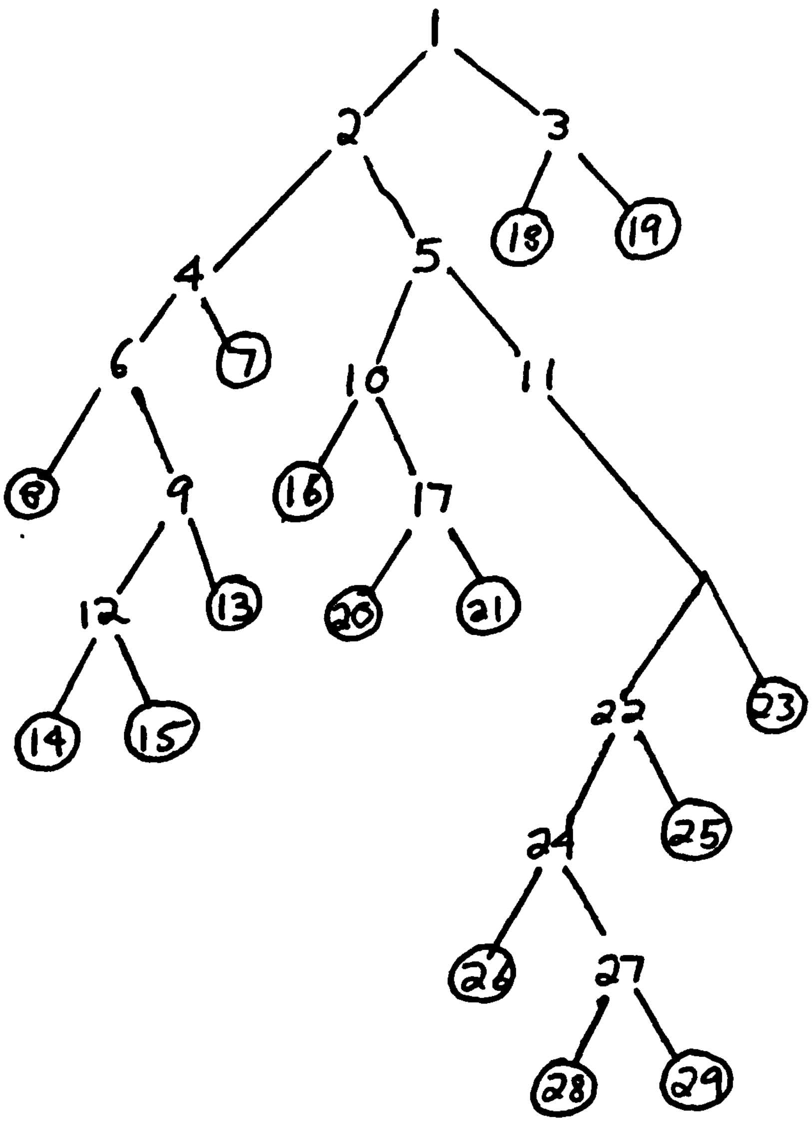


FIGURE 3. Decision tree for image samples from Huntsville, AL.

