NASA TECHNICAL MEMORANDUM

8 0 - 1 0 0 4 2
TM-78227

NASA TM-78227

EVALUATION OF REGISTRATION, COMPRESSION, ANDD CLASSIFICATION ALGORITHMS – VOLUME 2 (DOCUMENTATION)

By R. Jayroe, R. Atkinson, L. Callas, J. Hodges, B. Gaggini, and J. Peterson

(E80-10042) EVALUATION OF REGISTRATION, COMPRESSION, AND CLASSIFICATION ALGORITHMS. VOLUME 2: DOCUMENTATION (NASA) 331 p HC A15/MF A01 CSCL 08B G3/43

N80-16392

Unclas 00042

February 1979

NASA

George C. Marshall Space Flight Center
Marshall Space Flight Center, Alabama

TECHNICAL REPORT STANDARD TITLE PAGE

1. REPORT NO. NASA TM-78227	2. GOVERNMENT ACCESSION NO.	3. RECIPIENT'S CATALOG NO.
4. TITLE AND SUBTITLE Evaluation of Registration, Compression and Classification Algorithms. Volume 2 (Documentation)		5. REPORT DATE February 1979
		6. PERFORMING ORGANIZATION CODE
7. AUTHOR(S) R. Jayroe, R. Atkinson, L. Callas, J. Hodges, B. Gaggini, and J. Peterson		8. PERFORMING ORGANIZATION REPORT #
9. PERFORMING ORGANIZATION NAME AND ADDRESS George C. Marshall Space Flight Center Marshall Space Flight Center, Alabama 35812		10. WORK UNIT NO.
		11. CONTRACT OR GRANT NO.
12. SPONSORING AGENCY NAME AND ADDRESS National Aeronautics and Space Administration Washington, D. C. 20546		13. TYPE OF REPORT & PERIOD COVERED Technical Memorandum
		14. SPONSORING AGENCY CODE

15. SUPPLEMENTARY NOTES

Prepared by the Data Systems Laboratory.

16. ABSTRACT

Volume I examines the effects that are produced by three registration and seven compression approaches on Landsat imagery and on results obtained from three classification approaches. The registration, compression and classification algorithms were selected on the basis that such a group would include most of the different and commonly used approaches. The results of the investigation indicate clear-cut, cost-effective choices for registering, compressing and classifying multispectral imagery. Volume 2 is a programmer's user manual containing IBM-360/75 Fortran listings of the algorithms used in the investigation.

17. KEY WORDS		18. DISTRIBUTION STATEMENT Unclassified-Unlimited	
19. SECURITY CLASSIF. (of this report) Unclassified	20. SECURITY CLASSIF. (of this page) Unclassified	21. NO. OF PAGES 329	22. PRICE NTIS

MSFC - Form 3292 (Rev. December 1972)

For sale by National Technical Information Service, Springfield, Virginia 22151

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS

CHAPTER I

INTRODUCTION

This volume documents the computer programs which were used to implement and evaluate the image processing algorithms described in Volume I. All of the programs described were run on an IBM 360/75 computer, ana core requirements are given in units of 8-bit bytes. The Landsat-1 and -2 data used as input was formatted into bytes, such that the four spectral measurements appear in one full word. The data sets used were seasonal passes covering a LACIE sample segment of 117 by 196 pixels, and a 1200 pixel square segment containing Mobile, Alabama.

CHAPTER II

EVALUATION OF PROCESSING EFFECTS

AVERAGE UNCERTAINTY (ENTROPY)

I. <u>NAME</u>

ENTRPY

II. <u>DESCRIPTION</u>

The average uncertainty (entropy) of a data set X is given by

$$H(X) = -\sum_{i=1}^{N} P(X_i) \log_2 P(X_i)$$

where $P(X_i)$ is the probability of occurrence of the ith event in data set X. This subroutine computes the entropy of a data set X.

III. <u>CALLING SEQUENCE</u>

CALL ENTRPY (IX, NREC, NEL, NDEVI, IBAND, XMEAN, SIGMAX, XNTRPY)

where

IX is an array into which the data set is read;

NREC and NEL are the number of records and the number of pixels (bytes) per record in the data set;

NDEVI is the logical unit number of the data set;

IBAND is the band (one out of four) of the data set; and,

XMEAN, SIGMAX, and XNTRPY are outputs of the subroutine giving the mean, variance, and entropy, respectively, of the data set.

IV. <u>INPUT/OUTPUT</u>

1. INPUT

   The input to this program is a sequential data set on logical unit NDEVI, having NREC records each NEL 4-band pixels long, stored in unformatted FORTRAN mode.

2. OUTPUT

   The outputs of the program are the three variables XMEAN, SIGMAX, and XNTRPY denoting mean, variance, and entropy of the data set.

V. DESCRIPTION OF SUBROUTINES

No subroutines are required by this routine.

VI. PERFORMANCE SPECIFICATIONS

1. STORAGE

   This subroutine is 1594 bytes long.

2. EXECUTION TIME

   For a data set of 112 records, each 192 elements long, it takes about 1/3 second on the IBM 360/75 computer to compute the mean, variance, and entropy of one band of data.

VII. METHOD

A simplified program for computing the mean, variance, and entropy of a data set is shown in Figure 1. First, a histogram of the data is obtained giving the number of occurrences of each intensity. Then, the probability of each intensity is found by dividing the number of occurrences by the total number of pixels. Once the probabilities are determined, the entropy is computed from the relationship given in Section II. The mean, $\bar{X}$, is determined from

$$\bar{X} = \frac{1}{N} \sum_{i=1}^{N} X_i$$

where N is the total number of pixels (NREC X NEL) and the X_i's are the intensities.

The variance, S^2, is found from

$$S^2 = \frac{1}{N-1} \sum_{i=1}^{N} (X_i - \bar{X})^2.$$

VIII. COMMENTS

None

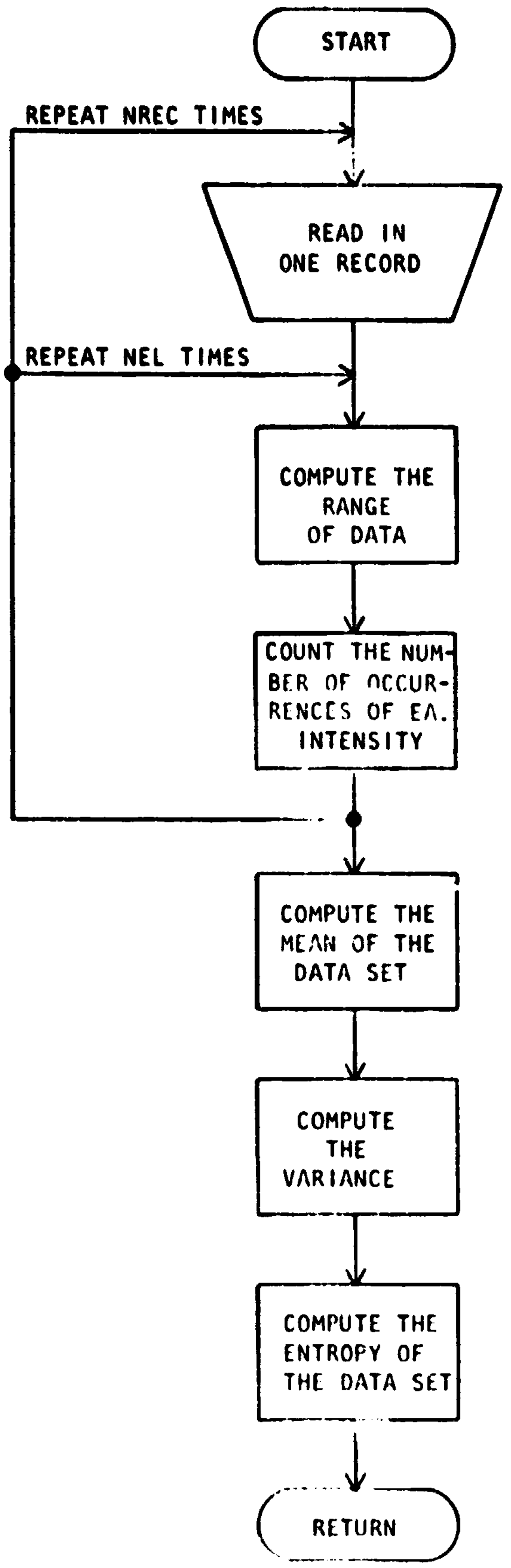

START
REPEAT NREC TIMES
READ IN ONE RECORD
REPEAT NEL TIMES
COMPUTE THE RANGE OF DATA
COUNT THE NUMBER OF OCCURRENCES OF EA. INTENSITY
COMPUTE THE MEAN OF THE DATA SET
COMPUTE THE VARIANCE
COMPUTE THE ENTROPY OF THE DATA SET
RETURN

Figure 1. A Simplified Flow Diagram for Computing the Mean, Variance, and Entropy of a Data Set

IX. TESTS

This program has been tested on the LACIE data (112 lines x 192 pixels/line). For the 10/22/75 pass, band 1, the mean and variance were found to be 25.1 and 12.4, respectively. When the logarithm to the base of two was used, the entropy was 3.7 bits.

X. LISTINGS

The listing for ENTRPY is attached at the end of this section.

```
      SUBROUTINE ENTRPY(IX,NREC,NEL,NDEVI,IBAND,XMEAN,SIGMAX,XNTRPY)
C
C     THIS SUBROUTINE COMPUTES THE MEAN, VARIANCE, AND ENTROPY
C     OF A DATA SET
C
C     ...........................................................................
C     ***************************************************************************
C
      LOGICAL*1 IX(4,NEL)
      DIMENSION IH(128)
      DATA NPTS /128/
C
C     COMPUTE THE HISTOGRAM OF THE DATA
      DO 5 I=1,NPTS
    5 IH(I) = 0
      DO 10 I=1,NREC
      READ(NDEVI) IX
      DO 10 J=1,NEL
      M = IX(IBAND,J)+1
      IH(M)=IH(M)+1
   10 CONTINUE
C
C     COMPUTE THE MEAN AND VARIANCE OF THE DATA
      N=0
      NXL=0
      NXM=0
      DO 20 I=1,NPTS
      N = N + IH(I)
      NXL = NXL + IH(I)*(I-1)
      NXM = NXM + IH(I)*(I-1)**2
   20 CONTINUE
      XMEAN = FLOAT(NXL)/FLOAT(N)
      SIGMAX = (NXM-N*XMEAN**2) / (N-1)
C
C     COMPUTE THE AVERAGE UNCERTAINTY (ENTROPY)
      SUM=0.0
      ALN2=ALOG10(2.0)
      DO 30 I=1,NPTS
      IF (IH(I).EQ.0) GO TO 30
      PROBX=FLOAT(IH(I))/FLOAT(N)
      SUM=SUM+PROBX*(ALOG10(PROBX)/ALN2)
   30 CONTINUE
      XNTRPY=-SUM
      RETURN
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

AVERAGE INFORMATION TRANSFERRED FROM X TO Y (TRANSINFORMATION)

I. **NAME**

TRNSIN

II. **DESCRIPTION**

The average information transferred from X to Y (transinformation) is given by

$$I(X;Y) = \sum_{i=1}^{N} \sum_{j=1}^{M} P(X_i,Y_j) \log_2 \frac{P(X_i/Y_j)}{P(X_i)}$$

where $P(X_i,Y_j)$ is the joint probability of occurrence of the i^{th} event in data set X and the j^{th} event in the data set Y. $P(X_i/Y_j)$ is the conditional probability of occurrence of the i^{th} event in data set X given that the j^{th} event in data set Y has occurred. $P(X_i)$ is the probability of occurrence of the i^{th} event in data set X. This subroutine computes the transinformation between a data set X and a data set Y (compressed data set X).

TRNSIN computes the average percent deviation and the average of the differences squared between the data sets X and Y. These relationships are determined from

$$AVPDEV = \frac{100}{N} \sum_{i=1}^{N} \left| \frac{Y_i - X_i}{X_i} \right|$$

where N is the number of elements in each data set and

$$AVDFSQ = \frac{1}{N} \sum_{i=1}^{N} (Y_i - X_i)^2.$$

Other relationships available from this subroutine are the joint probability of X and Y, the marginal probability of X, the marginal probability of Y, the conditional probability of X given Y, the mutual information between X and Y, and the summation over X of the conditional probability of X given Y times the mutual information between X and Y.

III. CALLING SEQUENCE

CALL TRNSIN (IX, IY, NREC, NEL, NDEVIX, NDEVIY, IBAND1, IBAND2, XINF, MAX1, MIN1, AVDFSQ, N)

where

IX and IY are arrays into which the data sets X and Y are read;
NREC and NEL are the number of records and the number of pixels (bytes) per record in the data sets;
NDEVIX and NDEVIY are the logical unit numbers of the data sets;
IBAND1 and IBAND2 are the bands (channels) of the data sets; and,
XINF, MAX1, MIN1, AVDFSQ, and N are outputs of the subroutine giving the transinformation, data limits, average of the differences squared, and number of pixels, respectively.

IV. INPUT/OUTPUT

1. INPUT

The inputs to this program are two sequential data sets. One data set is the original data set X on logical unit NDEVIX and the other data set is the compressed data set Y on logical unit NDEVIY. Each data set consists of NREC records each NEL elements long stored in unformatted FORTRAN mode.

2. OUTPUT

The outputs of this program are the computed quantities described in Section II above.

V. DESCRIPTION OF SUBROUTINES

No subroutines are called by this program.

VI. PERFORMANCE SPECIFICATIONS

1. STORAGE

This subroutine is 68116 bytes long and the common block PROB is 198,144 bytes.

2. EXECUTION TIME

For a data set of 112 records, each 192 elements long, it takes about 2.2 seconds to execute the program.

VII. METHOD

A simplified flow diagram of subroutine TRNSIN is shown in Figure 2. This subroutine finds the ranges of the data (min. to max.) and constructs a joint histogram of the data sets X and Y. Each entry of the two-dimensional histogram contains the number of occurrences of the intensity of each pixel. The average percent deviation and the average of the differences squared between the two data sets are then computed. The subroutine also determines the probabilities P(X,Y), P(X), P(Y), and P(X/Y) which are used to obtain the mutual information and transinformation of the data sets X and Y.

VIII. COMMENTS

None.

IX. TESTS

This program was tested using LACIE data obtained by LANDSAT-1 on May 6, 1976 for data set X and this same data (compressed at 1 bit/pixel using the Adaptive Differential Pulse Code Modulation technique) for data set Y. The following results were obtained for band 1:

Average Percent Deviation = 5.32
Average of the Differences Squared = 8.40
Transinformation = 1.67 Bits/Symbol

X. LISTINGS

The listing for TRNSIN is attached at the end of this section.

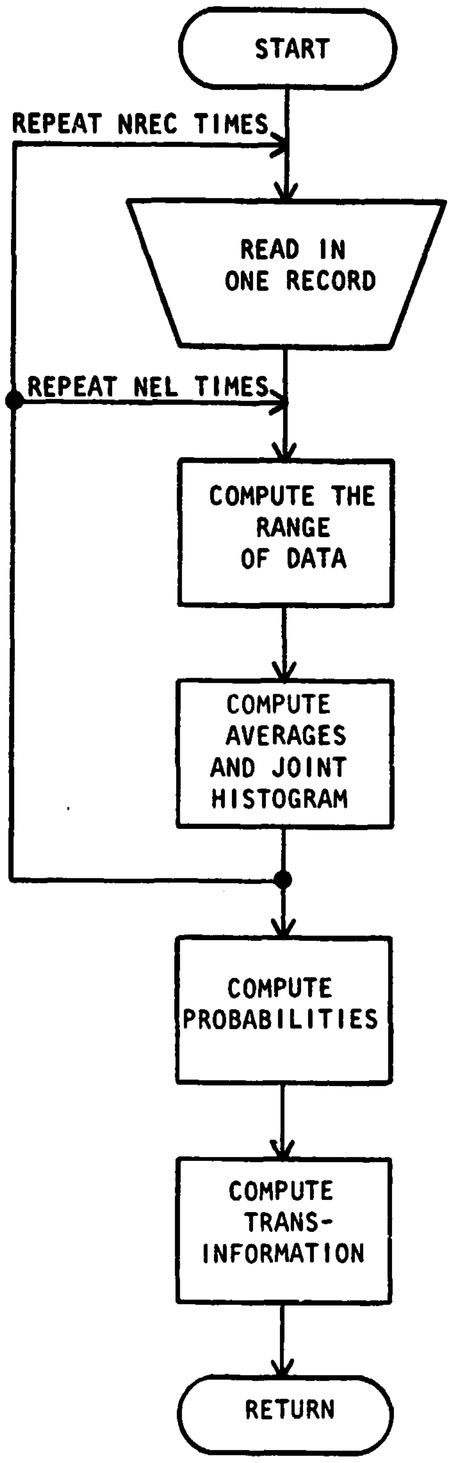

START
REPEAT NREC TIMES
READ IN ONE RECORD
REPEAT NEL TIMES
COMPUTE THE RANGE OF DATA
COMPUTE AVERAGES AND JOINT HISTOGRAM
COMPUTE PROBABILITIES
COMPUTE TRANS-INFORMATION
RETURN

Figure 2. A Simplified Flow Diagram of Subroutine TRNSIN