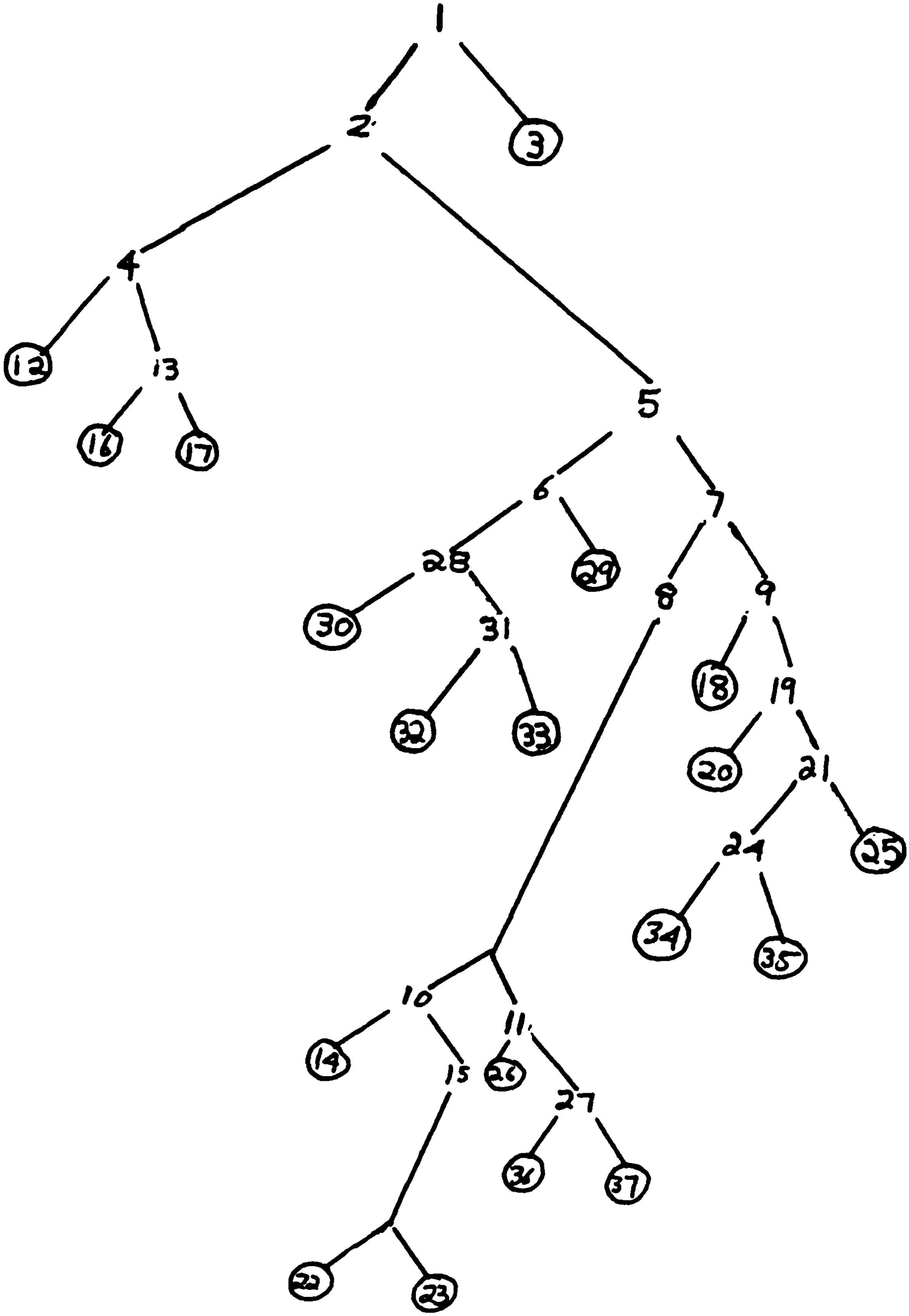


FIGURE 4. Decision tree for image samples from Elizabeth City, NC.

TRUE CAT / REC CAT	FIE	WAT	FOR	CIT
FIE	98	0	1	0
WAT	2	100	0	0
FOR	0	0	99	0
CIT	0	0	0	100

FIGURE 5. Classified results for image samples from Huntsville, AL.

TRUE CAT / REC CAT	FIE	WAT	FOR	CIT
FIE	100	0	1	0
WAT	0	100	0	0
FOR	0	0	96	2
CIT	0	0	3	98

FIGURE 6. Classified results for image samples from Elizabeth City, NC.

Appendix A. Data for Creation of Decision Tree

Huntsville, Alabama

Node #	Features Used in Separation of Node
1	THRSH, HOU, COV
2	THRSH, HOU
3	HOU
4	AVV, THRSH, HOU
5	AVV, SKW, THRSH
6	SKW
9	SKW
10	SKW
11	HOU
12	THRSH
17	HOU
22	THRSH, HOU
24	AVV
27	AVV

Key:
- HOU — Hough Transform
- SKW — Skewness
- COV — Covariance
- THRSH — Threshold
- AVV — Average Pixel Value.

Appendix A. (Cont'd)

Elizabeth City, North Carolina

Node #	Features Used in Separation of Node
1	SKW
2	THRSH
4	HOU
5	AVV, THRSH
6	AVV, COV
7	AVV, HOU
8	AVV
9	SKW, HOU
10	AVV, SKW
11	SKW, HOU, COV
13	THRSH
15	AVV
19	AVV
20	COV
21	SKW
24	SKW
27	SKW
28	AVV, THRSH
31	AVV, SKW

Appendix A. (Con't)

Elizabeth City, North Carolina

NODE #	WATER	FOREST	CITY	FIELD
1	100	100	100	100
2	0	100	100	100
3	100	0	0	0
4	0	12	0	100
5	0	88	100	0
6	0	63	2	0
7	0	25	98	0
8	0	19	24	0
9	0	6	74	0
10	0	9	21	0
11	0	10	3	0
12	0	5	0	0
13	0	7	0	100
14	0	1	17	0
15	0	8	4	0
16	0	1	0	100
17	0	6	0	0
18	0	0	67	0
19	0	6	7	0
20	0	2	2	0
21	0	4	5	0
22	0	8	1	0
23	0	0	3	0
24	0	2	5	0
25	0	2	0	0
26	0	8	0	0
27	0	2	3	0
28	0	45	2	0
29	0	18	0	0
30	0	17	0	0
31	0	28	2	0
32	0	27	0	0
33	0	1	2	0
34	0	0	5	0
35	0	2	0	0
36	0	2	0	0
37	0	0	3	0
38	0	0	2	0
39	0	2	0	0

Vector Distribution at Each Node of Decision Tree

Appendix A. (Con't)

Huntsville, Alabama

NODE #	WATER	FOREST	CITY	FIELD
1	100	100	100	100
2	5	100	100	97
3	95	0	0	3
4	5	10	88	95
5	0	90	12	2
6	0	10	88	95
7	5	0	0	0
8	0	0	0	95
9	0	10	88	0
10	0	7	12	0
11	0	83	0	2
12	0	10	25	0
13	0	0	63	0
14	0	10	0	0
15	0	0	25	0
16	0	4	0	0
17	0	3	12	0
18	0	0	0	2
19	95	0	0	1
20	0	3	0	0
21	0	0	12	0
22	0	62	0	2
23	0	21	0	0
24	0	13	0	2
25	0	49	0	0
26	0	2	0	0
27	0	11	0	2
28	0	10	0	0
29	0	1	0	2

Vector Distribution at Each Node of Decision Tree

```
&TREC3 T=00004 IS ON CR00024 USING 00030 BLKS R=0000

FTN4X
      PROGRAM TREC3
C
C     THIS PROGRAM READS DATA OFF THE DISK WHICH WAS CREATED BY
C     PROGRAM "TREEF". THIS DATA IS USED TO IMPLEMENT A DECISION
C     TREE.
C
      DIMENSION COV(5,5,61),MEAN(5,61),IDCB(144),ITREE(61,2)
      DIMENSION XIN(400,5),IDCB4(144),LABS(4),INP2(400)
      DIMENSION LUOT(5),LL(4,61),LABL(4),WV(5,61),XS(5)
      DIMENSION INAME(3),X(5),INDEX(61)
      DIMENSION NODE(61),MEAN1(5),COV1(5,5),COV2(25)
      DIMENSION MEANS(5),NAME(3)
      REAL MEAN,MEAN1,MEANS
C
C                    VARIABLE ARRAYS
C                    ---------------
C
C       COV - COVARIANCE MATRICES OF ALL NODES
C       COV1 - COVARIANCE MATRIX FOR A NODE BEFORE MASK
C       COV2 - COVARIANCE MATRIX FOR A NODE AFTER MASK
C       INDEX - NUMBER OF VECTORS AT EACH NODE
C       INP2 - LABELS OF VECTORS AT A NODE
C       ITREE - TREE STRUCTURE READ FROM TREE FILE
C       LABL - THE CLASS LABELS READ FROM TREE FILE
C       LABS - THE CLASS LABELS READ FROM PRP FILE (NOT USED)
C       LL -  NUMBER OF VECTORS IN A CLASS AT A NODE
C       MEAN - MEANS OF ALL NODES (REAL)
C       MEAN1 - MEAN OF A NODE BEFORE MASK (REAL)
C       MEANS - MEAN OF A NODE AFTER MASK (REAL)
C       NODE - THE NODE NUMBERS OF THE TREE (NOT USED)
C       WV - THE MASKS FOR ALL NODES READ FROM TREE FILE
C       X - THE VECTOR BEING CLASSIFIED BEFORE MASK
C       XIN - THE VECTORS TO BE CLASSIFIED READ FROM PRP FILE
C       XS - THE VECTOR BEING CLASSIFIED AFTER MASK
C
C                    VARIABLES
C                    ---------
C
C        IANS - USER INPUT: ANSWER
C        INDX1 - INDEX USED IN PACKING VECTOR VIA THE MASK
C        INDX2 - INDEX USED IN PACKING VECTOR VIA THE MASK
C        IRGT - THE RIGHT NODE OF NODEP
C        KLAC - THE NUMBER OF CLASSES (READ FROM PROPERTY FILE)
C        KLAS - THE NUMBER OF CLASSES (READ FROM TREE FILE)
C        LEFT - THE LEFT NODE OF NODEP
C        LVEC - THE NUMBER OF VECTORS IN THE PROPERTY FILE
C        MAX - THE INDEX NUMBER OF THE DOMINANT CLASS OF A NODE
C        MAXN - THE NUMBER OF VECTORS IN THE DOMINANT CLASS OF A NODE
C        NELE - NUMBER OF ELEMENTS BEFORE PACKING (USING MASK)
C        NODEP - THE NODE AT WHICH THE VECTOR IS BEING TESTED
C        NUMND - THE NUMBER OF NODES IN THE TREE STRUCTURE
C
C
      CALL RMPAR(LUOT)
      CALL ERLU(LUOT)
C
C     OPEN FILES AND READ TREE STRUCTURE
```

```
C
      WRITE(LUOT,774)
774   FORMAT(" ENTER NAME OF TREE-FILE")
      READ(LUOT,776)INAME
776   FORMAT(3A2)
      WRITE(LUOT,778)
778   FORMAT(" ENTER PROPERTY FILE TO BE CLASSIFIED")
      READ(LUOT,776)NAME
      WRITE(LUOT,150)
150   FORMAT("        DO YOU WANT A HARD COPY ?")
      READ(LUOT,250)IANS
250   FORMAT(1A1)
      IF(IANS.EQ.1HY) LUOT=38
      CALL OPEN (IDCB,IERR,INAME)
      CALL READF(IDCB,IERR,KLAS)
      CALL READF(IDCB,IERR,NUMND)
      CALL READF(IDCB,IERR,LABL)
      CALL READF(IDCB,IERR,LL)
      CALL READF(IDCB,IERR,ITREE)
      CALL READF(IDCB,IERR,WV)
      CALL READF(IDCB,IERR,INDEX)
      DO 1 N=1,NUMND
          CALL READF(IDCB,IERR,NODE(N))
          CALL READF(IDCB,IERR,NELE)
          CALL READF(IDCB,IERR,MEAN1)
          CALL READF(IDCB,IERR,COV1)
            DO 2 I=1,NELE
            DO 2 J=1,NELE
2             COV(I,J,N)=COV1(I,J)
             DO 3 I=1,NELE
3             MEAN(I,N)=MEAN1(I)
1     CONTINUE
C
      CALL CLOSE(IDCB)
C
C
C     INPUT THE VECTORS TO BE CLASSIFIED
C
      CALL OPEN(IDCB4,IERR,NAME)
      CALL READF(IDCB4,IERR,LVEC)
      CALL READF(IDCB4,IERR,LELE)
      CALL READF(IDCB4,IERR,KLAC)
      CALL READF(IDCB4,IERR,LABS)
      CALL READF(IDCB4,IERR,XIN)
      CALL READF(IDCB4,IERR,INP2)
      CALL CLOSE(IDCB4)
5     DO 1223 IVEC=1,LVEC
      DO 99 IP=1,NELE
99    X(IP)=XIN(IVEC,IP)
C
C     PRESENT NODE OPERATED ON IS SET TO "1"
C
      NODEP=1
C