```
      SUBROUTINE TRNSIN (IX, IY, NREC, NEL, NDEVIX, NDEVIY, IBAND1,
     .                    IBAND2, XINF, MAX1, MIN1, AVDFSQ, N)
C
C     THIS SUBROUTINE COMPUTES THE AVERAGE INFORMATION TRANSFERRED FROM
C     X TO Y (TRANSINFORMATION)
C
C     ..........................................................................
C     ************************************************************************
C
      LOGICAL*1 IX(4,NEL), IY(4,NEL)
      DIMENSION IH(128,128)
      INTEGER OBS, EXP, DIFSQ
      COMMON/PROB/PRBXAY(128,128),PRBX(128),PRBY(128),PRBXGY(128,128),
     .   XMUT(128,128),XINFYI(128)
C
      DO 10 I=1,128
      DO 10 J=1,128
      IH(I,J)=0
   10 CONTINUE
      SUMABS=0.0
      DIFSQ = 0
C
      DO 15 I=1,NREC
      READ(NDEVIX) IX
      READ(NDEVIY) IY
C
      DO 15 J=1,NEL
      L=IX(IBAND1,J)+1
      M=IY(IBAND2,J)+1
      IH(L,M)=IH(L,M)+1
C
      EXP=IX(IBAND1,J)
      OBS=IY(IBAND2,J)
      IF (EXP.EQ.0) GO TO 11
      DIFOEX = (OBS-EXP)/FLOAT(EXP)
      GO TO 12
   11 DIFOEX=OBS
C
   12 SUMABS = SUMABS+ABS(DIFOEX)
      DIFSQ = DIFSQ+(OBS-EXP)**2
   15 CONTINUE
C
      N = NREC*NEL
      AVPDEV = 100.0*SUMABS/N
      AVDFSQ = FLOAT(DIFSQ)/N
C
C     FIND THE RANGE OF THE DATA (MIN TO MAX)
      DO 16 NR=1,128
      DO 16 J=NR,128
      IF (IH(J,NR).NE.0) GO TO 17
      IF (IH(NR,J).NE.0) GO TO 17
   16 CONTINUE
   17 MIN = NR-1
      DO 18 NR1=1,128
      NR = 129-NR1
      DO 18 J=1,NR
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      IF (IM(NR,J).NE.0) GO TO 19
      IF (IM(J,NR).NE.0) GO TO 19
   18 CONTINUE
   19 MAX = NR-1
      MAX1=MAX-MIN+1
      MIN1=MIN
C
C     COMPUTE JOINT PROBABILITY OF X AND Y
      DO 20 I=1,MAX1
      DO 20 J=1,MAX1
   20 PRBXAY(I,J) = FLOAT(IM(I+MIN,J+MIN))/N
C
C     COMPUTE MARGINAL PROBABILITY OF X
      DO 25 I=1,MAX1
      PRBX(I)=0.0
   25 CONTINUE
      DO 30 I=1,MAX1
      DO 30 J=1,MAX1
      PRBX(I)=PRBX(I)+PRBXAY(I,J)
   30 CONTINUE
C
C     COMPUTE MARGINAL PROBABILITY OF Y
      DO 35 I=1,MAX1
      PRBY(I)=0.0
   35 CONTINUE
      DO 40 J=1,MAX1
      DO 40 I=1,MAX1
      PRBY(J)=PRBY(J)+PRBXAY(I,J)
   40 CONTINUE
C
C     COMPUTE CONDITIONAL PROBABILITY OF X GIVEN Y
      DO 50 J=1,MAX1
      DO 50 I=1,MAX1
      IF(PRBY(J).LE.0.0) GO TO 45
      PRBXGY(I,J)=PRBXAY(I,J)/PRBY(J)
      GO TO 50
   45 PRBXGY(I,J)=0.0
   50 CONTINUE
C
C     COMPUTE MUTUAL INFORMATION BETWEEN X AND Y
      ALN2=ALOG10(2.0)
      DO 65 I=1,MAX1
      DO 65 J=1,MAX1
      IF(PRBXGY(I,J).LE.0.0) GO TO 60
      IF(PRBX(I).LE.0.0) GO TO 60
      XMUT(I,J)=ALOG10(PRBXGY(I,J)/PRBX(I))/ALN2
      GO TO 65
   60 XMUT(I,J)=0.0
   65 CONTINUE
C
C     COMPUTE AVERAGE INFORMATION TRANSFERRED FROM X TO Y
C     (TRANSINFORMATION)
      XINF=0.0
      DO 85 I=1,MAX1
      DO 85 J=1,MAX1
      XINF=XINF+PRBXAY(I,J)*XMUT(I,J)
   85 CONTINUE
C
C     COMPUTE THE SUMMATION OVER X OF THE CONDITIONAL PROBABILITY OF
C     X GIVEN Y TIMES THE MUTUAL INFORMATION BETWEEN X AND Y
      DO 70 I=1,MAX1
      XINFYI(I)=0.0
   70 CONTINUE
      DO 75 J=1,MAX1
      DO 75 I=1,MAX1
      XINFYI(J)=XINFYI(J)+PRBXGY(I,J)*XMUT(I,J)
   75 CONTINUE
C
      WRITE(6,100) MIN, MAX
      WRITE(6,350) N
      WRITE(6,200) AVPDEV
      WRITE(6,950) IBAND1,XINF
      RETURN
C
  100 FORMAT (/ 1X,'RANGE OF DATA=',I5,2X,'TO',I5)
  200 FORMAT (/ 1X,'AVERAGE PERCENT DEVIATION =',F12.5)
  350 FORMAT (/ 1X,'TOTAL NUMBER OF OCCURRENCES=',I6)
  950 FORMAT (/ 1X,'TRANSINFORMATION FOR BAND',I2,' =',F12.5,
     *  '  BIT/SYMBOL')
      END
```

CHI-SQUARE STATISTIC

I. **NAME**

XSQ

II. **DESCRIPTION**

The chi-square statistic for two data sets X and Y is defined by

$$X^2 = \sum_{i=1}^{N} \frac{\left(o_i - e_i\right)^2}{e_i}$$

where o_i is the observed frequency and e_i is the expected frequency of occurrences of events from data sets X and Y. The total frequency of occurrence is N, and

$$\sum o_i = \sum e_i = N.$$

The values for o_i and e_i are determined from

$$o_i = NP(Y_i) \text{ and}$$
$$e_i = NP(X_i)$$

where $P(Y_i)$ is the marginal probability of Y_i and $P(X_i)$ is the marginal probability of X_i. The chi-square statistic is obtained by summing the values indicated above for $e_i > 5$. Those values of e_i equal to or smaller than 5 are pooled with previous values of e_i. Corresponding values of o_i must also be pooled. The number of degrees of freedom is equal to the number of e_i or o_i after pooling has been considered.

For comparison with tabulated values of the chi-square statistic, XSQ approximates tabulated values by

$$X^2 = 1 - \frac{2}{9} N + Z\sqrt{2/9\ N}$$

where Z is the standardized variable of a distribution with values of 1.28, 1.645, 2.33, 2.58, and 2.88 representing confidence levels of 90%, 95%, 99%, 99.5%, and 99.8%, respectively.

III. CALLING SEQUENCE

CHISQ = XSQ (N, MAX1)

where

N is the number of elements in data sets X and Y.

MAX1 is the range of the data (minimum to maximum).

IV. INPUT/OUTPUT

1. INPUT

The inputs to the function XSQ are obtained from the common/PROB/ which is generated by subroutine TRNSIN. These inputs are the marginal probability of X, the marginal probability of Y, the total number of elements, N, and the range of the data, MAX1.

2. OUTPUT

The output of the function is the chi-square statistic.

V. DESCRIPTION OF SUBROUTINES

No subroutines are called by this program.

VI. PERFORMANCE SPECIFICATIONS

1. STORAGE

This function is 2224 bytes long.

2. EXECUTION TIME

About 5 seconds to execute the program on an IBM 360/75 computer.

VII. METHOD

Figure 3 shows a simplified flow diagram of the function XSQ. The observed and expected frequencies of occurrence are computed from the marginal probabilities which are obtained from subroutine TRNSIN. Values of the expected frequency less than 5 are pooled together. The chi-square statistic for the data sets X and Y, the number of degrees of freedom, NDF, and approximations to tabulated values are computed using the relationships given in Section II.

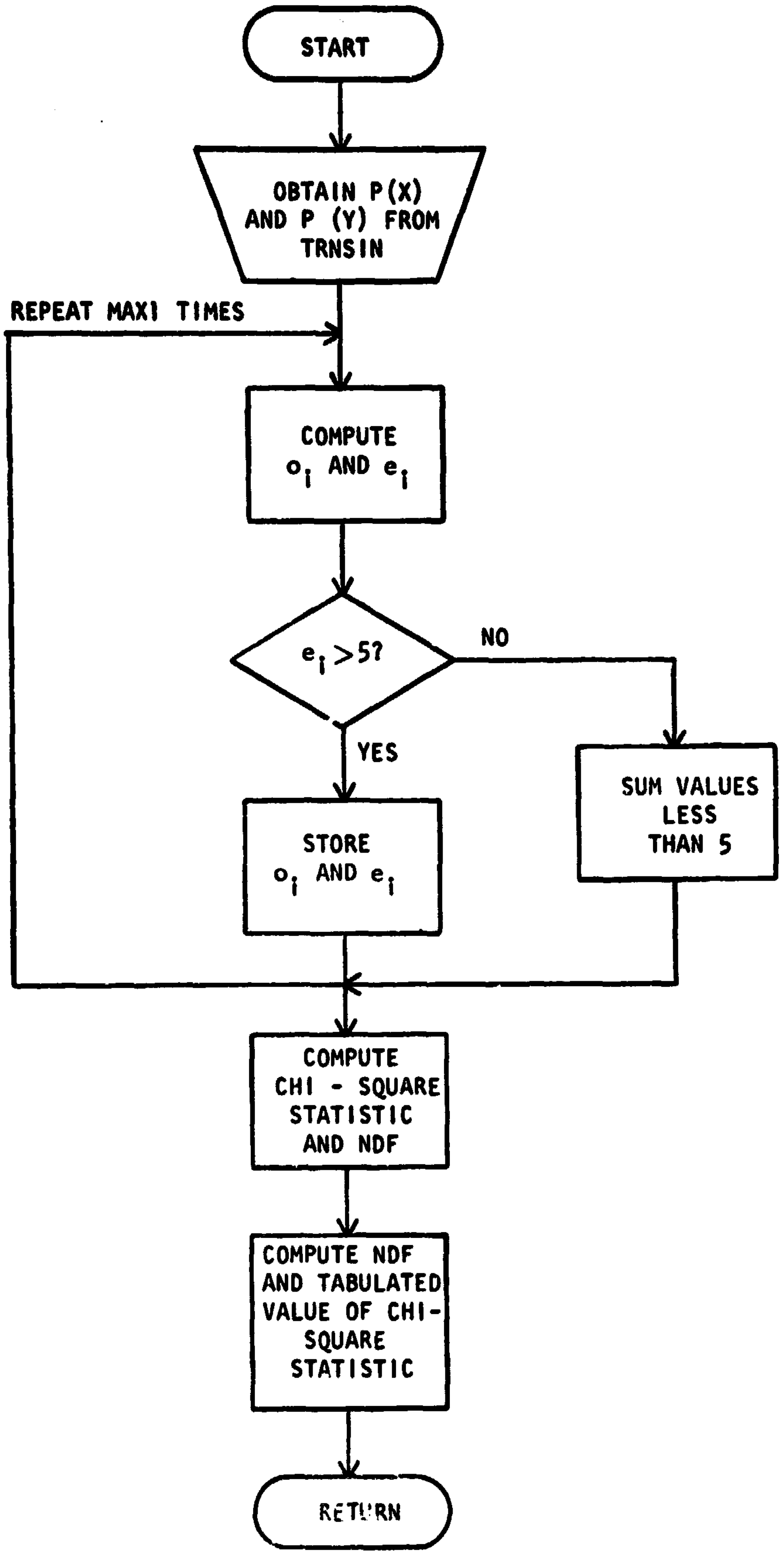

START
OBTAIN P(X) AND P (Y) FROM TRNSIN
REPEAT MAXI TIMES
COMPUTE o_i AND e_i
$e_i > 5?$
NO
YES
STORE o_i AND e_i
SUM VALUES LESS THAN 5
COMPUTE CHI - SQUARE STATISTIC AND NDF
COMPUTE NDF AND TABULATED VALUE OF CHI-SQUARE STATISTIC
RETURN

Figure 3. Simplified Flow Diagram of XSQ

VIII. COMMENTS

None

IX. TESTS

The chi-square statistic for LACIE data obtained by Landsat-1 on May 6, 1976 (representing expected values), and the same data compressed using the ADPCM technique (representing observed values) was found to be 9501.64. Normalizing and comparing with the tabulated values showed that the hypothesis that the compressed and uncompressed data sets are the same should be rejected at each of the confidence levels.

X. LISTINGS

The listing for XSQ is attached at the end of this section.