C     THE LEFT AND RIGHT CHILD OF EACH NUDE IS FOUND
C
10    LEFT=ITREE(NODEP,1)
      IRGT=ITREE(NODEP,2)
C
C     SEE IF NUDE IS TERMINAL
```

```
C
        IF( (LEFT.EQ.0).OR.(IRGT.EQ.0) ) GOTO 200
C
C       FIND DISTANCE FOR LEFT AND RIGHT NODES
C
        DO 50 I=1,NELE
        DO 50 J=1,NELE
50      COV1(I,J)=COV(I,J,LEFT)
        DO 55 I=1,NELE
55      MEAN1(I)=MEAN(I,LEFT)
        INDX1=0
        INDX2=0
        DO 20 ITEL=1,NELE
        IF(WV(ITEL,NODEP).NE.0.0) THEN
          INDX1=INDX1+1
          MEANS(INDX1)=MEAN1(ITEL)
          XS(INDX1)=X(ITEL)
        ENDIF
        DO 20 JTEL=1,NELE
        IF(WV(ITEL,NODEP).NE.0.AND.WV(JTEL,NODEP).NE.0) THEN
         INDX2=INDX2+1
         COV2(INDX2)=COV1(ITEL,JTEL)
        ENDIF
20      CONTINUE
        RLFT=INDEX(ITREE(NODEP,1))
        ALLV=INDEX(NODEP)
        CALL MDST(COV2,NELE,MEANS,XS,W1,INDX1,NELE**2,ALLV,RLFT)
        DO 60 I=1,NELE
        DO 60 J=1,NELE
60      COV1(I,J)=COV(I,J,IRGT)
        DO 65 I=1,NELE
65      MEAN1(I)=MEAN(I,IRGT)
        INDX1=0
        INDX2=0
        DO 2000 ITEL=1,NELE
        IF(WV(ITEL,NODEP).NE.0) THEN
          INDX1=INDX1+1
          MEANS(INDX1)=MEAN1(ITEL)
          XS(INDX1)=X(ITEL)
        ENDIF
        DO 2000 JTEL=1,NELE
        IF(WV(ITEL,NODEP).NE.0.AND.WV(JTEL,NODEP).NE.0) THEN
           INDX2=INDX2+1
          COV2(INDX2)=COV1(ITEL,JTEL)
        ENDIF
2000    CONTINUE
C
        RRGT=INDEX(ITREE(NODEP,2))
        CALL MDST(COV2,NELE,MEANS,XS,W2,INDX1,NELE**2,ALLV,RRGT)
C
C       MAKE THE DECISION OF WHETHER TO MAKE TO LEFT OR RIGHT NODE THE
C       PRESENT NODE
C
        IF(W1.LE.W2) THEN
            NODEP=LEFT
               ELSE
           NODEP=IRGT
        ENDIF
C
C       CLEAR COV2 AND MEANS
```

```
C
      DO 9000 I=1,NELE**2
9000  COV2(I)=0.0
      DO 9001 I=1,NELE
9001  MEANS(I)=0.0
C
C     GO TO NEXT NODE
C
      GOTO 10
C
C     WRITE THE FINAL DESTINATION OF THE VECTOR
C
200   MAX=1
      MAXN=LL(1,NODEP)
      DO 220 I=1,KLAS
       IF (MAXN.LT.LL(I,NODEP)) THEN
         MAX=I
         MAXN=LL(I,NODEP)
       ENDIF
220   CONTINUE
      WRITE(LUOT,101) LABL(MAX),NODEP
101   FORMAT(" CLASS: ",1A2,3X,I3)
C
C     REPEAT BY ASKING FOR ANOTHER VECTOR
C
1223  CONTINUE
C
      STOP
      END
C
      SUBROUTINE MDST(COV,NELE,MEAN,X,W,INDX1,NELEQ,ALLV,RORL)
C
C     THIS IMPLEMENTS THE MINIMUM DISTANCE CLASSIFIER
C
      DIMENSION COV(NELEQ),MEAN(NELE),X(NELE)
      DIMENSION R(5),TR(5),R1(5)
      DIMENSION L(5),M(5)
      REAL MEAN
C
C
      CALL INV(COV,INDX1,DET,L,M)
      CALL SUMAT(X,MEAN,R,INDX1,1)
      CALL TRMAT(R,TR,INDX1,1,0)
      CALL PRMAT(TR,COV,R1,1,INDX1,INDX1)
      CALL PRMAT(R1,R,W,1,INDX1,1)
C
C     CALCULATE PROBABLITY
C
      PROB=RORL/ALLV
      IF(DET.NE.0.0) THEN
        W=PROB-(ALOG(ABS(DET)))/2.0-W/2.0
        W=-W
       ELSE
        W=PROB+20E20/2.0-W/2.0
        W=-W
      ENDIF
      RETURN
      END
      END$
```

```
TREEF T=00004 IS ON CR00024 USING 00072 BLKS R=0000

      FTN4X
                                 PROGRAM TREEF
C
C           THIS PROGRAM CREATES A BINARY TREE STRUCTURE AND
C           STORES IT ON DISK. AT EACH NODE A K-MEANS ALGORITHM
C           IS CALLED TO SEPARATE THE VECTORS INTO TWO NEW NODES.
C
C           THIS PROGRAM READS IN A PROPERTY FILE CREATED BY PROGRAM
C           'MKPRP'. THE FORMAT OF THIS FILE IS:
C
C     VARIABLE                SIZE                    DESCRIPTION
C     --------                ----                    -----------
C
C     NVEC                     1                      # OF VECTORS
C
C     NELE                     1                      # OF ELEMENTS
C
C     KLAS                     1                      # OF CLASSES
C
C     LABL                    KLAS                    NAMES OF CLASSES
C
C     XX                NVEC X NELE X 2               VECTORS
C
C     IN1                     NVEC                    VECTOR LABELS / THEY
C                                                     IN THE SAME ORDER AS
C                                                     THE VECTORS IN XX.
C
C
C
C
C                             VARIABLE ARRAYS
C                             ---------------
C
C      C1 -- WORK ARRAY FOR MVCOV
C      C2 -- WORK ARRAY FOR MVCOV
C      COV - COVARIANCE MATRIX IN MVCOV
C      IN1 - LABEL BUFFER FOR LEFT NODE
C      IN2 - LABEL BUFFER FOR RIGHT NODE
C      INB - LABEL BUFFER FOR INPUT TO CLUS
C      INDEX - THE NUMBER OF VECTORS AT EACH NODE
C      IREC - STORAGE FOR RECORD THAT CORRESPONDS TO A NODE NUMBER
C      ITREE - THE TREE STRUCTURE
C      LABL - THE NAMES (LABELS) OF THE CLASSES
C      LL -- THE NUMBER OF VECTORS IN EACH CLASS AT EACH NODE
C      MEAN - THE MEAN CALCULATED IN MVCOV. THIS IS A REAL ARRAY.
C      MEAN1 - THE TWO MEANS CALCULATED FROM CLUS. THIS IS A REAL ARRAY.
C      OLDM -- THE OLD MEAN VALUES IN CLUS
C      W -- THE MASKS FOR ALL NODES
C      WOLD -- THE PREVIOUS MASK VALUE WHEN IN "SQ" ROUTINE
C      WORTH - STORES HOW WELL SEPARATED EACH NODE IS
C      WPRIM - THE BEST MASK IS STORED HERE AFTER "SQ" ROUTINE
C      X1 -- THE VECTOR BUFFER FOR THE LEFT NODE
C      X2 -- THE VECTOR BUFFER FOR THE RIGHT NODE
C      XX -- THE VECTOR BUFFER FOR THE NODE BEING CLUSTERED
C
C                          VARIABLES
C                          ---------
C
```

```
C      ICMND - INPUT FROM USER IN COMMAND LOOP
C      IND1 - NUMBER OF VECTORS IN LEFT NODE
C      IND2 - NUMBER OF VECTORS IN RIGHT NODE
C      KLAS - TOTAL NUMBER OF CLASSES
C      KNODE - MAXIMUM NUMBER OF NODES ALLOWED
C      KREC -- PRESENT NUMBER OF RECORDS
C      N1 -- LEFT NODE NUMBER
C      N2 -- RIGHT NODE NUMBER
C      NELE - MAXIMUM NUMBER OF ELEMENTS
C      NODE - NUMBER OF NODE TO BE CLUSTERED OR OPERATED ON
C      NVEC - NUMBER OF VECTORS IN PROPERTY FILE
C      OLDVL - THE OLD VALUE OF THE VARIABLE "VALUE"
C      PCLAS - NUMBER OF VECTORS IN DOMINANT CLASS AT A NODE
C      SMSH1 - INTERMEDIATE VARIABLE FOR CALCULATING "VALUE"
C      SMSH2 - INTERMEDIATE VARIABLE FOR CALCULATING "VALUE"
C      VALUE - MEASURE OF THE CLUSTER. LARGE VALUES ARE GOOD.
C      VIN1 - INTERMEDIATE VARIABLE FOR CALCULATING "VALUE"
C      VIN2 - INTERMEDIATE VARIABLE FOR CALCULATING "VALUE"
C      WMIN - THE MINIMUM VALUE OF WORTH(NODE). THIS IS THE BEST VALUE.
C
       DIMENSION LUOT(5),XX(400, 5),OLDM( 2, 5),C1( 5, 5)
       DIMENSION IN1(400),C2(5,5),COV(5,5),MEAN1(2,5),IN2(400)
       DIMENSION LABL(4),INAME(3),MEAN(5),LL(4,61),INB(400),W(5,61)
       DIMENSION X1(400,5),X2(400,5),ITREE(61,2),INDEX(61),WPRIM(5)
       DIMENSION INAM1(3),WOLD(5),WORTH(61)
       REAL MEAN,MEAN1
       DIMENSION IDCB(144),IDCB1(144),IDCB2(144),IREC(100)
C
C      SET OLD VALUE TO 0
C
       OLDVL=0.0
C
C      SET LEFT, RIGHT, AND STARTING NODE TO INITIAL VALUES
C
       NODE=1
       N1=0
       N2=1
C
C      SET MAX. NODES
C
       KNODE=61
C
C
C      INIT. LU
C
       CALL RMPAR(LUOT)
       LU=LUOT(1)
       CALL ERLU(LU)
C
C
3      FORMAT(1A2)
C
C      READ FILES
C
       WRITE(LUOT,333)
333    FORMAT(" ENTER NAME OF PROPERTY FILE")
       READ(LUOT,334)INAME
       WRITE(LUOT,335)
335    FORMAT(" ENTER THE FILE-NAME FOR THE TREE STRUCTURE")
       READ(LUOT,334)INAM1
```

```
334    FORMAT(3A2)
       CALL OPEN(IDCB,IERR,INAME)
       CALL READF(IDCB,IERR,NVEC)
       CALL READF(IDCB,IERR,NELE)
       CALL READF(IDCB,IERR,KLAS)
       CALL READF(IDCB,IERR,LABL)
       CALL READF(IDCB,IERR,XX)
       CALL READF(IDCB,IERR,IN1)
       CALL CLOSE(IDCB)
C
C      PURGE OLD AND CREATE NEW SCRATCH FILE
C
       CALL PURGE(IDCB1,IERR,5HXBUFR)
       CALL PURGE(IDCB2,IERR,5HIBUFC)
       CALL CREAT(IDCB1,IERR,5HXBUFR,500,3,0,-24)
       CALL CREAT(IDCB2,IERR,5HIBUFC,500,3,0,-24)
       IF(IERR.LT.0) WRITE(LUOT,50)IERR
C
C      SET FIRST INDEX
C
       INDEX(NODE)=NVEC
C
C      FILL LABEL ARRAY FOR NODE 1
C
       DO 1192 I=1,KLAS
1192   LL(I,1)=100
C