```
      FUNCTION XSQ(N,MAX1)
C
C     THIS FUNCTION COMPUTES THE CHI-SQUARE STATISTIC
C
C     ..............................................................
C     **************************************************************
C
      DIMENSION FO(128),FE(128),Z(5)
      COMMON/PROB/PROBXAY(128,128),PROBX(128),PROBY(128),PROBXGY(128,128),
     *   XMUT(128,128),XINFYI(128)
      DATA Z/1,28,1,645,2,33,2,58,2,88/
C
      M=0
      SUMEXP=0,0
      SUMOBS=0,0
      FN=FLOAT(N)
C
C     COMPUTE THE OBSERVED AND EXPECTED VALUES
      DO 20 I=1,MAX1
      SUMOBS=SUMOBS+FN*PROBY(I)
      SUMEXP=SUMEXP+FN*PROBX(I)
      IF(SUMEXP,GT,5,0) GO TO 10
      GO TO 20
   10 M=M+1
      L=I
C
C     STORE THE OBSERVED AND EXPECTED VALUES
      FO(M)=SUMOBS
      FE(M)=SUMEXP
      SUMEXP=0,0
      SUMOBS=0,0
   20 CONTINUE
      IF(L,EQ,MAX1) GO TO 40
C
C     ADD LAST VALUES LESS THAN FIVE TO PREVIOUS CLASS
      IF(M,EQ,0) GO TO 70
      FO(M)=FO(M)+SUMOBS
      FE(M)=FE(M)+SUMEXP
C
C     COMPUTE CHI-SQUARE STATISTIC
   40 XSQ=0,0
      WRITE(6,600)(FE(J),J=1,M)
      DO 50 I=1,M
      DIFSQ=(FO(I)-FE(I))**2
      DIFOEX=DIFSQ/FE(I)
      XSQ=XSQ+DIFOEX
   50 CONTINUE
      WRITE(6,100) XSQ
      NDF=M-1
      WRITE(6,200) NDF
      FNDF=FLOAT(NDF)
      XCALC=(XSQ/FNDF)**(1,0/3,0)
      WRITE(6,300) XCALC
      DO 60 I=1,5
      TWO9N=2,0/(9,0*FNDF)
      XTAB=1,0-TWO9N+Z(I)*SQRT(TWO9N)
      WRITE(6,400) XTAB,Z(I)
   60 CONTINUE
      GO TO 80
   70 WRITE(6,500)
   80 RETURN
C
  100 FORMAT(//,1X,'CHI-SQUARE=',F12.5)
  200 FORMAT(//,1X,'DEGREES OF FREEDOM=',I12)
  300 FORMAT(//,1X,'X**2/N**(1/3)=',F12.5,/)
  400 FORMAT(1X,'1-2/9N+Z(2/9N)**(1/2)=',F12.5,2X,'AT Z=',F12.5)
  500 FORMAT(1X,'THE SUM OF ALL CLASSES IS LESS THAN FIVE')
  600 FORMAT(1X,10F10.1)
      END
```

MULTIDIMENSIONAL HISTOGRAM OF FEATURE VECTORS

I. NAME

HASH

II. DESCRIPTION

The routine obtains the histogram of the four-dimensional vectors representing Landsat pixels in a scene. Using the histogram, the mean, variance, and entropy are computed.

III. CALLING SEQUENCE

CALL HASH (A, N, NPOP, IFEAT, NREC, NPIX, JFAC, IMOD, JMOD)

where

A is the input buffer,

N is an array for storing the table of vectors,

NPOP is the array of frequencies of occurrences of the vectors,

IFEAT is the maximum number of different vectors allowed (determines dimensions of N and NPOP),

NREC is the number of records in the data set,

NPIX is the number of pixels per record,

JFAC is the multiplier used in determining the table location of a vector,

IMOD is the divisor used, and

JMOD is the base used.

IV. INPUT/OUTPUT

1. INPUT

The input data should be on unit 10 in bytes.

2. OUTPUT

The program prints intermediate vector counts for every 100th and 1000th input vector, the vectors that occur at least 1000 times, and the number and percentages of vectors that occur 1 to 99 times, and multiples of 100 and 1000 times.

V. DESCRIPTION OF SUBROUTINES

No other subroutines are called.

VI. PERFORMANCE SPECIFICATIONS

1. STORAGE REQUIREMENTS

The subroutine requires 3770 bytes of storage. The storage required in the calling program is twice the maximum number of vectors allowed, in words (for the arrays N and NPOP), and the input buffer array.

2. EXECUTION TIME

The processing rate varies greatly with the distribution of vectors, but is approximately 9000 input vectors per second. The execution time increases if the length of the frequency table (NPOP) is not somewhat greater than the number of vectors found.

VII. METHOD

A straightforward table of occurrences can not be used because the maximum possible number of vectors from Landsat data is $128 \times 128 \times 128 \times 64 = 134,217,728$. Consequently, the divisor, base, and multiplier are applied to a vector to compute a location in a shorter table. Each component of the vector is divided by the divisor and the remainder for each component obtained. Using the specified base, the remainders are used to obtain a four digit number. Since this number is not unique with respect to input vectors, the number and hence the available table locations are multiplied by the multiplier. This final number is the table location at which the search for new vectors begins. Additional details and results are given in reference (1) and a flow chart in Figure 4.

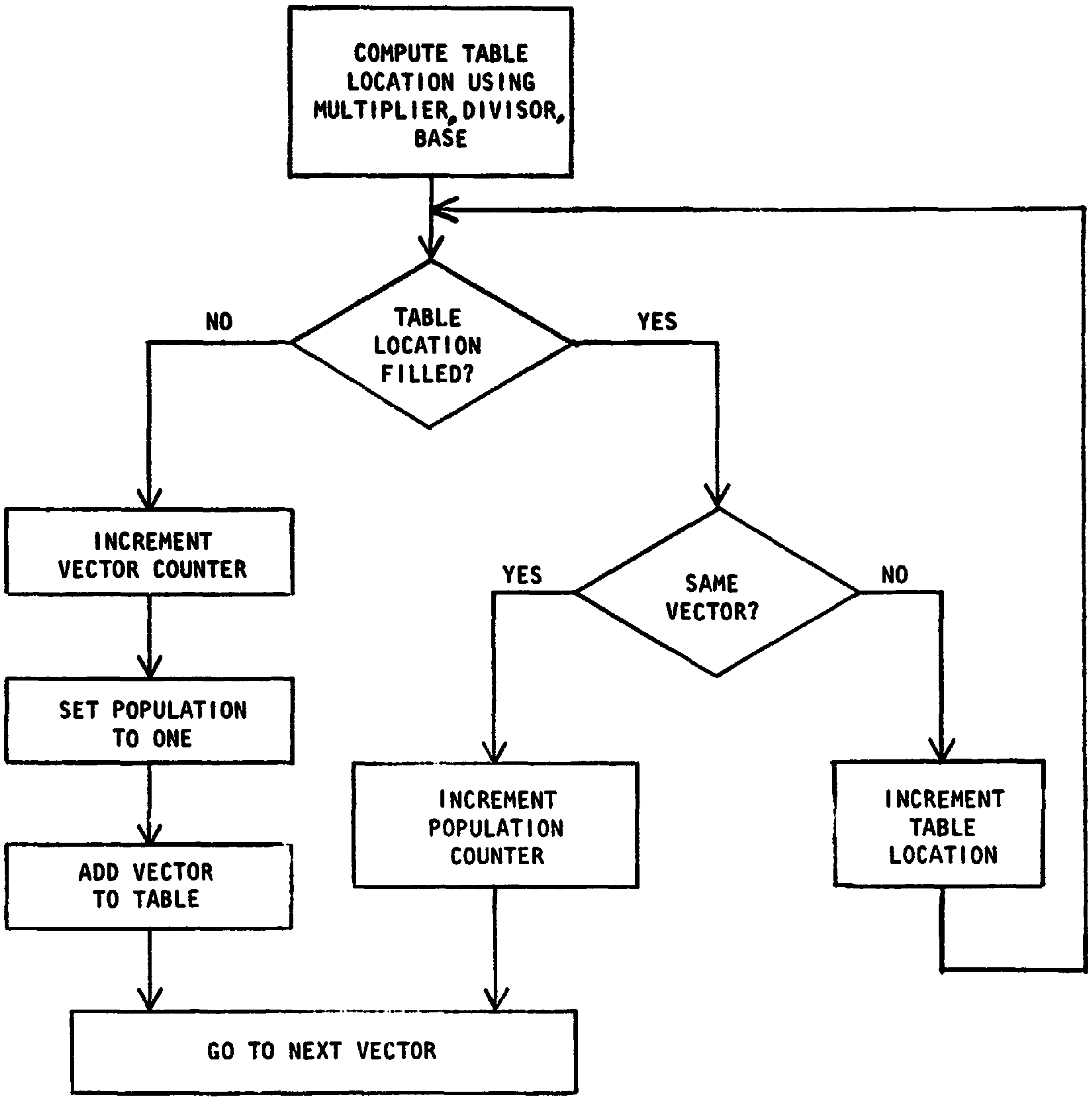


Figure 4. Flow Chart for Creating Vector Table in Subroutine HASH

VIII. COMMENTS

The storage required can be estimated from the following relation between the number of vectors (N) and the divisor, base, and multiplier:

$$N \geq M \cdot (D-1) \cdot (1+B+B^2+B^3) \text{ with } B \geq D \text{ and } M \geq 1.$$

For the Mobile Bay data set, the optimum values were found to be $D = 11$, $B = 12$, $M = 3$.

IX. TESTS

The algorithm has been compared with other techniques on test data sets.

X. LISTING

A listing of the routine follows.

```
      SUBROUTINE HASH (A, N, NPOP, IFEAT, NREC, NPIX, JFAC, IMOD, JMOD)
C
C     THE PROGRAM COMPUTES A 4 DIMENSIONAL HISTOGRAM.
C     THE HISTOGRAM IS USED TO COMPUTE THE MEAN, VARIANCE, AND ENTROPY.
C
C     THE VECTORS ARE READ IN ONE RECORD AT A TIME USING THE VARIABLE A,
C     N(I)=COMPONENTS OF THE I'TH VECTOR IN THE TABLE
C     NPOP(I)=THE NUMBER OF OCCURRENCES OF THE ITH VECTOR IN THE TABLE
C     IFEAT=MAXIMUM NUMBER OF DIFFERENT VECTORS ALLOWED
C     NREC=NUMBER OF RECORDS TO EXAMINE
C     NPIX=NUMBER OF PIXELS PER RECORD
C     JFAC=MULTIPLIER
C     IMOD=DIVISOR
C     JMOD=BASE
C     LM,IM=THE COMPONENTS OF A VECTOR (LOGICAL*1 AND INTEGER)
C     NBAND=NUMBER OF SPECTRAL IMAGES OR VECTOR DIMENSION
C     KNT=A VARIABLE USED TO COUNT THE NUMBER OF PICTURE ELEMENTS(PELS)
C     KOUNT=A VARIABLE USED TO COUNT THE NUMBER OF DIFFERENT VECTORS
C
C     ....................................................................
C     ********************************************************************
C
      INTEGER A(NPIX)
      DIMENSION N(IFEAT), NPOP(IFEAT), NN(200), IH(128)
      LOGICAL*1 LM(4)
      EQUIVALENCE (IM, LM(1))
      DATA NBAND /4/
C
C     INITIALIZE ALL VECTOR POPULATIONS TO ZERO
      DO 85 I=1,IFEAT
   85 NPOP(I)=0
      KOUNT=0
      MFAC=100
      KLU = 0
      KSS = 0
      KMS = 0
      WRITE (6,750) NPIX, NREC, JFAC, IMOD, JMOD
      WRITE (6,710)
C
C     LOOP 500 READ IN NREC RECORDS
      DO 500 I=1,NREC
C
C     READ IN ONE RECORD
      READ (10) A
C
C     ACCUMULATE FOUR DIMENSIONAL HISTOGRAM FOR EACH RECORD
      DO 200 J=1,NPIX
C
C     GET EACH VECTOR OUT OF A AND PUT INTO M
      IM = A(J)
C
C     4 DIMENSIONAL HISTOGRAM ROUTINE (COMBINATION TABLE LOOK UP/SEARCH
C     PROCEDURE)
C
C     COMPUTE TABLE LOCATION FROM VECTOR COMPONENTS
      L=0
```

RIGINAL PAGE IS
OF POOR QUAL

```
      DO 99 NB=1,NBAND
      NXX = LM(NB)
   99 L = L*JMOD + MOD(NXX,IMOD)
      L = JFAC*L + 1
      LL=L
C
C     CHECK FOR EMPTY TABLE LOCATION
  100 IF(NPOP(L).NE.0)GO TO 110
C
C     HAVE FOUND A NEW VECTOR, INCREMENT VECTOR COUNTER
      KOUNT=KOUNT+1
C
C     CHECK TO SEE IF LIMIT ON NUMBER OF VECTORS IS EXCEEDED
      IF (KOUNT.GT.IFEAT) GO TO 400
C
C     SET POPULATION OF NEW VECTOR TO ONE
      NPOP(L)=1
C
C     PUT NEW VECTOR INTO TABLE
      N(L) = IM
      IF (LL.EQ.L) NLU = NLU + 1
C
C     PRINT PEL NUMBER FOR EVERY 100TH AND 1000TH VECTOR
      IF (MOD(KOUNT,MFAC).NE.0) GO TO 200
      KNT = NPIX*(I-1) + J
      X = FLOAT(KNT)/FLOAT(KOUNT)
      WRITE(6,800)I,KNT,KOUNT,X
      IF (KOUNT.GE.9900) MFAC = 1000
      GO TO 200
C
C     TABLE LOCATION IS FILLED,
C     CHECK TO SEE IF VECTOR IS IN TABLE
  110 IF (N(L).NE.IM) GO TO 130
C
C     VECTOR IS IN TABLE, INCREMENT POPULATION COUNTER
      NPOP(L)=NPOP(L)+1
      IF (LL.EQ.L) NSS = NSS + 1
      GO TO 200
C
C     VECTOR IS NOT THE SAME AS THE ONE WITH INDEX L
C     TRY THE NEXT INDEX
      NMS = NMS + 1
C
C     CHECK TO SEE IF INDEX IS LARGER THAN END OF TABLE
C     IF SO, SET INDEX TO ONE AND START AT BEGINNING OF TABLE
      IF (L.GT.IFEAT) L = 1
      GO TO 100
C
C     RETURN TO NEXT VECTOR
  200 CONTINUE
C
C     RETURN TO NEXT RECORD
  300 CONTINUE
C
C     HAVE COMPLETED HISTOGRAM
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
C     PRINT LAST NEW VECTOR AND PEL NUMBER THAT OCCURRED
  400 I=I-1
      KNT = NPIXA(I-1) + J-1
      CNT = KNT
      COUNT=KOUNT
      X = CNT/COUNT
      WRITE (6,800) I, KNT, KOUNT, X
      WRITE (6,805) NLU, NSS, NPS
C
C     WRITE OUT FEATURE VECTORS THAT OCCUR AT LEAST 1000 TIMES
      NVEC = 0
      DO 450 I=1,IFEAT
      IF (NPOP(I).LT.1000)GO TO 450
      LM = N(I)
      NVEC = NVEC + 1
      IF (NVEC.EQ.1) WRITE (6,810)
      IF (MOD(NVEC,2).EQ.1) WRITE (6,811) LM, NPOP(I)
      IF (MOD(NVEC,2).EQ.0) WRITE (6,812) LM, NPOP(I)
  450 CONTINUE
C
C     POPULATION DISTRIBUTION IN LOGARITHMIC INCREMENTS
      DO 500 I=1,200
  500 NN(I)=0
C
      DO 550 I=1,IFEAT
      IF (NPOP(I).EQ.0) GO TO 550
C
C     COUNT THE NUMBER OF VECTORS THAT OCCUR 1000'S OF TIMES
      II=NPOP(I)/1000
      IF(II.LT.1)GO TO 510
      II=II+108
      GO TO 540
C
C     COUNT THE NUMBER OF VECTORS THAT OCCUR 100'S OF TIMES
  510 II=NPOP(I)/100
      IF (II.LT.1)GO TO 530
      II=II+99
      GO TO 540
C
C     COUNT THE NUMBER OF VECTORS THAT OCCUR FROM 1 TO 99 TIMES
  530 II=NPOP(I)
  540 NN(II)=NN(II)+1
  550 CONTINUE
C
C     PRINT THE NUMBER OF VECTORS THAT OCCUR 1-99 TIMES, 100'S AND
C     1000'S OF TIMES
      WRITE (6,815)
      J = 0
      INC = 1
      DO 560 I=1,200
      IF (I.EQ.101) INC = 100
      IF (I.EQ.110) INC = 1000
      J = J + INC
      IF (NN(I).EQ.0) GO TO 560
      X=NN(I)*100/COUNT
      WRITE (6,820) J, NN(I), X
  560 CONTINUE
C
C     COMPUTE THE MEAN, VARIANCE AND ENTROPY FROM THE HISTOGRAM
      ALN2 = ALOG10(2.0)
      DO 610 NB=1,NBAND
      NXM = 0
      NXV = 0
      SUM = 0.0
      DO 585 I=1,128
  585 IH(I) = 0
      DO 600 I=1,IFEAT
      IF (NPOP(I).EQ.0) GO TO 600
      IM = N(I)
      NXX = LM(NB)
      NXM = NXM + NPOP(I)*NXX
      NXV = NXV + NPOP(I)*NXX**2
      IH(NXX+1) = IH(NXX+1) + NPOP(I)
  600 CONTINUE
      XMEAN = NXM / CNT
      SIGMA = (NXV-CNT*XMEAN**2) / (CNT-1.0)
      DO 605 I=1,128
      IF (IH(I).EQ.0) GO TO 605
      PROBX = IH(I) / CNT
      SUM = SUM + PROBX*ALOG10(PROBX)/ALN2
  605 CONTINUE
      ENTRPY = -SUM
  610 WRITE (6,830) NB, XMEAN, SIGMA, ENTRPY
      RETURN
C
  710 FORMAT (//15X,'SCAN NO.',10X,'PIXEL NO.',10X,'VECTOR NO.',10X,
     ,'P/V RATIO'/)
  750 FORMAT ('1 PIXELS USED =',I5,5X,'RECORDS USED =',I5,5X,'MULTIPLIER
     , =',I3,5X,'DIVISOR =',I3,5X,'BASE =',I3)
  800 FORMAT (3I20,F20.4)
  805 FORMAT (/'  LOOK UPS =',I7,'   SINGLE SEARCHES =',I8,'   MULTIPLE
     ,SEARCHES =',I8///)
  810 FORMAT ('1',20X,'VECTORS WITH POPULATIONS OF AT LEAST 1000'//)
  811 FORMAT (10X,4I4,I10)
  812 FORMAT ('+',50X,4I4,I10)
  815 FORMAT ('1',10X,'NO. OF TIMES'/8X,'A VECTOR OCCURRED',10X,'NO. OF
     ,VECTORS',10X,'PERCENT OF TOTAL'/)
  820 FORMAT (I20,I25,F25.4)
  830 FORMAT (/10X,'BAND',I3,5X,'MEAN =',F8.3,5X,'VARIANCE =',F8.3,5X,
     ,'ENTROPY =',F8.3)
      END
```

COMPARISON OF SUPERVISED CLASSIFICATION MAPS

I. NAME

COMPMP

II. DESCRIPTION

To compare two supervised classification maps (or a Ground Truth Map and Supervised Classification Map) and print their joint histogram and the numbers and percentages of various types of differences.

III. CALLING SEQUENCE

CALL COMPMP (IX, IY, LY, CLASS, NREC, NEL, M, N) where IX, IY, LY are arrays dimensioned:

$$\left.\begin{array}{l}\text{IX (NEL,3)}\\ \text{IY (NEL)}\\ \text{LY (NEL)}\end{array}\right\}\ \text{bytes}$$

CLASS (maximum of M and N) double words

NREC = number of records in the two maps

NEL = number of pixels per record

M, N are the numbers of classes in maps 1 and 2

IV. INPUT/OUTPUT

1. INPUT

The input maps 1 and 2 to this program should be on units 8 and 11 respectively. They should have NREC records, NEL pixels per record and one byte per pixel in unformatted FORTRAN readable form. A title of up to 72 characters is input by card.

2. OUTPUT

Besides the printout, this program writes difference map on unit 12, with NREC unformatted FORTRAN records of NEL pixels each having one byte per pixel.

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

V. DESCRIPTION OF SUBROUTINES

The only subroutine called is READAR which reads a specified number of bytes into an array. This is to avoid implied DO loops in read statements which are excessively time consuming.

VI. PERFORMANCE SPECIFICATION

1. STORAGE

The routine requires 3790 bytes of storage. The work arrays dimensioned in the calling program require 5 x NEL bytes of storage.

2. EXECUTION TIME

The processing speed is 16,700 pixels per second, averaged over several runs.

VII. METHOD

This program first sets an M by N matrix JNTH to zero, and then finds the joint histogram between the two maps.

The next step is to separate the types of differences between the two maps and indicate them by different symbols. The numbers 0, 1, 2, and 3 are used to indicate exterior points, no difference between the maps, boundary points where the maps are different and interior points where the maps are different, respectively. The "exterior points" are defined as those where the "class labels" in either of the maps are equal to zero. The "boundary points" are those whose class labels are different from that of at least one of their four nearest neighbors (top, left, bottom, and right) in map 1. Points which are neither exterior nor boundary points are called "interior points".

These indicators are generated for each of the points in the maps and written on an NREC by NEL pixel sequential data set (unit 12). The numbers and percentages of occurrences of these indicators in the output data set are counted and printed (except for the exterior points). The percentages of occurrences are evaluated based on all but the exterior points.

VIII. COMMENTS

The data set on unit 12 can be used directly to generate a difference map.

IX. TESTS

This program has been used in deriving the difference maps and similarity measures between several pairs of classification maps and found to work satisfactorily. An example of the printed output follows. The joint histogram is augmented to include the inventory counts and percentages, the total number of pixels, the inventory accuracy, the classification accuracy, and the number of correctly classified pixels (with respect to Map 1).

X. LISTINGS

The listing of the program is attached at the end of this section.