C      STORE VECTOR NAMES ON SCRATCH FILE
C
       CALL WRITF(IDCB2,IERR,IN1,NVEC)
C
C      INITIALIZE RECORD NUMBER
C
       KREC=1
C
C      STORE DATA ON SCRATCH FILE
C
       CALL WRITF(IDCB1,IERR,XX,2*NVEC*NELE)
       IF(IERR.LT.0) WRITE(LUOT,50)IERR
       IREC(1)=KREC
C
C      ENTER COMMAND LOOP. THE POSSIBLE COMMANDS ARE :
C
C      LU -- CHANGE OUTPUT DEVICE
C      TR -- TERMINATE PROGRAM
C      SQ -- SEQUENCE W-MASK THROUGH ALL COMBINATIONS AND CREATE TREE
C      CT -- CREATE TREE USING A W-MASK OF ALL 1'S (ALL FEATURES USED)
C
96     WRITE(LUOT,250)
250    FORMAT(" ENTER COMMAND",/,"??")
       READ(LUOT,3)ICMND
       IF(ICMND.EQ.2HSQ) GOTO 252
       IF(ICMND.EQ.2HCT) THEN
         DO 3000 I=1,NELE
         DO 3000 J=1,KNODE
3000     W(I,J)=1.0
         GOTO 252
       ENDIF
       IF(ICMND.EQ.2HTR) STOP
C
```

```
      IF(ICMND.EQ.2HLU) THEN
256     READ(LUOT,*)LU
          IF(LU.LE.0.OR.LU.GE.39) GOTO 256
      ENDIF
C
C     IF NO RECOGNIZABLE COMMAND IS ENTERED, THEN ASK FOR ANOTHER
C
      GOTO 96
C
C     SPLIT THE NODE SELECTED INTO TWO NEW NODES
C
252   NODE=1
4444  IF(ICMND.EQ.2HSQ) THEN
        DO 7721 I=1,NELE
7721    W(I,NODE)=0.0
        CALL SEQUN(WOLD,W,NODE,KNODE,NELE)
      ENDIF
5994  FORMAT(/,10X,"W MASK ==> ",5F2.0)
       N1=N1+2
       N2=N2+2
C
C     STORE LEFT AND RIGHT CHILD IN ARRAY, "ITREE"
C
       ITREE(NODE,1)=N1
       ITREE(NODE,2)=N2
C
C     FIND CORRECT RECORDS
C
      CALL POSNT(IDCB1,IERR,IREC(NODE),1)
      CALL POSNT(IDCB2,IERR,IREC(NODE),1)
C
      IF(IERR.LT.0) WRITE(LUOT,50)IERR
50    FORMAT(" FILE-ERROR #",I4)
C
      CALL READF(IDCB1,IERR,XX)
      IF(IERR.LT.0) WRITE(LUOT,50)IERR
C
C     READ LABELS FOR VECTORS AT THIS NODE
C
      CALL READF(IDCB2,IERR,INB)
C
C     CLUSTER THE VECTORS,
C
770   CALL CLUS(XX,NELE,INDEX(NODE),X1,IND1,X2,IND2,NODE,N1,N2,MEAN1
     $,OLDM,KNODE,NVEC,INB,IN1,IN2,W)
C
      IF(ICMND.EQ.2HNX) WRITE(LU,5994)(W(K,NODE),K=1,NELE)
      IF(ICMND.EQ.2HCT) WRITE(LU,5994)(W(K,NODE),K=1,NELE)
C
C     STORE LEFT AND RIGHT NODE NAMES IN LIST
C
      INDEX(N1)=IND1
      INDEX(N2)=IND2
C
C     PRINT THE RESULTS OF THE CLUSTERING AND ASK IF IT IS TO BE STORED
C
      DO 27 I=N1,N2
      IF(ICMND.EQ.2HNX) WRITE(LU,29)I
      IF(ICMND.EQ.2HCT) WRITE(LU,29)I
29    FORMAT(16X,"NODE #",I3)
```

```
      IF(ICMND.EQ.2HNX) WRITE(LU,26) (LABL(J),J=1,KLAS)
      IF(ICMND.EQ.2HCT) WRITE(LU,26) (LABL(J),J=1,KLAS)
      DO 30 NUMK=1,KLAS
      LL(NUMK,I)=0
      DO 30 J=1,INDEX(I)
      IF(I.EQ.N1) THEN
        IF(IN1(J).EQ.LABL(NUMK))LL(NUMK,I)=LL(NUMK,I)+1
      ELSE
        IF(IN2(J).EQ.LABL(NUMK))LL(NUMK,I)=LL(NUMK,I)+1
      ENDIF
30    CONTINUE
C
      IF(ICMND.EQ.2HNX) WRITE(LU,32)(LL(INUMB,I),INUMB=1,KLAS)
27    IF(ICMND.EQ.2HCT) WRITE(LU,32)(LL(INUMB,I),INUMB=1,KLAS)
32    FORMAT(12X,4(1X,I3),/)
C
      IF(ICMND.EQ.2HSQ) THEN
C
C     HERE IS THE IMPLEMENTATION OF A FORMULA THAT WILL
C     TELL HOW GOOD A SEPARATION IS
C
      SMSH1=0.0
      SMSH2=0.0
      DO 7225 IJ=1,KLAS
      IF(LL(IJ,N1).NE.0) THEN
        SMSH1=FLOAT(LL(IJ,N1))*ALOGT(FLOAT(LL(IJ,N1)))+SMSH1
      ENDIF
      IF(LL(IJ,N2).NE.0) THEN
        SMSH2=FLOAT(LL(IJ,N2))*ALOGT(FLOAT(LL(IJ,N2)))+SMSH2
      ENDIF
7225  CONTINUE
C
      VIN1=FLOAT(IND1)/FLOAT(IND1+IND2)
      VIN2=FLOAT(IND2)/FLOAT(IND1+IND2)
      VALUE=VIN1*SMSH1+VIN2*SMSH2
C
C     SET VALUE TO 0 IF IND1 OR IND2 IS 0 OR 1
C
      IF(IND1.EQ.0.OR.IND2.EQ.0.OR.IND1.EQ.1.OR.IND2.EQ.1) VALUE=0.0
C
C     IF THIS IS THE LARGEST VALUE YET, THEN SAVE THE CORRESPONDING
C     W MASK
C
      IF(VALUE.GT.OLDVL) THEN
        DO 6461 ISW=1,NELE
6461     WPRIM(ISW)=W(ISW,NODE)
        OLDVL=VALUE
      ENDIF
C
        CALL SEQUN(WOLD,W,NODE,KNODE,NELE)
        IQUIT=1
        DO 1055 I=1,NELE
1055    IF(W(I,NODE).EQ.1) IQUIT=0
C
            IF(IQUIT.EQ.1) THEN
                ICMND=2HNX
                VALUE=0.0
                OLDVL=0.0
                DO 9911 I=1,NELE