```
MOBILE BAY GTM VS. LINEAR CLASSIFICATION        2D MAD  1 BIT
IMAGE SIZE= 1200 BY 1200. NUMBERS OF CLASSES IN MAPS 1 AND 2 ARE  6 AND  6.
TOTAL NO. OF VALID (NON-EXTERIOR) POINTS = 743328

                      *******************
                      * JOINT HISTOGRAM *
                      *******************

             URBAN  AGRICULT   FOREST    WATER  WETLAND   VACANT  INVNTORY  PERCENT
    URBAN    25726    20034    20474     1580     4005     1474    73293     9.86
 AGRICULT    27030    66038    16511       12     2458       48   112067    15.08
   FOREST    18356    33019   214516      162    17899      107   284059    38.21
    WATER      621      106      303   213477     3333      728   218568    29.40
  WETLAND     3723     1914    17524     2042    23919      151    49273     6.63
   VACANT     1591     1399     1315      319      420     1024     6068     0.82
 INVNTORY    77017   122510   270643   217592    52034     3532   743328    73.28
  PERCENT    10.36    16.48    36.41    29.27     7.20     0.48    97.72   544700
 ACCURACY    35.10    58.93    75.52    97.67    48.54    16.88

      TYPE      NO. OF OCCURRENCES     PERCENTAGE

   NO ERROR          544700              73.28
   BOUNDARY           52323               7.04
   NO BOUND          146305              19.68
   EXTERIOR          696672
```

TOTAL PIXELS — CLASSIFICATION ACCURACY

INVENTORY ACCURACY — CORRECT PIXELS

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

Figure 5. Sample Output of Subroutine COMPMP

```
      SUBROUTINE COMPMP (IX, IY, LY, CLASS, NREC, NEL, M, N)
C
C     THIS ROUTINE FINDS AND PRINTS THE JOINT HISTOGRAM
C     BETWEEN TWO MAPS, EACH WITH NREC LINES AND NEL PIXELS PER LINE.
C     THE FIRST MAP SHOULD HAVE M CLASSES OR LESS, AND THE SECOND, N OR
C     LESS. INPUT MAPS 1 AND 2 ON UNITS 8 AND 11 HAVE NUMBERS 0 THRU M,
C     0 THRU N RESPECTIVELY (UNFORMATTED FORTRAN).
C
C     THE OUTPUTS OF THIS ROUTINE ARE:
C           1) PRINT OF THE JOINT HISTOGRAM, THE NUMBER AND PERCENTAGE
C           OF CORRECT CLASSIFICATIONS, ERRORS AT BOUNDARY POINTS AND
C           ERRORS AT INTERIOR POINTS;
C           2) OUTPUT MAP ON UNIT 12 SHOWING THE TYPES OF ERRORS.
C
      LOGICAL*1 IX(NEL,3), IY(NEL), LY(NEL), ODY, TITLE(72)
      DIMENSION IH(4), PCTACC(21)
      COMMON/CONTAB/JNTM(21,21),IH1(21),IH2(21),PERCNT(21),PCTIH2(21),
     *    C1(21,21),NTCT
      INTEGER POINT(3)
      DOUBLE PRECISION CLASS(1), TYPE(4), INV, PCT
      LOGICAL*1 FMT1(22), FMT2(20), DIGITS(10)
      DATA TYPE /'EXTERIOR', 'NO ERROR', 'BOUNDARY', 'NO BOUND'/
      DATA INV, PCT /'INVATCHY', ' PERCENT'/
      DATA DIGITS /'0123456789'/, FMT1 /'(1H+,29X, (10X),F10.2)'/,
     *FMT2 /'(1H+,29X, (10X),I10)'/
C
      DO 1 I=1,M
      DO 1 J=1,N
    1 JNTM(I,J) = 0
      DO 2 I=1,4
    2 IH(I) = 0
C
      CALL READAR (8, IX, NEL)
      DO 3 I=1,NEL
    3 IX(I,2) = IX(I,1)
      DO 5 I=1,3
    5 POINT(I) = I
C
      DO 11 I=1,NREC
      K1 = POINT(1)
      K2 = POINT(2)
      K3 = POINT(3)
      IF (I.LT.NREC) CALL READAR (8, IX(1,K3), NEL)
      READ (11) IY
C
      DO 22 J=1,NEL
      LY(J) = 1
      K=IX(J,K2)
      L=IY(J)
C
C     CHECK IF EXTERIOR PIXEL IN EITHER MAP
      IF (L.EQ.0.OR.K.EQ.0) GO TO 30
C
C     GENERATE CLASS ASSIGNMENT MATRIX
      JNTM(K,L) = JNTM(K,L) + 1
```

ORIGINAL PAGE IS
OF POOR QUAL[illegible]

```
C
C     CHECK WHETHER CLASS NUMBERS AGREE
      IF (K.EQ.L) GO TO 35
C
C     CHECK WHETHER ANY OF THE 4 NEAREST NEIGHBORS OF THE IX PIXEL
C     ARE A DIFFERENT CLASS
                BDY = IX(J,K2).NE.IX(J,K1)
                   .OR.IX(J,K2).NE.IX(J,K3)
     ..OR.J.GT.1  .AND.IX(J,K2).NE.IX(J-1,K2)
     ..OR.J.LT.NEL.AND.IX(J,K2).NE.IX(J+1,K2)
      IF (BDY) LY(J) = 2
      IF (.NOT.BDY) LY(J) = 3
      GO TO 35
C
   30 LY(J) = 0
   35 IW = LY(J) + 1
      IH(IW)=IH(IW)+1
   22 CONTINUE
      DO 25 J=1,3
   25 POINT(J) = MOD(POINT(J),3) + 1
   11 WRITE (12) LY
C
C     FIND NUMBER OF NON-EXTERIOR PIXELS AND CLASSIFICATION ACCURACY
      NTOT = NREC*NEL - IH(1)
      FAC=100./NTOT
      ACC = FAC * IH(2)
C
C     FIND CLASS OCCUPANCIES IN MAP 1 AND MAP 2
      DO 9 I=1,M
    9 IH1(I) = 0
      DO 10 I=1,M
      DO 10 J=1,N
   10 IH1(I) = IH1(I) + JNTH(I,J)
      DO 19 J=1,N
   19 IH2(J) = 0
      DO 20 J=1,N
      DO 20 I=1,M
   20 IH2(J) = IH2(J) + JNTH(I,J)
C
C     FIND SIMILARITY MEASURES BASED ON CLASS POPULATIONS ONLY.
      MINC = MINO (M, N)
      INV = 0
      DO 65 I=1,MINC
   65 INV = INV + MINO (IH1(I), IH2(I))
      SIM = FAC * INV
C
C     PRINT HISTOGRAM AND PERCENTAGE OCCUPANCIES
      FMT1(10) = DIGITS(N)
      FMT2(10) = DIGITS(N)
      READ (5,100) TITLE
      WRITE (6,200) TITLE
      WRITE (6,400) NREC, NEL, M, N
      WRITE (6,300) NTOT
      WRITE (6,500)
      WRITE (6,102) (CLASS(J), J=1,N), INV, PCT
      DO 70 J=1,N
   70 PCTIH2(J) = FAC * IH2(J)
      DO 80 I=1,M
      PERCNT(I) = FAC * IH1(I)
      PCTACC(I) = 100.0 * JNTH(I,I) / IH1(I)
      WRITE (6,101) CLASS(I), (JNTH(I,J), J=1,N), IH1(I)
   80 WRITE (6,FMT1) PERCNT(I)
      WRITE (6,101) INV, (IH2(J), J=1,N), NTOT
      WRITE (6,FMT1) ACC
      WRITE (6,103) PCT, (PCTIH2(J), J=1,N), SIM
      WRITE (6,FMT2) IH(2)
      WRITE (6,104) (PCTACC(J), J=1,N)
      WRITE (6,600)
      DO 90 I=2,4
      PRCNT=IH(I)*FAC
   90 WRITE(6,700) TYPE(I),IH(I),PRCNT
      WRITE (6,700) TYPE(1), IH(1)
      RETURN
C
  100 FORMAT (72A1)
  101 FORMAT ('0',A9,11I10)
  102 FORMAT (/11X,12A10)
  103 FORMAT ('0',A9,11F10.2)
  104 FORMAT ('0 ACCURACY',11F10.2)
  200 FORMAT ('1',5X,72A1)
  300 FORMAT ('0  TOTAL NO. OF VALID (NON-EXTERIOR) POINTS =',I7)
  400 FORMAT ('0  IMAGE SIZE='I5,' BY'I5,', NUMBERS OF CLASSES IN MAPS 1
     . AND 2 ARE'I3,' AND'I3,'.')
  500 FORMAT (//30X,19('*')/30X,'* JOINT HISTOGRAM *'/30X,19('*'))
  600 FORMAT (//10X,'TYPE',5X,'NO. OF OCCURRENCES',5X,'PERCENTAGE'/)
  700 FORMAT (/A16,I15,F20.2)
      END
      SUBROUTINE READAR (NTAPE1, W, NSAMP)
C
C   READ NSAMP BYTES INTO ARRAY W FROM LOGICAL UNIT NTAPE1
C
      LOGICAL*1 W(NSAMP)
C
      READ (NTAPE1) W
      RETURN
C
C     ***********************************************************************
C     ***********************************************************************
C
      ENTRY WRITAR (NTAPE1, W, NSAMP)
C
C   WRITE NSAMP BYTES FROM ARRAY W ONTO LOGICAL UNIT NTAPE1
C
      WRITE (NTAPE1) W
      RETURN
      END
```

COMPUTATION OF CONTINGENCY MATRICES

I. NAME

CONMAT

II. DESCRIPTION

To obtain and print "contingency matrices," showing, for all pairs of classes in two classification maps, the numbers of simultaneous occurrences of various types of transitions (no boundary, horizontal boundary, vertical boundary and boundaries in both directions).

III. CALLING SEQUENCE

CALL CONMAT (NTAP1, NTAP2, NREC, NEL, M, N, IX, IY, IH)
where
NTAP1, NTAP2 are the unit numbers for reading the map class numbers,
NREC, NEL are number of records and number of pixels per record, respectively,
M, N are the numbers of classes in the two maps,
IX, IY, IH are work arrays dimensioned IH(4, 8, M, N) words,
IX (NEL,2), IY (NEL,2) bytes.

IV. INPUT/OUTPUT

1. INPUT

The input maps should be sequential data sets on units NTAP1 and NTAP2 with NREC records and NEL pixels per record on each of them. The number of bytes per pixel should be 1 and the records should be unformatted and FORTRAN readable.

A title of up to 80 characters is card input.

V. DESCRIPTION OF SUBROUTINES

The subroutine linkage is indicated in the following table:

CALLING PROGRAM	PROGRAMS CALLED
CONMAT	SARN
	INTLOG
	CONMXP

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

VI. PERFORMANCE SPECIFICATIONS

1. STORAGE

The storage required by the routines CONMAT, INTLOG and CONMXP is 1772, 212 and 5276 bytes, respectively.

2. EXECUTION TIME

Depends largely on image size and number of classes. For the case NREC=1600, NEL=850, M=N=6, it requires approximately 8 minutes.

VII. METHOD

The definitions of contingency matrices used here have been discussed in [2] and will not be covered here. The subroutine CONMAT is used to find a four dimensional array IH (dimensioned (4, 8, M, N)) and the routine CONMXP is used to print

a. M*N matrices (size 4 by 8) showing counts of agreements and disagreements for each type of transition for each pair of classes;

b. M*N matrices (size 4 by 4) showing counts of each type of transition for each pair of classes obtained by adding the right and left halves of the corresponding matrices from a. and dividing by 3;

c. A 4 by 4 matrix showing totals of each type of transition obtained by all the matrices in b;

```
MATRICES SHOWING COUNTS OF AGREEMENTS AND DISAGREEMENTS FOR EACH TYPE OF TRANSITION
MAP SIZE= 1600 BY  850.

CLASS NUMBER IN MAP 1= 2 CLASS NUMBER IN MAP 2= 3

       0     3821     4051     5814  |    39051    13153    15110     7989

    1282       17     1028      133  |     2885      697      940      416

    1182     1057        6      136  |     2739      863      699      437

    1026       89      113       13  |      627      196      259      122

CLASS NUMBER IN MAP 1= 2 CLASS NUMBER IN MAP 2= 4

       0        0        0        3  |       15        3       15       15

       0        0        1        0  |        0        0        2        0

       0        2        0        0  |        0        4        0        0

       0        0        1        0  |       12        3        2        0

CLASS NUMBER IN MAP 1= 2 CLASS NUMBER IN MAP 2= 5

       0      224      202      460  |     5022     2527     3095     2828

       1       21       14       38  |      131       81      133      121

       0        6       21       40  |      147       96       84      119

       0        5        8       24  |       36       22       25       24

CLASS NUMBER IN MAP 1= 2 CLASS NUMBER IN MAP 2= 6

       0        6       13       34  |       15       30       44       59

       3        0        1        1  |        9        0        2        5

       0        0        0        0  |        0        3        0        0

       0        0        0        0  |        3        0        0        3
```

Figure 6. A Portion of the Printout from CONMAT

d. An M by N matrix which is the joint histogram (contingency table) of the two input maps, whose (I, J)th element is obtained by adding all the 16 elements in the 4 x 4 matrix corresponding to classes (I, J) defined in b;

e. The individual histograms (inventories) of the two maps obtained by adding the columns (for map 1) and rows (for map 2).

Also, the transition and point by point similarity counts (traces of the 4 by 4 and M by N matrices, respectively) and percentage similarity measures are printed.

VIII. COMMENTS

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

The maximum class number is 10.

IX. TESTS

Portions of the printout from a test run on the Mobile Bay ground truth vs. linear classification (December 5, 1973) are shown at the end of this section.

X. LISTINGS

The listings of CONMAT, CONMXP, and INTLOG are attached at the end of this section.

```
      SUBROUTINE CONMAT(NTAP1,NTAP2,NREC,NEL,M,N,IX,IY,IM)
C
C     THIS PROGRAM FINDS CONTINGENCY MATRICES INDICATING AGREEMENTS
C     BETWEEN TWO MAPS IN TERMS OF CLASS LABELS AND THE BOUNDARY TYPES.
C     IM(*,*,I,J) REFERS TO LOCATIONS WITH CLASS I IN MAP 1 AND CLASS J
C     IN MAP 2.  LEFT HALF OF IM GIVES A COUNT OF AGREEMENTS AND THE
C     RIGHT HALF, DISAGREEMENTS.
C     ROWS OF IM(*,*,I,J) CORRESPOND TO MAP 1, AND COLUMNS TO MAP2.
C     ROW NUMBERS 1, 2, 3, 4 INDICATE NO BOUNDARY, CHANGE IN VERTICAL
C     DIRECTION, CHANGE IN HORIZONTAL DIRECTION, CHANGE IN BOTH DIREC-
C     TIONS, RESPECTIVELY, IN MAP 1.  SIMILARLY COLUMN NUMBERS INDICATE
C     TYPES OF TRANSITIONS IN MAP2.
C     THE PROGRAM HANDLES THE PRESENT ROW OF THE MAP 1 IN IX(*,I2)
C     AND THE IMMEDIATELY PREVIOUS ROW IN IX(*,I1).  THE ROWS OF MAP 2
C     ARE HANDLED SIMILARLY IN IY.
C
      DIMENSION IM(4,8,M,N)
      LOGICAL*1 IX(NEL,2), IY(NEL,2)
C
C     INITIALIZE THE ARRAYS IX AND IY.  THE "PREVIOUS" ROW TO ROW 1 IS
C     CONSIDERED IDENTICAL TO ROW 1.
      CALL SARN (NTAP1, IX, NEL)
      CALL SARN (NTAP2, IY, NEL)
      DO 10 J=1,NEL
      IX(J,2)=IX(J,1)
   10 IY(J,2)=IY(J,1)
      DO 20 I=1,4
      DO 20 J=1,8
      DO 20 K=1,M
      DO 20 L=1,N
   20 IM(I,J,K,L) = 0
      I1=1
      I2=2
C
C     LOOP ON RECORDS.
      DO 40 I=1,NREC
C
C     LOOP ON PIXELS.
      DO 30 J=1,NEL
      JP=MAX0(1,J-1)
C
C     NONPOSITIVE VALUES OF MAP LABELS ARE NOT OF INTEREST.
      IF (IX(J,I2).EQ.0.OR.IY(J,I2).EQ.0) GO TO 30
C
C     FIND ROW AND COLUMN NUMBERS IN IM TO BE INCREMENTED.
C
C     CHECK THE NATURE OF THE BOUNDARIES IN BOTH THE MAPS.
      K=IX(J,I2)
      L=IY(J,I2)
      II=1+INTLOG(IX(J,I2).NE.IX(J,I1))+2*INTLOG(IX(J,I2).NE.IX(JP,I2))
      JJ=1+INTLOG(IY(J,I2).NE.IY(J,I1))+2*INTLOG(IY(J,I2).NE.IY(JP,I2))
C
C     FIND THE INCREMENTS.
C     INC = NUMBER OF AGREEMENTS; INC3 = NUMBER OF DISAGREEMENTS.
      INC=INTLOG(IX(J,I2).EQ.IY(J,I2))
```

```
     . + INTLOG(IX(J,I1).EQ.IY(J,I1)) + INTLOG(IX(JP,I2).EQ.IY(JP,I2))
      INC3=3*INC
      IH(II,JJ,K,L)=IH(II,JJ,K,L) + INC
      IH(II,JJ+4,K,L)=IH(II,JJ+4,K,L)+INC3
   30 CONTINUE
C
C     EXCHANGE I1 AND I2
      IW=I1
      I1=I2
      I2=IW
C
C     READ NEXT RECORDS INTO IX(*,I2),IY(*,I2)
      IF (I.EQ.NREC) GO TO 40
      CALL SARN (NTAP1, IX(1,I2), NEL)
      CALL SARN (NTAP2, IY(1,I2), NEL)
   40 CONTINUE
C
C     PRINT THE MATRICES SHOWING TRANSITION COUNTS
      CALL CONMXP (IH, NREC, NEL, M, N)
      RETURN
      END
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```
      SUBROUTINE CONMXP (IM, NREC, NEL, M, N)
C
C     PRINT MATRICES SHOWING NUMBERS OF AGREEMENTS AND DISAGREEMENTS
C     OF TRANSITIONS FOR EACH PAIR OF CLASSES.
C     INPUTS:  TITLE IS AN 80 CHARACTER(MAX) TITLE TO BE PRINTED ON TOP
C              OF EACH PAGE OF OUTPUT.  NREC, NEL, M, N ARE NUMBER OF
C              RECORDS, NUMBER OF PIXELS/RECORD, NUMBER OF CLASSES IN
C              MAP 1 AND NUMBER OF CLASSES IN MAP 2, RESPECTIVELY.
C
      DIMENSION IM(4,8,M,N), IM1(4,4), IM2(20,20), INV1(20), INV2(20)
      LOGICAL*1 TITLE(80)
C
      READ(5,100) TITLE
      L=0
      DO 10 I=1,M
      DO 10 J=1,N
      IF(MOD(L,4).NE.0)GO TO 15
      WRITE(6,110)TITLE
      WRITE(6,400)NREC,NEL
15    CONTINUE
      L=L+1
      WRITE(6,500)I,J
      DO 20 K=1,4
20    WRITE(6,300)(IM(K,KK,I,J),KK=1,8)
10    CONTINUE
C
C     FIND AND PRINT MATRICES SHOWING COUNTS OF EACH TYPE OF TRANSITION
C     FOR EACH PAIR  OF CLASSES.  THESE SHOW, FOR ALL JOINT OCCURRENCES
C     OF CLASSES (I,J) IN MAPS 1,2, THE NUMBERS OF JOINT OCCURRENCES
C     OF EACH TYPE OF TRANSITION IN THE TWO MAPS.
      DO 30 I=1,M
      DO 30 J=1,N
      DO 30 K=1,4
      DO 30 L=1,4
30    IM(K,L,I,J)=(IM(K,L,I,J)+ IM(K,L+4,I,J))/3
C
      L=0
      DO 40 I=1,M
      DO 40 J=1,N
      IF(MOD(L,4).NE.0)GO TO 45
      WRITE(6,110)TITLE
      WRITE(6,410)NREC,NEL
45    CONTINUE
      L=L+1
      WRITE(6,500)I,J
      DO 50 K=1,4
50    WRITE(6,310)(IM(K,KK,I,J),KK=1,4)
40    CONTINUE
C
C     FIND IM1, THE MATRIX OF COUNTS OF TRANSITION TYPES (WITHOUT REGARD
C     TO CLASS LABELS) AND IM2, THE MATRIX OF JOINT OCCURRENCES OF CLASS
C     LABELS (WITH NO REGARD TO TRANSITION TYPES).
      DO 55 I=1,4
      DO 55 J=1,4
55    IM1(I,J) = 0
```

```
      DO 56 I=1,M
      DO 56 J=1,N
56    IM2(I,J) = 0
      DO 60 I=1,M
      DO 60 J=1,N
      DO 60 K=1,4
      DO 60 L=1,4
      IM1(K,L)=IM1(K,L)+IM(K,L,I,J)
60    IM2(I,J)=IM2(I,J)+IM(K,L,I,J)
C
C     PRINT IM1 AND THE CORRESPONDING SIMILARITY MEASURE.
      WRITE(6,110)TITLE
      WRITE(6,420) NREC,NEL
      DO 70 K=1,4
70    WRITE(6,310)(IM1(K,L),L=1,4)
      ITR=0
      ISUM=0
      DO 90 I=1,4
      ITR=ITR+IM1(I,I)
      DO 90 J=1,4
90    ISUM=ISUM+IM1(I,J)
      PCT=ITR*100./ISUM
      WRITE(6,600)ITR,ISUM,PCT
      WRITE(6,430)
C
C     PRINT IM2 AND THE CORRESPONDING SIMILARITY MEASURE.
      DO 80 I=1,M
80    WRITE(6,310)(IM2(I,J),J=1,N)
      ITR=0
      ISUM=0
      DO 95 I=1,M
      IF(I.LE.N)ITR=ITR+IM2(I,I)
      DO 95 J=1,N
95    ISUM=IM2(I,J)+ISUM
      PCT=ITR*100./ISUM
      WRITE(6,700)ITR,ISUM,PCT
C
      DO 81 I=1,M
81    INV1(I) = 0
      DO 82 J=1,N
82    INV2(J) = 0
      DO 85 I=1,M
      DO 85 J=1,N
      INV1(I)=INV1(I)+IM2(I,J)
85    INV2(J)=INV2(J)+IM2(I,J)
      WRITE(6,800)(INV1(I),I=1,M)
      WRITE(6,810)(INV2(J),J=1,N)
      RETURN
C
100   FORMAT(80A1)
110   FORMAT('1'20X80A1)
300   FORMAT('0'4I8,'  I '4I8)
310   FORMAT('0'15I8)
400   FORMAT(/' MATRICES SHOWING COUNTS OF AGREEMENTS AND DISAGREEMENTS
     ,FOR EACH TYPE OF TRANSITION'/' MAP SIZE='I5,' BY'I5,'.')
```

```
410   FORMAT(/' MATRICES SHOWING COUNTS OF EACH TYPE OF TRANSITION'/
     .      /' MAP SIZE='I5,' BY'I5,'.')
420   FORMAT(/' MATRIX SHOWING TOTALS OF EACH TYPE OF TRANSITION'/
     .      ' MAP SIZE='I5,' BY'I5,'.')
430   FORMAT(//' CONTINGENCY TABLE')
500   FORMAT(//' CLASS NUMBER IN MAP 1='I2,' CLASS NUMBER IN MAP 2='I2)
600   FORMAT(' TRANSITION SIMILARITIES='I7,'; TOTAL='I7,'; PERCENTAGE='
     .      F7.2)
700   FORMAT(' NUMBER OF POINT BY POINT SIMILARITIES='I7,' TOTAL='I7,
     .      ' PERCENTAGE='F7.2)
800   FORMAT(/' INVENTORY OF MAP1:'/(1X15I8))
810   FORMAT(/' INVENTORY OF MAP2:'/(1X15I8))
      END
      FUNCTION INTLOG(L)
C
C     CONVERT A LOGICAL VARIABLE TO INTEGER
C       IF  TRUE, FUNCTION RETURNS 1
C       IF FALSE, FUNCTION RETURNS 0
C
      LOGICAL L
C
      INTLOG=0
      IF(L)INTLOG=1
      RETURN
      END
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

CHAPTER III

REGISTRATION OF IMAGE DATA

MAGNIFICATION OF IMAGERY

I. NAME

CUMAG (Cubic Magnification)

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

II. DESCRIPTION

This subroutine magnifies a specified segment of imagery data. The magnification is by means of cubic interpolation, which is used to compute the densities of the additional samples in the magnified image. The routine also removes the distortions present in Landsat MSS imagery which are due to Earth rotation and "sensor delay" in the A/D conversion of sensor data. The purpose of this routine is to allow determination, to within a fraction of a pixel spacing, of the coordinates of ground control points.

III. CALLING SEQUENCE

CALL CUMAG (X, OUT1, OUT2, NB, NEL, NSOUT, MAG, IGCPL, IGCPS)

where

X is the input buffer array,

OUT1 is the intermediate output holding four lines of interpolated data,

OUT2 is the output array,

NB is the number of channels of data,

NEL is the number of input pixels,

NSOUT is the number of output pixels,

MAG is the magnification,

IGCPL is the ground control point line coordinate, and

IGCPS is the ground control point sample coordinate.

IV. INPUT/OUTPUT

The input data and the magnified output data are in bytes, arranged by vectors containing the data for each channel. The input should be a direct access file and the input and output logical units are 10 and 11.

V. DESCRIPTION OF SUBROUTINES

The subroutines required are given in the following table.

SUBROUTINES FOR CUBIC MAGNIFICATION

NAME	STORAGE (Bytes)	FUNCTION
CUMAG	2306	Determine input data coordinates allowing for each rotation and sensor delays, magnify the segment by cubic interpolation.
DELAY	496	Function which computes shift due to earth rotation and sensor delay in units of pixels.

VI. PERFORMANCE SPECIFICATIONS

1. STORAGE REQUIREMENTS

The program requires a total of 50K bytes when magnifying portions of a LACIE sample segment by 8 times to a size of 241 x 241.

2. EXECUTION TIME

The time required for the 8X magnification referred to in the preceding paragraph is approximately 15 seconds.

VII. METHOD

The cubic interpolation formula

$I = d^3 (-I1+I2-I3+I4) + d^2 (2I1-2I2+I3-I4) + d (-I1+I3) + I2,$

where I is density and d is distance from the second pixel, is used to insert MAG pixels between the second and third pixels in groups of four pixels. Thus, given a string of input data containing IN input pixels and (IN-1) interpixel spaces, there are (IN-3) spaces in which to add MAG pixels. The number of output samples obtained is

$$NSOUT = MAG\ (IN-3) +1.$$

This is illustrated by the following diagram.

	CENTER
ORIGINAL	0 · 0 0 0 0
INTERPOLATED PIXELS	$\underbrace{X\ X\ X\ X}\ \underbrace{X\ X\ X\ X}\ X$
NUMBER OF PIXELS	MAG MAG 1

It is also apparent from the diagram that the magnified image is centered on an input pixel if the number of output pixels is chosen to be

$$2n\ MAG + 1,$$

where n is a positive integer. Also, since interpolation starts at the second pixel in groups of four, the first output pixel is the second input pixel, and the spacing of the output pixels is 1/MAG. Thus, if the magnification is 10, the input coordinates are obtained to 0.1 pixel. The input coordinate values are given by

$$IN = NS1 + 1+(OUT-1)/MAG,$$

where NS1 is the beginning sample used for interpolation, and OUT is the output sample number.

In the case of Landsat 1 and Landsat 2 imagery, NS1 will be the beginning sample in the distortion-free offset image, which is the image used for determining ground control points and geometric transformation functions. The beginning pixel in the offset system is a constant for the magnified image, while the input pixel numbers in the data vary from line to line due to the delays.

The distortions in the imagery, due to earth rotation during the scanner retrace and delay in sampling by the A/D converter, are referred to as earth rotation delay and sensor delay and are computed by the function DELAY. As the Earth rotates from the west, successive mirror scans cover more westerly areas of the Earth, and, consequently, should be offset to smaller pixel values. The scanner sweeps an additional distance to the east during sensor delay and, therefore, creates an opposite effect. However, rotational delay per swath is larger up to a latitude of 48.8°.

The beginning pixel in the offset image is determined by subtracting the delay for the central line of the magnified region. (It then follows that when the offset pixel coordinates are converted to input coordinates, original pixels occur at every MAG output pixel on the central line.) However, in general, due to the varying amounts of delay from line to line, the beginning input pixel occurs at a fractional value, which determines the initial value of d in the interpolation formula. The interpolation then proceeds along a line, incrementing the input pixel number and adjusting d to be the distance from the second pixel in the interpolation group.

VIII. COMMENTS

The program does not check for input pixels falling outside the data limits.

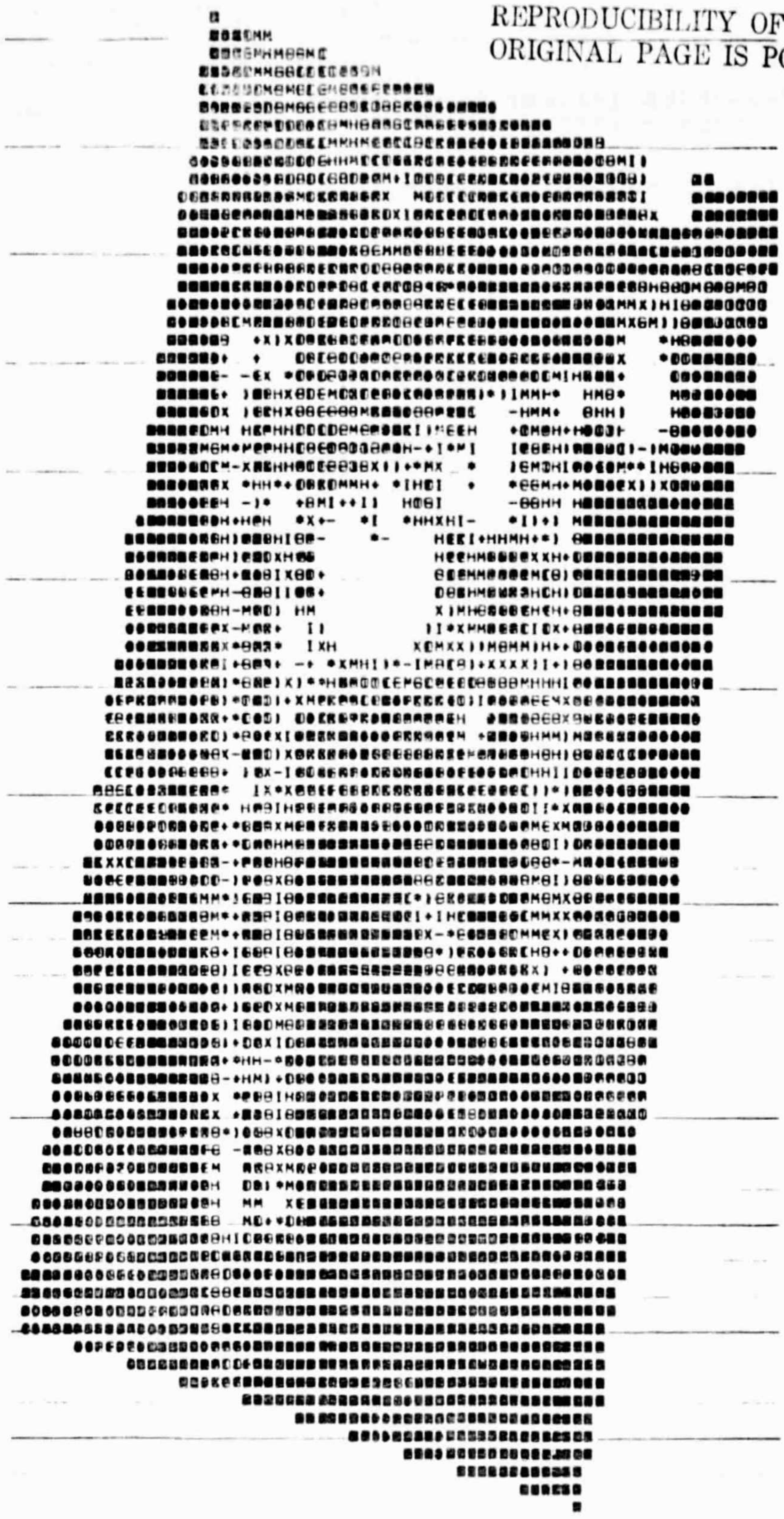
IX. TESTS

The output imagery has been examined carefully using printer plots and has been found to have the distortions properly removed. (See Fig. 7.)

X. LISTING

A listing of the routine follows.