9911            W(I,NODE)=WPRIM(I)
```

```
              ENDIF
        GOTO 770
        ENDIF
C
C
C       STORE INFORMATION ON FILE
C
C       MOVE TO EOF
C
        CALL POSNT(IDCB1,IERR,KREC+1,1)
        CALL POSNT(IDCB2,IERR,KREC+1,1)
C
C       INCREMENT RECORD NUMBER
C
        KREC=KREC+1
C
C       STORE RECORD #
C
        IREC(N1)=KREC
C
        IF(IERR.LT.0) WRITE(LUOT,50)IERR
C
C       WRITE CLASS ONE VECTORS TO FILE (THESE ARE FROM THE LEFT NODE)
C
        CALL WRITF(IDCB1,IERR,X1,2*NVEC*NELE)
        CALL WRITF(IDCB2,IERR,IN1,NVEC)
C
        IF(IERR.LT.0)WRITE(LUOT,50)IERR
C
C       INCREMENT RECORD #
C
        KREC=KREC+1
C
C       STORE RECORD #
C
        IREC(N2)=KREC
C
C       WRITE CLASS TWO VECTORS TO FILE (RIGHT NODE VECTORS)
C
        CALL WRITF(IDCB1,IERR,X2,2*NVEC*NELE)
        CALL WRITF(IDCB2,IERR,IN2,NVEC)
C
        IF(IERR.LT.0) WRITE(LUOT,50)IERR
C
C       CALCULATE THE STATUS OF EACH NODE
C
        DO 882 I=1,N2
        WORTH(I)=0.0
        PINDX=INDEX(I)
        IF(INDEX(I).NE.0) THEN
          PCLAS=0.0
          DO 881 J=1,KLAS
881       IF(PCLAS.LT.LL(J,I)) PCLAS=LL(J,I)
        IF(PCLAS.NE.0) WORTH(I)=PCLAS*ALOGT(PCLAS/PINDX)
        IF(INDEX(I)-PCLAS.EQ.1) WORTH(I)=0.0
        ENDIF
882       CONTINUE
24      FORMAT(3X,20I3,/)
26      FORMAT(13X,4(2X,1A2) )
C
```

```
C
C      THE INITIAL MINIMUM VALUE IS SET TO THE PROVIOUSLY CREATED LEFT
C      NODE. THIS IS DONE TO INSURE THAT A NODE WITH ITREE ROOTS OF 0
C      IS SELECTED.(EG. NO NODES ALREADY DONE CAN BE DONE AGAIN.)
C
       WMIN=WORTH(N1)
C
       DO 9933 I=2,N2
       IF(ITREE(I,1).EQ.0.AND.ITREE(I,2).EQ.0.AND.WORTH(I).LE.WMIN)THEN
         WMIN=WORTH(I)
         NODE=I
       ENDIF
9933   CONTINUE
       IF(ICMND.EQ.2HNX) ICMND=2HSQ
4923   FORMAT(15X,"THE NEXT NODE IS",I3,/)
       IF(N2.LT.KNODE.AND.WMIN.NE.0.0) WRITE(LU,4923) NODE
       IF(N2.LT.KNODE.AND.WMIN.NE.0.0) GOTO 4444
C
C      PURGE OLD FILES AND STORE NEW DATA IN THEM
C
100    CALL PURGE(IDCB,IERR,INAM1)
       CALL CREAT(IDCB,IERR,INAM1,500,3,0,-24)
       CALL WRITF(IDCB,IERR,KLAS,1)
       CALL WRITF(IDCB,IERR,N2,1)
       CALL WRITF(IDCB,IERR,LABL,KLAS)
       CALL WRITF(IDCB,IERR,LL,KLAS*KNODE)
       CALL WRITF(IDCB,IERR,ITREE,2*KNODE)
       CALL WRITF(IDCB,IERR,W,2*KNODE*NELE)
       CALL WRITF(IDCB,IERR,INDEX,KNODE)
       DO 91 NODE=1,N2
       INND=INDEX(NODE)
C
C      POSITION AND READ FILES
C
       CALL POSNT(IDCB1,IERR,IREC(NODE),1)
       CALL READF(IDCB1,IERR,XX)
C
C      WRITE NODE # AND # OF FEATURES AT THAT NODE
C
       CALL WRITF(IDCB,IERR,NODE,1)
       CALL WRITF(IDCB,IERR,NELE,1)
C
C
C      CREATE MEAN VECTOR AND COVARIANCE MATRIX
       CALL MVCOV(XX,MEAN,COV,C1,C2,NELE,INND,NVEC)
C
       CALL WRITF(IDCB,IERR,MEAN,NELE*2)
91     CALL WRITF(IDCB,IERR,COV,(NELE**2)*2)
       CALL CLOSE(IDCB1)
       CALL CLOSE(IDCB2)
       CALL CLOSE(IDCB)
       WRITE(LU,6119)(LABL(I),I=1,KLAS)
6119   FORMAT(57X,4(2X,1A2))
       DO 6117 I=1,N2
6117   WRITE(LU,6116) I,(ITREE(I,J),J=1,2),(LL(INUM,I),INUM=1,KLAS)
6116   FORMAT(10X,"NODE#",I3,5X,"LEFT-->",I3,5X,"RIGHT-->",I3,7X,
      $4(1X,I3))
       STOP
       END
C
```

```
C                        SUBROUTINE MVCOV
C
C
C
C               A ----- MEAN VECTOR
C               NFC --- DIMENSION OF VECTORS
C               C1.C2 - BUFFERS
C               C ----- COVARIANCE MATRIX
C               NDATA-- NUMBER OF DATA POINTS
C               X ----- INPUT VECTORS
C
C      SUBROUTINE TO COMPUTE MEAN VECTOR AND COVARIANCE MATRIX
C      BASED ON THE INCOMING TEXTURE MEASUREMENT VECTOR.X
C
       SUBROUTINE MVCOV(X,A,C,C1,C2,NFC,NDATA,NVEC)
       DIMENSION C(NFC,NFC),C1(NFC,NFC),C2(NFC,NFC),A(NFC)
       DIMENSION X(NVEC,NFC)
       DO 100 I=1,NFC
       DO 100 J=1,NFC
       C1(I,J)=0.0
  100  C2(I,J)=0.0
       DO 10 I=1,NFC
       A(I)=0.0
       Z=NDATA
       DO 20 J=1,NDATA
   20  A(I)=A(I)+X(J,I)
   10  A(I)=A(I)/Z
       DO 50 I=1,NFC