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR



ORIGINAL PAGE IS OF POOR QUALITY

Figure 7. Huntsville Airport Registered to UTM Coordinates with Earth Rotation and Sensor Delay Effects Removed

```
      SUBROUTINE CUMAG (X,OUT1,OUT2,NB,NEL,NSOUT,MAG,IGCPL,IGCPS)
C
C     THIS ROUTINE REMOVES EARTH ROTATION AND SENSOR DELAY DISTORTIONS
C     AND MAGNIFIES IMAGERY SURROUNDING A GROUND CONTROL POINT USING
C     CUBIC INTERPOLATION.
C
      LOGICAL*1 X(NB,NEL),OUT2(NB,NSOUT)
      DIMENSION OUT1(NB,NSOUT,4)
      INTEGER POINT(4)
C
      AMAG = MAG
      DO 10 I=1,4
   10 POINT(I)=I
C
C     COMPUTE INPUT PIXELS REQUIRED.  NOTE:  NSOUT SHOULD BE 1 + AN EVEN
C     INTEGER MULTIPLE OF MAG TO BE CENTERED ON AN INPUT PIXEL
      INPIX = (NSOUT+3*MAG-1) / MAG
C
C     COMPUTE BEGINNING LINE AND BEGINNING ELEMENT IN OFFSET COORDINATES
      NL1 = IGCPL - (INPIX-1)/2
      NS1 = IGCPS - (INPIX-1)/2
      OFFNS1 = NS1 - DELAY(IGCPL)
C
C     SELECT REGIONS CENTERED ABOUT GROUND CONTROL COORDINATES
      NL = NL1 - 1
      NREC = 0
   12 CONTINUE
      NL = NL + 1
      READ (10'NL) X
      NROW = POINT(1)
      DO 20 I=1,4
   20 POINT(I) = MOD(POINT(I),4) + 1
C
C     COMPUTE DELAY FOR THIS LINE AND BEGINNING ELEMENT IN CCT COORDS.
      CCTNS1 = OFFNS1 + DELAY(NL)
C
C     FIND BEGINNING CCT PIXEL NUMBER AND DISTANCE TO FRACTIONAL PIXEL
C     AT STARTING PIXEL AND AT ALL SUCCEEDING PIXELS
      DO 50 IB=1,NB
      NS = CCTNS1
      D1 = CCTNS1 - NS
      D2 = AMOD (D1, 1.0/AMAG)
      NPIX = 0
C
   30 CONTINUE
      I1 = X(IB,NS)
      I2 = X(IB,NS+1)
      I3 = X(IB,NS+2)
      I4 = X(IB,NS+3)
      A0 = I2
      A1 = I3 - I1
      A2 = I3 - I4 + 2*(I1-I2)
      A3 = I4 - I3 + I2 - I1
C
C   CUBIC INTERPOLATE A LINE, TO ENLARGE "MAG" TIMES
```

```
      INTS = 0
   40 CONTINUE
      D = D1 + INTS/AMAG
      IF (D.GT.1.0) GO TO 35
C
      NPIX = NPIX + 1
      OUT1(IB,NPIX,NROW) = D* (D* (D*A3 + A2) + A1) + A0
      IF (NPIX.EQ.NSOUT) GO TO 50
      INTS = INTS + 1
      GO TO 40
C
C     ADJUST NS AND D1 FOR NEXT CCT PIXEL
   35 NS = NS + 1
      D1 = D2
      GO TO 30
   50 CONTINUE
C
C  ADD "MAG" INTERPOLATED LINES, AFTER OBTAINING 4 INTERPOLATED LINES
C  IN ARRAY 'OUT1'.
      IF (NL-NL1.LT.3) GO TO 12
      DO 70 INTL=1,MAG
      D = (INTL-1)/AMAG
C
      DO 60 IB=1,NB
      DO 60 NP=1,NSOUT
      R1=OUT1(IB,NP,POINT(1))
      R2=OUT1(IB,NP,POINT(2))
      R3=OUT1(IB,NP,POINT(3))
      R4=OUT1(IB,NP,POINT(4))
      CUPIX = D* (D* (D* (R4-R3+R2-R1) + (R3-R4-2.0*R2 + 2.0*R1)) +
     .(R3-R1)) + R2
      CUPIX = AMAX1 (CUPIX, 0.0)
      CUPIX = AMIN1 (CUPIX, 255.0)
      OUT2(IB,NP)=CUPIX+0.5
   60 CONTINUE
C
      WRITE (11) OUT2
      NREC = NREC + 1
      IF (NREC.EQ.NSOUT) RETURN
   70 CONTINUE
      GO TO 12
      END
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
      FUNCTION DELAY (LINE)
C
C     COMPUTE ROTATIONAL AND SENSOR DELAY RELATIVE TO FIRST SWATH
C
C     DEGCEN = LATITUDE AT THE CENTER OF THE LANDSAT SCENE
C     PIXDLY = EQUATORIAL EARTH ROTATION PER SWATH IN PIXELS
C     DEGPLN = CHANGE IN LATITUDE PER SCAN LINE
C     RADDEG = RADIANS PER DEGREE
C     SDELAY = SENSOR SAMPLING INTERVAL BETWEEN LINES (2) / TOTAL NUMBER
C              OF DETECTORS (25)
C     LINESW = LINE NUMBER IN THE SWATH (1 - 6)
C
      DIMENSION SDELAY(6)
      INTEGER CCTLIN, SWATH
      LOGICAL CCT
      COMMON /LANDST/ CCT, LLC, DEGCEN, SAMPOF, LINOFF, AMPL, PHASE
      DATA PIXDLY, DEGPLN, RADDEG, SDELAY /0.6, 0.00071086, 0.01745329,
     .0.0, 0.08, 0.16, 0.24, 0.32, 0.40/
C
      IF (CCT) GO TO 10
      DELAY=0.0
      RETURN
C
   10 CONTINUE
      CCTLIN = LINE + LINOFF
      DEGLAT = DEGCEN + (1170.5-CCTLIN) * DEGPLN
      RDELAY = PIXDLY * COS (DEGLAT*RADDEG)
      SWATH = (CCTLIN-1)/6 + 1
      LINESW = MOD(CCTLIN-1,6) + 1
      DELAY = RDELAY*(SWATH-1) - SDELAY(LINESW)
      RETURN
      END
```

FINDING COEFFICIENTS OF REGISTRATION MAPPING FUNCTIONS

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

I. NAME

GCPFIT (Ground Control Points Fit)

II. DESCRIPTION

The coordinates of a set of ground control points are determined in the two scenes or maps to be registered. If the first scene is Landsat imagery which is being registered to a map, the nonlinear distortions due to Earth rotation and mirror velocity are removed from the coordinates. A least squares fit to the transformation polynomial coefficients is made and the predicted and observed results are compared.

III. CALLING SEQUENCE

CALL GCPFIT (GCP, MGCP, MAXDEG)

where

GCP is an array of dimensions 4 by MGCP containing the ground control point coordinates,

MGCP is the number of control points, and

MAXDEG is the highest degree polynomial obtained (up to degree 5 only).

IV. INPUT/OUTPUT

1. INPUT

In addition to the arguments, the following Landsat parameters are input via COMMON block LANDST:

CCT	-logical variable; TRUE if Landsat corrections are to be applied.
LLC	-line length code (length of raw scan line obtained from Landsat computer compatible tape).
DEGCEN	-degrees in latitude at the center of the Landsat scene.
SAMPOF LINOFF	-samples and lines by which the imagery is offset from the beginning of the Landsat scene.
AMPL PHASE	-amplitude and phase (in pixels) of mirror velocity profile.

2. OUTPUT

Printed output is tables of Landsat corrections, coefficients, and error analysis. Coefficients are output via COMMON block LSQCFC.

V. DESCRIPTION OF SUBROUTINES

The subroutines called are listed in the following table.

EXTERNAL LINKAGES

CALLING PROGRAM	PROGRAMS CALLED
GCPFIT	LSQCF EVPOLY GCPCOR
LSQCF	DLLSQ
GCPCOR	DELAY MVPOFF ERCURV

Descriptions of the subroutines are given in the following table.

DESCRIPTION OF SUBROUTINES

SUBROUTINE OR ENTRY	STORAGE (BYTES)	FUNCTION
GCPFIT	2932	Bias data to zero means, call least squares fit routine, perform accuracy analysis of fit.
LSQCF	21748	Setup data point arrays and call least squares fit routine.
EVPOLY	912	Evaluate a polynomial.
GCPCOR	838	Correct the GCP's for mirror velocity and Earth curvature effects.
DLLSQ	3198	Double precision solution of a system of simultaneous linear equations.
DELAY	540	Function which computes rotational and sensor delays.

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

DESCRIPTION OF SUBROUTINES

SUBROUTINE OR ENTRY	STORAGE (BYTES)	FUNCTION
MVPOFF	374	Function which computes mirror velocity correction.
ERCURV	578	Function which computes Earth curvature correction.

VI. PERFORMANCE SPECIFICATION

1. STORAGE

The storage required for the subroutines listed is 31120 bytes.

2. EXECUTION TIME

The execution time required to obtain one fit is approximately one second.

VII. METHOD

The first step is the removal of nonlinear distortions in the Landsat coordinates. Earth rotation and sensor delay had been previously removed because they produce visible distortions in the imagery. They are added back to get the original CCT pixel number and hence the mirror velocity and Earth curvature corrections. Thus, the least squares fitting to the transformation polynomials is performed in a coordinate system from which rotational delay, sensor delay, mirror velocity, and Earth curvature distortions have been removed.

Next, the mean values for each set of coordinates (pixel, line, easting, northing) are subtracted. This is to prevent computation errors due to the large values of easting and northing.

The problem is to minimize the Euclidean norm

$$||A\ x - B||$$

where x is the solution matrix of polynomial coefficients, A is the input coordinates raised to the appropriate powers, and B is the transformed coordinates. For a second degree polynomial, A is of the form:

$$\begin{pmatrix} 1 & GCP(3,1) & GCP(4,1) & GCP(3,1)^2 & GCP(3,1)GCP(4,1) & GCP(4,1)^2 \\ 1 & GCP(3,2) & GCP(4,2) & GCP(3,2)^2 & GCP(3,2)GCP(4,2) & GCP(4,2)^2 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

and B is of the form:

$$\begin{pmatrix} GCP(1,1) & GCP(2,1) \\ GCP(1,2) & GCP(2,2) \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{pmatrix}$$

The solution is obtained by the subroutine DLLSQ, which is in the IBM Scientific Subroutine Package.

VIII. COMMENTS

The routines are dimensioned to allow a maximum polynomial degree of five.

IX. TESTS

The results are identical to those obtained by a solution of the least squares normal equations employing partial derivatives with respect to the fit coefficients.

X. LISTINGS

Listings of the routines except DLLSQ follow. (DLLSQ is a routine in the IBM Scientific Subroutine Package.)

```
      SUBROUTINE GCPFIT (GCP, MGCP, MAXDEG)
C
C     TO FIND GEOMETRIC TRANSFORMATION NEEDED FOR GEOGRAPHIC REFERENCING
C
      DIMENSION GCP(4,MGCP)
      DOUBLE PRECISION COEF(21,4)
      LOGICAL CCT
      INTEGER NTERM(5)/3,6,10,15,21/
      COMMON /LANDST/ CCT, LLC, DEGCEN, SAMPOF, LINOFF, AMPL, PHASE
      COMMON /LSQCFC/ COEF, LSQDEG, IER1(2), TOL
      COMMON / MEANS/ GCPM(4)
C
C     GCP IS GROUND CONTROL POINT TABLE INPUT BY USER
C     1. PIXEL WITHIN LINE
C     2. LINE WITHIN FRAME
C     3. EASTING
C     4. NORTHING
C     COEF(21,I)= COEFFICIENTS FOR I 1= PIXEL
C                                    2= LINE
C                                    3= EASTING
C                                    4= NORTHING
C
      IF (CCT) CALL GCPCOR (GCP, MGCP)
      AMGCP=MGCP
      RADDEG = 180.0/3.14159265
      TOL=1.E-20
      DO 1 I=1,4
    1 GCPM(I) = 0.0
      WRITE (6,101)
      WRITE (6,1020)
C
C     COMPUTE SUMS OF INPUT DATA
      DO 10 J=1,MGCP
      DO 5 I=1,4
    5 GCPM(I) = GCPM(I) + GCP(I,J)
      WRITE (6,102) J, (GCP(I,J), I=1,4)
   10 CONTINUE
C
C     COMPUTE MEANS OF INPUT DATA
      DO 11 I=1,4
   11 GCPM(I) = GCPM(I) / AMGCP
      WRITE (6,104) GCPM
C
C     SUBTRACT MEANS OF INPUT DATA
      WRITE (6,1019)
      WRITE (6,1020)
      DO 13 J=1,MGCP
      DO 12 I=1,4
   12 GCP(I,J) = GCP(I,J) - GCPM(I)
      WRITE (6,102) J, (GCP(I,J), I=1,4)
   13 CONTINUE
C
C     FIND POLYNOMIAL FITS FOR DEGREES 1 - INPUT VALUE OF LSQDEG
      DO 100 LSQDEG=1,MAXDEG
      ISTOP=NTERM(LSQDEG)
      IF (ISTOP.GT.MGCP) GO TO 400
      CALL LSQCF (0, GCP, MGCP)
      WRITE(6,383) LSQDEG,IER1,((COEF(J,I),I=1,4),J=1,ISTOP)
      WRITE(6,111)
      RESPX=0.0
      RESUT=0.0
C
C     PERFORM ACCURACY ANALYSIS OF FIT
      DO 388 I=1,MGCP
      GP=GCP(1,I) + GCPM(1)
      GL=GCP(2,I) + GCPM(2)
      GE=GCP(3,I) + GCPM(3)
      GN=GCP(4,I) + GCPM(4)
C
      CALL EVPOLY(1,GE,GN,ANS)
      DP=GP-ANS
      CALL EVPOLY(2,GE,GN,ANS)
      DL=GL-ANS
      SQ = DP**2 + DL**2
      RESPX = RESPX + SQ
      XMAG = SQRT (SQ)
      XDIR = RADDEG * ATAN2(-DL,DP)
C
      CALL EVPOLY(3,GP,GL,ANS)
      DE=GE-ANS
      CALL EVPOLY(4,GP,GL,ANS)
      DN=GN-ANS
      SQ = DE**2 + DN**2
      RESUT = RESUT + SQ
      UMAG = SQRT (SQ)
      UDIR = RADDEG * ATAN2(DE,DN)
      WRITE (6,109) I, XMAG, XDIR, DP, DL, UMAG, UDIR, DE, DN
388   CONTINUE
C
      AGCP1=MGCP-1
      RESPX = SQRT (RESPX/AGCP1)
      RESUT = SQRT (RESUT/AGCP1)
      WRITE (6,390) RESPX, RESUT
  100 CONTINUE
      LSQDEG = MAXDEG
      RETURN
C
  400 WRITE (6,1100) LSQDEG, ISTOP, MGCP
      LSQDEG = LSQDEG - 1
      RETURN
C
C     FORMAT STATEMENTS.
  101 FORMAT ('1',20X,'GROUND CONTROL POINTS'/)
  102 FORMAT (1X,I4,4F15.3)
  104 FORMAT (///20X,'MEANS OF INPUT DATA ARE'//5X,4F15.3/
     *       'OTHESE MEANS ARE FIRST SUBTRACTED')
  109 FORMAT (1X,I3,2(F10.3,F10.1,8X,2F10.3,8X))
  111 FORMAT (/35X,'COMPARISON OF OBSERVED AND PREDICTED VALUES'/
     ,25X,'LANDSAT',50X,'GEOGRAPHIC'/15X,'ERROR',20X,'OBS - PRED',20X,
     ,'ERROR',20X,'OBS - PRED'/5X,'MAGNITUDE  DIRECTION',10X,'P ERROR
     .L ERROR',10X,'MAGNITUDE  DIRECTION',10X,'E ERROR   N ERROR'/)
 383 FORMAT ('1  LEAST SQUARES FIT OF DEGREE',I3,25X,'ERROR CODES =',
     .2I3/38X,'COEFFICIENTS'/11X,'PIXEL',15X,'LINE',15X,'EASTING',12X,
     .'NORTHING'/(1X,1P4E20.6))
 390 FORMAT (/2(15X,'RMS ERROR =',E13.6,15X))
1019 FORMAT ('1',7X'CONTROL POINTS AFTER SUBTRACTING MEANS'/)
1020 FORMAT (20X,'LANDSAT',20X,'GEOGRAPHIC'/18X,'COORDINATES',18X,'COOR
     .DINATES'/' GCP NO.',7X,'PIXEL',10X,'LINE',9X,'EASTING',7X,'NORTHIN
     .G'/)
1100 FORMAT (/20X,'FIT OF DEGREE',I2,' REQUIRES',I3,' GROUND CONTROL PO
     .INTS.',I5,' WERE SUPPLIED.'/)
     END
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
      SUBROUTINE LSQCF (IDEL, GCP, MGCP)
C
C     THIS SUBROUTINE CALCULATES THE LEAST-SQUARES COEFFICIENTS
C     FOR THE BIVARIATE POLYNOMIAL
C
      DIMENSION GCP(4,MGCP)
      DOUBLE PRECISION COEF(21,4), CORE(2500), X(45), B(200), AUX(100),
     .A(2100), S(45), XXX, YYY, AMAX, C
      INTEGER NTERM(5)/3,6,10,15,21/
      INTEGER XP(21)/0,1,0,2,1,0,3,2,1,0,4,3,2,1,0,5,4,3,2,1,0/
      INTEGER YP(21)/0,0,1,0,1,2,0,1,2,3,0,1,2,3,4,0,1,2,3,4,5/
      INTEGER IPIV(50),IND1(2)/1,3/,IND2(2)/3,1/
      EQUIVALENCE (CORE(1),X(1)), (CORE(46),B(1)), (CORE(246),AUX(1)),
     .(CORE(346),A(1)), (CORE(2446),S(1))
      COMMON /LSQCFC/ COEF, LSQDEG, IER1(2), TOL
C
C     GCP IS GROUND CONTROL POINT TABLE INPUT BY USER
C     1.  PIXEL WITHIN LINE
C     2.  LINE WITHIN FRAME
C     3.  EASTING
C     4.  NORTHING
C     COEF(21,I)= COEFFICIENTS FOR I 1= PIXEL
C                                    2= LINE
C                                    3= EASTING
C                                    4= NORTHING
      L=MGCP
      L1=MGCP
      IF(IDEL.NE.0) L=MGCP-1
      ISTOP=NTERM(LSQDEG)
      IF (ISTOP.GT.L) RETURN
C
      DO 10 II=1,2
      III=IND1(II)
      [illegible]=(II-1)*2+1
      IV=IND2(II)
      NGCP1=0
C
      DO 50 NGCP=1,L1
      IF(NGCP.EQ.IDEL) GO TO 50
      NGCP1=NGCP1+1
      B(NGCP1)=GCP(III,NGCP)
      NN=NGCP1+L
      B(NN)=GCP(III+1,NGCP)
      DO 50 I=1,ISTOP
      XXX=1.0D0
      IF(XP(I).NE.0) XXX=GCP(IV,NGCP)**XP(I)
      YYY=1.0D0
      IF(YP(I).NE.0) YYY=GCP(IV+1,NGCP)**YP(I)
      K=(I-1)*L+NGCP1
      A(K)=XXX*YYY
50    CONTINUE
C
C     SCALE THE MATRIX
      DO 70 I=1,ISTOP
      AMAX=0.D0
```

```
C
      DO 60 J=1,L
      K=(I-1)*L+J
      C=DABS(A(K))
60    AMAX=DMAX1(AMAX,C)
      IF(AMAX.EQ.0.D0) AMAX=1.D0
C
      DO 70 J=1,L
      K=(I-1)*L+J
      A(K)=A(K)/AMAX
70    S(I)=AMAX
C
      CALL DLLSQ(A,B,L,ISTOP,2,X,IPIV,TOL,IER,AUX)
      IER1(II)=IER
      DO 80 I=1,ISTOP
      J=I+ISTOP
      COEF(I,IIII)=X(I)/S(I)
      COEF(I,IIII+1)=X(J)/S(I)
80    CONTINUE
10    CONTINUE
      RETURN
      END
      SUBROUTINE EVPOLY (IFUN, X, Y, ANS)
C
C     EVALUATE POLYNOMIAL FIT FUNCTIONS
C
      DOUBLE PRECISION COEF(21,4), ANSD, XD, YD, XXX, YYY
      INTEGER NTERM(5)/3,6,10,15,21/
      INTEGER XP(21)/0,1,0,2,1,0,3,2,1,0,4,3,2,1,0,5,4,3,2,1,0/
      INTEGER YP(21)/0,0,1,0,1,2,0,1,2,3,0,1,2,3,4,0,1,2,3,4,5/
      COMMON /LSQCFC/ COEF, LSQDEG, IERI(2), TOL
      COMMON / MEANS/ GCPM(4)
C
C     COEF(21,I)= COEFFICIENTS FOR I 1= PIXEL
C                                    2= LINE
C                                    3= EASTING
C                                    4= NORTHING
      ISTOP = NTERM(LSQDEG)
      ANSD = 0.0
      IF (IFUN.EQ.3.OR.IFUN.EQ.4) GO TO 1
      XD = X - GCPM(3)
      YD = Y - GCPM(4)
      GO TO 5
    1 XD = X - GCPM(1)
      YD = Y - GCPM(2)
    5 CONTINUE
C
      DO 10 I=1,ISTOP
      XXX = 1.0
      IF (XP(I).NE.0) XXX = XD**XP(I)
      YYY = 1.0
      IF (YP(I).NE.0) YYY = YD**YP(I)
   10 ANSD = ANSD + COEF(I,IFUN)*XXX*YYY
C
      ANS = ANSD + GCPM(IFUN)
      RETURN
      END
      SUBROUTINE GCPCOR (GCP, MGCP)
C
C     GCPCOR REMOVES MIRROR VELOCITY PROFILE AND EARTH CURVATURE
C     DISTORTIONS PRIOR TO OBTAINING LEAST SQUARES FITS
C
C     GCP IS GROUND CONTROL POINT TABLE INPUT BY USER
C     1.  PIXEL WITHIN LINE
C     2.  LINE WITHIN FRAME
C     3.  EASTING
C     4.  NORTHING
C
      DIMENSION GCP(4,MGCP)
      REAL*4 MVPOFF
      COMMON /LANDST/ CCT, LLC, DEGCEN, SAMPOF, LINOFF, AMPL, PHASE
C
C     ADD BACK DELAY TO GET ORIGINAL CCT PIXEL NUMBER
      WRITE (6,100)
      DO 10 NGCP = 1,MGCP
      WRITE (6,101) GCP(2,NGCP), GCP(1,NGCP)
      LINE = GCP(2,NGCP) + 0.5
      SAMP = GCP(1,NGCP) + DELAY(LINE)
C
C     APPLY MIRROR VELOCITY PROFILE CORRECTION
      OFF = MVPOFF(SAMP)
      GCP(1,NGCP) = GCP(1,NGCP) - OFF
C
C     APPLY EARTH CURVATURE CORRECTION
      CURV = ERCURV(SAMP-OFF)
      GCP(1,NGCP) = GCP(1,NGCP) - CURV
C
      SAMPO = SAMP + SAMPOF
      SHIFT = OFF + CURV
      WRITE (6,102) SAMP, SAMPO, OFF, CURV, SHIFT, GCP(1,NGCP)
   10 CONTINUE
      RETURN
C
  100 FORMAT ('1',20X,'GROUND CONTROL POINT CORRECTIONS'//19X,'OFFSET',
     .5X,'ORIGINAL',3X,'CCT SCENE',7X,'MVP',9X,'CURV',8X,'NET',8X,
     .'CORR.'/7X,'RECORD',3(6X,'SAMPLE'),3(7X,'SHIFT'),6X,'OFFSET'/)
  101 FORMAT (1X,2F12.3)
  102 FORMAT ('+',24X,8F12.3)
      END
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
      FUNCTION DELAY (LINE)
C
C     COMPUTE ROTATIONAL AND SENSOR DELAY RELATIVE TO FIRST SWATH
C
C     DEGCEN - LATITUDE AT THE CENTER OF THE LANDSAT SCENE
C     PIXDLY - EQUATORIAL EARTH ROTATION PER SWATH IN PIXELS
C     DEGPLN - CHANGE IN LATITUDE PER SCAN LINE
C     RADDEG - RADIANS PER DEGREE
C     SDELAY - SENSOR SAMPLING INTERVAL BETWEEN LINES (2) / TOTAL NUMBER
C              OF DETECTORS (25)
C     LINESW - LINE NUMBER IN THE SWATH (1 - 6)
C
      DIMENSION SDELAY(6)
      INTEGER CCTLIN, SWATH
      LOGICAL CCT
      COMMON /LANDST/ CCT, LLC, DEGCEN, SAMPOF, LINOFF, AMPL, PHASE
      DATA PIXDLY, DEGPLN, RADDEG, SDELAY /0.6, 0.00071086, 0.01745329,
     .0.0, 0.08, 0.16, 0.24, 0.32, 0.40/
C
      IF (CCT) GO TO 10
      DELAY=0.0
      RETURN
C
   10 CONTINUE
      CCTLIN = LINE + LINOFF
      DEGLAT = DEGCEN + (1170.5-CCTLIN) * DEGPLN
      RDELAY = PIXDLY * COS (DEGLAT*RADDEG)
      SWATH = (CCTLIN-1)/6 + 1
      LINESW = MOD(CCTLIN-1,6) + 1
      DELAY = RDELAY*(SWATH-1) - SDELAY(LINESW)
      RETURN
      END
      REAL FUNCTION MVPOFF (PS)
C
C     COMPUTE MIRROR VELOCITY PROFILE OFFSET
C
C     LLC = NUMBER OF PIXELS IN THE RAW SCAN LINE
C     SAMPOF = NUMBER OF PIXELS SKIPPED IN THE LANDSAT SCENE
C     AMPL, PHASE = AMPLITUDE, PHASE OF MIRROR VELOCITY PROFILE CURVE
C
      LOGICAL CCT
      COMMON /LANDST/ CCT, LLC, DEGCEN, SAMPOF, LINOFF, AMPL, PHASE
C
      IF (CCT) GO TO 12
      MVPOFF = 0.0
      RETURN
C
   12 CONTINUE
      PS1 = PS + SAMPOF
      MVPOFF = AMPL * SIN (6.2831853 * (PS1+PHASE-1.0) / (LLC-1))
      RETURN
      END
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
      FUNCTION ERCURV (PS)
C
C     COMPUTE EARTH CURVATURE CORRECTION
C
C     RE, RSAT = EARTH RADIUS, SATELLITE ORBIT RADIUS
C     TFOV = TOTAL FIELD OF VIEW OF THE SCANNER (11.56 DEGREES)
C     LLC  = NUMBER OF PIXELS IN THE RAW SCAN LINE
C     SAMPOF = NUMBER OF PIXELS SKIPPED IN THE LANDSAT SCENE
C
      LOGICAL CCT
      COMMON /LANDST/ CCT, LLC, DEGCEN, SAMPOF, LINCFF, AMPL, PHASE
      DATA RE, RSAT, TFOV /6367.4, 7285.6, 0.20176/
C
      IF (CCT) GO TO 10
      ERCURV = 0.0
      RETURN
C
C     COMPUTE SCANNER ANGLE AT PIXEL NO. PS AND ANGLE SUBTENDED AT
C     THE CENTER OF THE EARTH
   10 CONTINUE
      TFOV2 = TFOV/2.0
      PS1 = PS + SAMPOF
      ANGSCN = (PS1-1.0)*TFOV/(LLC-1) - TFOV2
      ANGERT = ARSIN(SIN(ANGSCN)*RSAT/RE) - ANGSCN
      TOTERT = ARSIN (SIN(TFOV2)*RSAT/RE) - TFOV2
C
C     FIND SCANNER ANGLE BASED ON FRACTION OF TOTAL ARC ON THE EARTH'S
C     SURFACE AND CONVERT TO PIXEL NUMBER
      ANGSC1 = TFOV2 * ANGERT / TOTERT
      PS2 = 1.0 + (ANGSC1+TFOV2) * (LLC-1) / TFOV
      ERCURV = PS1 - PS2
      RETURN
      END
```

REGISTRATION - 1
GENERATION OF MAPPING GRID

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

I. <u>NAME</u>

GRIDMP

II. <u>DESCRIPTION</u>

The process of geometric correction requires the computation of image (input) pixel coordinates at each map (output) coordinate by means of a mapping function (generally a low order polynomial). However, the evaluation of a nearly linear function at large numbers of map points is not necessary or practical. The computation required is greatly reduced by evaluating the function at each mesh point in a coarse grid covering the image area. GRIDMP computes the UTM and image coordinates at a specified number of mesh points covering the image. The corrections for the Landsat scanner mirror velocity profile and earth curvature are added to the image coordinates. (These corrections had been removed to reduce nonlinearity in the mapping functions.) The accuracy of interpolation within the grid cells is checked at the center of each cell by comparison with the actual mapping function value. If the error is large, a finer grid or lower order polynomial should be used. GRIDMP also computes the number of lines and samples in the geometrically corrected image.

III. <u>CALLING SEQUENCE</u>

CALL GRIDMP (MAPSET)

where

MAPSET is a LOGICAL variable. If it is TRUE, the output map coordinates are specified. If FALSE, GRIDMP computes the output image size which includes all of the image after registration.

IV. <u>INPUT/OUTPUT</u>

1. INPUT

   Common block MAPDAT includes STRTNO, STOPNO, STRTEA, and STOPEA which are the starting and stopping Easting and Northing coordinates of the output.

Common block PIXDAT contains:

NLINPX, NPIXLN - lines and pixels per line of the input image.

NHGL, NVGL - number of horizontal and vertical grid lines.

SRATE - sampling rate (distance between pixels) in the output map.

NLPI, NSPL - lines and samples in the output image (output of GRIDMP).

KDUM - value to which boundary pixels in the output map are set.

INTERP - interpolation method flag where 1 = nearest neighbor, 2 = bilinear, and 3 = bicubic.

NB - number of bands of data to be interpolated.

2. OUTPUT

Printed output is image and map coordinates at the mesh points, sample and easting values of the first and last points in each grid cell, interpolation errors at the grid cell centers, and the output image size.

V. DESCRIPTION OF SUBROUTINES

Description of the subroutines are given in the following table.

SUBROUTINE	LENGTH (BYTES)	FUNCTION
GRIDMP	3988	Compute image and map coordinates at the mesh points of a coarse grid, and compute errors where interpolating image coordinates between mesh points.
EVPOLY	912	Evaluate a polynomial given its order and coefficients.
DELAY	540	Compute Earth rotation and sensor delay for a given scan line.
MVPOFF MVPINV	374 374	Functions to compute mirror velocity profile offset and its inverse.
ERCURV ERCURI	578 734	Functions to compute Earth curvature correction and its inverse.

VI. PERFORMANCE SPECIFICATIONS

The time required to execute the routine is four milliseconds per cell.

VII. METHOD

GRIDMP computes the ranges of easting and northing occupied by the image. Using the specified number of grid lines and output pixel spacings, the grid spacings in meters and the output image size in pixels are computed. The coordinates at the mesh points are determined by stepping through the UTM grid and transforming to image coordinates using EVPOLY. Since the mapping functions were determined after removing mirror velocity and earth curvature offsets, the inverse offsets must be added to the grid coordinates to obtain the original input image coordinates. In other words, the mirror velocity and earth curvature corrections are applied at the grid points and interpolated linearly between grid points, as are the mapping functions. The image coordinates at the center of each grid cell are calculated by interpolation from the mesh points and by polynomial evaluation in order to determine the errors introduced by the interpolation. A table of output sample numbers and eastings for the first and last points in each grid cell is generated for later use in resampling over the cells.

VIII. COMMENTS

None.

IX. TESTS

A sample printout of grid point coordinates follows.

X. LISTING

```
                         INTERPOLATION GRID INTERSECTIONS
                         ********************************

      SAMPLE              LINE              EASTING              NORTHING

      -4.378             27.314             359.500              3421.500
     144.349             27.143             362.536              3421.500
     293.075             26.973             365.571              3421.500
     441.802             26.802             368.607              3421.500
     590.541             26.632             371.643              3421.500
     739.267             26.461             374.678              3421.500
     887.994             26.291             377.714              3421.500
    1036.721             26.120             380.750              3421.500
    1185.459             25.950             383.786              3421.500
    1334.186             25.779             386.821              3421.500
    1482.913             25.609             389.857              3421.500
    1631.651             25.438             392.893              3421.500
    1780.378             25.268             395.928              3421.500
    1929.105             25.097             398.964              3421.500
    2077.831             24.927             402.000              3421.500

      -3.609            163.586             359.500              3418.741
     145.117            163.416             362.536              3418.741
     293.844            163.245             365.571              3418.741
     442.571            163.075             368.607              3418.741
     591.309            162.904             371.643              3418.741
     740.036            162.733             374.678              3418.741
     888.762            162.563             377.714              3418.741
    1037.489            162.392             380.750              3418.741
    1186.228            162.222             383.786              3418.741
    1334.954            162.051             386.821              3418.741
    1483.681            161.881             389.857              3418.741
    1632.420            161.710             392.893              3418.741
    1781.146            161.540             395.928              3418.741
    1929.873            161.369             398.964              3418.741
    2078.600            161.199             402.000              3418.741

      -2.841            299.846             359.500              3415.983
     145.885            299.676             362.536              3415.983
     294.612            299.505             365.571              3415.983
     443.339            299.334             368.607              3415.983
     592.077            299.164             371.643              3415.983
     740.804            298.993             374.678              3415.983
     889.531            298.823             377.714              3415.983
    1038.257            298.652             380.750              3415.983
    1186.996            298.482             383.786              3415.983
    1335.723            298.311             386.821              3415.983
    1484.449            298.141             389.857              3415.983
    1633.188            297.970             392.893              3415.983
    1781.915            297.800             395.928              3415.983
    1930.641            297.629             398.964              3415.983
    2079.368            297.459             402.000              3415.983

      -2.073            436.106             359.500              3413.224
     146.654            435.936             362.536              3413.224
     295.380            435.765             365.571              3413.224
     444.107            435.594             368.607              3413.224
     592.845            435.424             371.643              3413.224
     741.572            435.253             374.678              3413.224
     890.299            435.083             377.714              3413.224
    1039.025            434.912             380.750              3413.224
    1187.764            434.742             383.786              3413.224
    1336.491            434.571             386.821              3413.224
```