       DO 40 J=1,NFC
       C(I,J)=0.0
       DO 30 K=1,NDATA
   30  C1(I,J)=C1(I,J)+X(K,I)*X(K,J)
       C1(I,J)=C1(I,J)/Z
   40  CONTINUE
   50  CONTINUE
       DO 70 I=1,NFC
       DO 60 J=1,NFC
       C2(I,J)=A(I)*A(J)
   60  CONTINUE
   70  CONTINUE
       DO 90 I=1,NFC
       DO 80 J=1,NFC
       C(I,J)=C1(I,J)-C2(I,J)
   80  CONTINUE
   90  CONTINUE
       RETURN
       END
C
C      THIS SUBROUTINE TAKES A GROUP OF VECTORS AND SEPARATES THEM
C      INTO TWO CLASSES. THIS IS DONE WITH A K-MEANS ALGORITHM.
C
       SUBROUTINE CLUS(X,NELE,NVEC,X1,IND1,X2,IND2,NODE,N1,N2,MEAN
      $,OLDM,KNODE,NVE2,INB,IN1,IN2,W)
C
C           THIS SUBROUTINE TAKES AN INPUT VECTOR SET," X ", AND
C           CLASSIFIES IT'S MEMBERS INTO ONE OF TWO REGIONS
C           DEFINED BY "X1" AND "X2" VIA A K-MEANS ALG. THIS ROUTINE
C           ALSO RETURNS THE TWO MEAN VECTORS IN THE ARRAY "MEAN".
C
            DIMENSION X(NVE2,NELE),X1(NVE2,NELE),X2(NVE2,NELE)
```

```
              DIMENSION MEAN(2,NELE),OLDM(2,NELE),INB(NVE2)
              DIMENSION  IN1(NVE2),IN2(NVE2),W(NELE,KNODE)
          REAL MEAN
C
C             INITIAL VALUES ARE CHOSEN FOR THE MEAN
C
          DO 99 I=1,2
          DO 99 J=1,NELE
          MEAN(I,J)=0.0
99        OLDM(I,J)=0.0
          DO 10 IELE=1,NELE
          MEAN(1,IELE)=X(1,IELE)
10        MEAN(2,IELE)=X(2,IELE)
          OK=0.0
          DO 11 I=1,NELE
11        IF(MEAN(1,I)*W(I,NODE).NE.MEAN(2,I)*W(I,NODE))OK=1.0
C
C         IF THE MEANS ARE THE SAME, INCREMENT THE ELEMENTS OF THE
C         FIRST MEAN BY ONE.
C
          IF(OK.EQ.0.0) THEN
            DO 1001 I=1,NELE
1001        MEAN(1,I)=MEAN(1,I)+1.0
          ENDIF
C
C
C         INITIALIZE INDICES FOR THE TWO NEW CLASSES
C
100       IND1=0
          IND2=0
          DO 20 J=1,NVEC
          SQ1=0.0
          SQ2=0.0
          DO 30 I=1,NELE
          SQ1=SQ1+(X(J,I)*W(I,NODE)-MEAN(1,I)*W(I,NODE))**2
30        SQ2=SQ2+(X(J,I)*W(I,NODE)-MEAN(2,I)*W(I,NODE))**2
          SQ1=SQRT(SQ1)
          SQ2=SQRT(SQ2)
          IF ( SQ2 .LE. SQ1 ) GOTO 50
          IND1=IND1+1
          DO 40 I=1,NELE
          IN1(IND1)=INB(J)
40        X1(IND1,I)=X(J,I)
          GOTO 20
C
C
50        IND2=IND2+1
          DO 60 I=1,NELE
          IN2(IND2)=INB(J)
60        X2(IND2,I)=X(J,I)
C
20        CONTINUE
C
C
C         THE MEAN IS COPIED FOR COMPARISON
C
C
          DO 5 IELE=1,NELE
          OLDM(1,IELE)=MEAN(1,IELE)
5         OLDM(2,IELE)=MEAN(2,IELE)
```

```
C
C      A NEW MEAN IS CREATED FOR THE TWO NEW CLASSES
C
       DO 14 K=1,2
       DO 14 I=1,NELE
 14    MEAN(K,I)=0.0
       DO 15 I=1,IND1
       DO 15 J=1,NELE
 15    MEAN(1,J)=X1(I,J)/FLOAT(IND1)+MEAN(1,J)
       DO 16 I=1,IND2
       DO 16 J=1,NELE
 16    MEAN(2,J)=MEAN(2,J)+X2(I,J)/FLOAT(IND2)
C
C      IF THE LAST ITERATION IS THE SAME AS THE PRESENT VALUE
C      THEN THE SOLUTION HAS BEEN REACHED AND THE SUBR. RETURNS.
C
       FLAG=0.0
       DO 45 K=1,2
       DO 45 I=1,NELE
 45    IF(OLDM(K,I)*W(I,NODE).NE.MEAN(K,I)*W(I,NODE))FLAG=1.0
       IF (FLAG.EQ.1.0) GOTO 100
       RETURN
       END
C
C      THIS SUBROUTINE DOES A BINARY COUNT OF THE W-MASK. THE
C      SEQUENCE IS:        00                (ON INITIAL ENTRY)
C                          10
C                          01
C                          11
C                          00
C      FOR A TWO BIT MASK.
C
       SUBROUTINE SEQUN(WOLD,W,NODE,KNODE,NELE)
       DIMENSION W(NELE,KNODE),WOLD(NELE)
C
C      STORE OLD MASK
C
       DO 10 I=1,NELE
 10     WOLD(I)=W(I,NODE)
C
       IF(WOLD(1).EQ.0.0) THEN
         W(1,NODE)=1.0
        ELSE
         W(1,NODE)=0.0
       ENDIF
C
       DO 20 J=2,NELE
       ITOG=1
       DO 30 I=1,J-1
 30    IF(WOLD(I).EQ.0) ITOG=0
       IF(ITOG.EQ.1) THEN
         IF(WOLD(J).EQ.0) THEN
           W(J,NODE)=1
          ELSE
           W(J,NODE)=0
         ENDIF
       ENDIF
 20    CONTINUE
       RETURN
       END
```

END
9-87
DTIC