Figure 8. Sample Printout of Grid Point Coordinates in the Landsat and UTM Systems

```
      SUBROUTINE GRIDMP (MAPSET)
C
C     GRIDMP COMPUTES THE INTERPOLATION GRID INTERSECTIONS AND THE
C     INTERPOLATION ERRORS IN IMAGE PIXEL LOCATIONS
C     ALLOWANCE IS MADE FOR THE STARTING NORTHING VALUE GREATER THAN
C     THE STOPPING VALUE (AS IN THE UTM SYSTEM)
C     GRID IS COORDINATES (UTM & PIXEL) OF HORIZONTAL & VERTICAL GRID
C     INTERSECTIONS
C
      DIMENSION GRIDEN(2,30), GRIDSL(2,30,30), NUMPXC(30), CPXVAL(30),
     *NUMLNC(30), ERRCEL(29)
      LOGICAL*1 CELLCG(29,29)
      REAL*8 MVPOFF, MVPINV, MVP, NORTH, NCEN
      LOGICAL MAPSET
      COMMON /PIXDAT/ NLINPX, NPIXLN, NHGL, NVGL, SRATE, NLPI, NSPL,
     *                NDUM, INTERP, NB
      COMMON /MAPDAT/ STRTNO, STOPNO, STRTEA, STOPEA
      COMMON /  GRID/ GRIDEN, GRIDSL, NUMPXC, CPXVAL, NUMLNC, CELLCG
C
C     COMPUTE UTM COORDINATES AT IMAGE CORNERS
      IF (MAPSET) GO TO 900
      FLINPX=NLINPX
      FPIXLN=NPIXLN
      MVP = MVPOFF(1.0)
      CRV = ERCURV(1.0-MVP)
      PS = 1.0 - DELAY(1) - MVP - CRV
      CALL EVPOLY (3, PS, 1.0, AE)
      CALL EVPOLY (4, PS, 1.0, AN)
      PS = 1.0 - DELAY(NLINPX) - MVP - CRV
      CALL EVPOLY (3, PS, FLINPX, CE)
      CALL EVPOLY (4, PS, FLINPX, CN)
      MVP = MVPOFF(FPIXLN)
      CRV = ERCURV(FPIXLN-MVP)
      PS = FPIXLN - DELAY(1) - MVP - CRV
      CALL EVPOLY (3, PS, 1.0, BE)
      CALL EVPOLY (4, PS, 1.0, BN)
      PS = FPIXLN - DELAY(NLINPX) - MVP - CRV
      CALL EVPOLY (3, PS, FLINPX, DE)
      CALL EVPOLY (4, PS, FLINPX, DN)
      IF (AN.LT.CN) STRTNO = AMIN1 (AN, BN)
      IF (AN.GT.CN) STRTNO = AMAX1 (AN, BN)
      IF (AN.LT.CN) STOPNO = AMAX1 (CN, DN)
      IF (AN.GT.CN) STOPNO = AMIN1 (CN, DN)
      STRTEA = AMIN1 (AE, CE)
      STOPEA = AMAX1 (BE, DE)
C
C     COMPUTE GX AND GY GRID SPACING AND OUTPUT IMAGE SIZE
  900 NHGL1=NHGL-1
      NVGL1=NVGL-1
      GX = (STOPEA-STRTEA) / NVGL1
      GY = (STOPNO-STRTNO) / NHGL1
      NSPL = (STOPEA-STRTEA+SRATE) / SRATE + 0.5
      NLPI = (ABS(STOPNO-STRTNO)+SRATE) / SRATE + 0.5
C
C     CALCULATE GRID INTERSECTIONS BY STEPPING THROUGH UTM GRID AND
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
C     TRANSFORMING TO IMAGE (PIXEL) COORDINATES
        DO 1010 I=1,NVGL
        EAST = STRTEA + (I-1)*GX
        GRIDEN(1,I) = EAST
          DO 1000 J=1,NHGL
          NORTH = STRTNO + (J-1)*GY
          GRIDEN(2,J) = NORTH
          CALL EVPOLY (1, EAST, NORTH, PS)
          CALL EVPOLY (2, EAST, NORTH, PL)
C
C         ADD INVERSES OF MIRROR VELOCITY PROFILE AND EARTH CURVATURE
C         CORRECTIONS TO GET RAW PIXEL COORDINATES AT GRID POINTS
          LINE = PL + SIGN(0.5,PL)
          PSCCT = PS + DELAY(LINE)
          MVP = MVPINV(PSCCT)
          CRV = ERCURI(PSCCT+MVP)
          PS = PS + MVP + CRV
          GRIDSL(1,J,I) = PS
          GRIDSL(2,J,I) = PL
 1000     CONTINUE
 1010   CONTINUE
      WRITE (6,1050)
      DO 1011 J=1,NHGL
 1011 WRITE (6,1051) (GRIDSL(1,J,K), GRIDSL(2,J,K), GRIDEN(1,K),
     ,GRIDEN(2,J), K=1,NVGL)
C
C       COMPUTE FINAL GRID ERRORS
          WRITE (6,1065)
        ERRMAX=0.
        DO 1030 I4=2,NHGL
        I41=I4-1
          DO 1020 J4=2,NVGL
          J41=J4-1
C
C         COMPUTE PIXEL COORDINATES BY BILINEAR INTERPOLATION IN A CELL
          PSB = (GRIDSL(1,I4,J4)+GRIDSL(1,I41,J4)+GRIDSL(1,I41,J41)
     .    +GRIDSL(1,I4,J41)) / 4.0
          PLB = (GRIDSL(2,I4,J4)+GRIDSL(2,I41,J4)+GRIDSL(2,I41,J41)
     .    +GRIDSL(2,I4,J41)) / 4.0
C
C         COMPUTE CENTER EASTING AND NORTHING
          ECEN = (GRIDEN(1,J4)+GRIDEN(1,J41)) / 2.0
          NCEN = (GRIDEN(2,I4)+GRIDEN(2,I41)) / 2.0
C
C         COMPUTE PIXEL COORDINATES BY FUNCTION
          CALL EVPOLY (1, ECEN, NCEN, PS)
          CALL EVPOLY (2, ECEN, NCEN, PL)
C
C         ADD INVERSES OF MIRROR VELOCITY PROFILE AND EARTH CURVATURE
C         CORRECTIONS TO GRID POINTS
          LINE = PL + SIGN(0.5,PL)
          PSCCT = PS + DELAY(LINE)
          MVP = MVPINV(PSCCT)
          CRV = ERCURI(PSCCT+MVP)
          PS = PS + MVP + CRV
```

```
C
C           COMPUTE ERROR
            ERR=SQRT((PSB-PS)**2+(PLB-PL)**2)
            ERRCEL(J41) = ERR
            ERRMAX = AMAX1 (ERRMAX, ERR)
 1020       CONTINUE
         WRITE (6,1070) (ERRCEL(J), J=1,NVGL1)
 1030    CONTINUE
      WRITE (6,1060) ERRMAX
C
C     COMPUTE TABLES OF FIRST OUTPUT SAMPLE NUMBERS AND COORDINATES AND
C     OUTPUT LINE NUMBERS IN EACH CELL
      NUMPXC(1) = 1
      CPXVAL(1) = GRIDEN(1,1)
      NUMLNC(1) = 1
      DO 1038 J=2,NVGL1
      NUMPXC(J) = (GRIDEN(1,J)-GRIDEN(1,1)+SRATE)/SRATE + 1.0
 1038 CPXVAL(J) = GRIDEN(1,1) + (NUMPXC(J)-1)*SRATE
      DO 1039 J=2,NHGL1
 1039 NUMLNC(J) = (ABS(GRIDEN(2,J)-GRIDEN(2,1))+SRATE)/SRATE + 1.0
      NUMPXC(NVGL) = NSPL + 1
      NUMLNC(NHGL) = NLPI + 1
C
      WRITE (6,1052)
      DO 1045 J=1,29
      IF (J.LE.NHGL1.OR.J.LE.NVGL1) WRITE (6,1054) J
      IF (J.LE.NVGL1) WRITE (6,1055) NUMPXC(J), CPXVAL(J)
 1045 IF (J.LE.NHGL1) WRITE (6,1056) NUMLNC(J)
      WRITE (6,1053) SRATE, NLPI, NSPL
      RETURN
C
 1050 FORMAT ('1',30X,'INTERPOLATION GRID INTERSECTIONS'/31X,32('*')//
     .14X,'SAMPLE',16X,'LINE',13X,'EASTING',12X,'NORTHING')
 1051 FORMAT (/(4F20.3))
 1052 FORMAT ('1'/30X,'GRID CELL STARTING POINTS'/30X,25('*')//10X,'CELL
     . NO.',5X,'SAMPLE NO.',5X,'COORD. VALUE',5X,'LINE NO.'/)
 1053 FORMAT (///21X,'OUTPUT PIXEL SPACING =',F7.3,'  UNITS'/21X,'OUTPUT
     . IMAGE SIZE =',I5,' LINES BY',I5,' SAMPLES')
 1054 FORMAT (I15)
 1055 FORMAT ('+',I30,F18.3)
 1056 FORMAT ('+',I60)
 1060 FORMAT (//21X,'MAXIMUM GRID ERROR =',1PE10.2,'  PIXELS')
 1065 FORMAT ('1',20X,'MAGNITUDE OF INTERPOLATION ERROR AT GRID CELL CEN
     .TERS'/21X,53('*'))
 1070 FORMAT (/(1P14F9.2))
      END
```

REGISTRATION - 2
PARTITIONING OF LARGE IMAGES BY CELLS

I. <u>NAME</u>

CELLMP

II. <u>DESCRIPTION</u>

When generating an output line of geometrically corrected imagery, the input data required will generally be drawn from several scan lines. Thus, a block of input data must be held in core. If the input scan lines are long, very large amounts of core are required, and the image may require segmenting. CELLMP segments the image on the basis of the grid cells produced by GRIDMP.

III. <u>CALLING SEQUENCE</u>

CALL CELLMP (CORE)

where CORE is an integer variable specifying the amount of core available for the input data block.

IV. <u>INPUT/OUTPUT</u>

The printed output is a matrix showing the segment number in which each grid cell is completed. Cells not requiring input data are indicated by "**".

V. <u>DESCRIPTION OF SUBROUTINES</u>

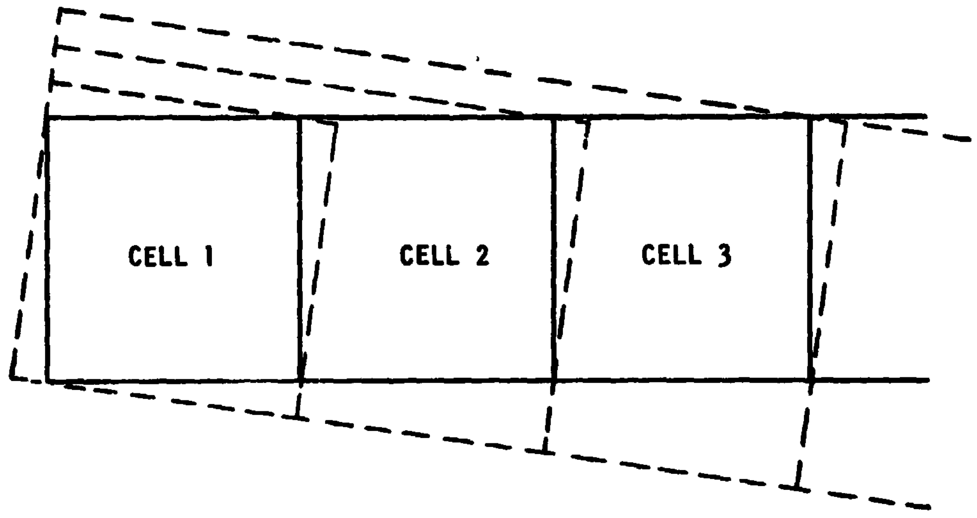
CELLMP calls the function DELAY.

VI. <u>PERFORMANCE SPECIFICATION</u>

The subroutine requires 2716 bytes of storage. For an image requiring four segments, the partitioning information is computed in approximately one second.

VII. <u>METHOD</u>

For each row of grid cells, CELLMP computes the range of input coordinates required for including successive cells along the row, as illustrated in the following Figure.



The dotted lines indicate the extents of the input data required for 1, 2, and 3 cells in the row. The actual input coordinates at the cell corners are adjusted to allow for the following factors:

- o bicubic interpolation requires +2 and -1 additional neighboring samples,
- o earth rotation and sensor delay shift the sample numbers in each line, and
- o one additional sample (± 1) to allow for delay shifts in lines adjacent to the cell corners.

For all rows of cells, a common number of input samples per line is determined in order to define the input image block. Using this information, the program again loops over the rows of cells and finds the starting input rows and columns and number of rows and columns in each input data segment.

VIII. COMMENTS

None.

IX. TESTS

The routine has been tested by printing the input data ranges for the cells and comparing with the cell coordinates produced by GRIDMP.

X. LISTING

The listing follows.

```
      SUBROUTINE CELLPP (CORE)
C
C     COMPUTE DIMENSIONS OF INPUT ARRAY WHICH CAN BE HELD IN CORE AT A
C      TIME AND NUMBER OF PARTITIONS REQUIRED.
C     PIX AND ROW VARIABLES HAVE DIRECTIONS APPENDED, E.G. PIXNW IS
C     NORTHWEST CORNER PIXEL OF A CELL
C
      INTEGER ROWNW, PIXNW, ROWNE, PIXNE, ROWSE, PIXSE, ROWSW, PIXSW,
     *RN, RS, SW, SE, R1, R2, S1, S2, CORE, RSTRT
      DIMENSION GRIDEN(2,30), GRIDSL(2,30,30), NUMPXC(30), CPXVAL(30),
     *NUMLNC(30)
      LOGICAL*1 CELLCG(29,29)
      COMMON /PIXDAT/ NLINPX, NPIXLN, NHGL, NVGL, SRATE, NLPI, NSPL,
     *                KDUM, INTRP, NB
      COMMON /  GRID/ GRIDEN, GRIDSL, NUMPXC, CPXVAL, NUMLNC, CELLCG
      COMMON /SEGMNT/ NCGP, NBSAM(10), NBLIN(10), NSTRT(10),
     *                RSTRT(29,10)
C
C     INITIALIZE CELL COLUMN GROUP INDICATOR (CELLCG)
      NHGL1 = NHGL - 1
      NVGL1 = NVGL - 1
      DO 1100 I=1,NHGL1
 1100 CELLCG(I,1) = 222
C
C     FIND NCGP, THE NUMBER OF COLUMN GROUPS,
C     S1 IS FIRST INPUT PIXEL FOR THIS COLUMN GROUP
      NCGP = 0
      S1 = 1
C
C     START A NEW COLUMN GROUP OR SEGMENT
 1800 CONTINUE
      NCGP = NCGP + 1
      MINNC = 1000000
C
C     LOOP OVER CELL ROWS
      DO 2010 I=1,NHGL1
      R1 = 1000000
      R2 = 0
      S2 = 0
C
C       FIND FIRST CELL IN THE ROW AVAILABLE FOR THIS COLUMN GROUP AND
C       START THE LOOP THERE (CELL IC1)
        DO 2001 IC1=1,NVGL1
        IF (CELLCG(I,IC1).EQ.222) GO TO 2002
 2001   CONTINUE
        GO TO 2010
C
C         FIND RANGE OF IMAGE COORDINATES REQUIRED FOR THE OUTPUT CELL
 2002     DO 2000 J=IC1,NVGL1
          ROWNW = GRIDSL(2,I,J) - 1.0
          PIXNW = GRIDSL(1,I,J) + DELAY(ROWNW) - 2.0
          ROWNE = GRIDSL(2,I,J+1) - 1.0
          PIXNE = GRIDSL(1,I,J+1) + DELAY(ROWNE) + 3.0
          ROWSE = GRIDSL(2,I+1,J+1) + 2.0
          PIXSE = GRIDSL(1,I+1,J+1) + DELAY(ROWSE) + 3.0
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
          ROWSW = GRIDSL(2,I+1,J) + 2.0
          PIXSW = GRIDSL(1,I+1,J) + DELAY(ROWSW) - 2.0
          RN = MAX0 (MIN0(ROWNW,ROWNE), 1)
          RS = MIN0 (MAX0(ROWSW,ROWSE), NLINPX)
          SW = MAX0 (MIN0(PIXNW,PIXSW), 1)
          SE = MIN0 (MAX0(PIXNE,PIXSE), NPIXLN)
C
C         CHECK FOR IMAGE COORDINATES OUTSIDE THE INPUT IMAGE
          IF (RN.GT.NLINPX.OR.RS.LT.1.OR.SW.GT.NPIXLN.OR.SE.LT.1)
     .    GO TO 1920
C
C           FIND RANGE OF IMAGE COORDINATES FOR THIS ROW OF CELLS AND
C           FIND NO. OF OUTPUT CELLS THAT CAN BE FILLED GIVEN CORE LIMIT
            R1 = MIN0 (R1, RN)
            R2 = MAX0 (R2, RS)
            S2 = MAX0 (S2, SE)
            NR = R2 - R1 + 1
            NC = S2 - S1 + 1
            IF (NR*NC.LE.CORE) GO TO 2005
C
C           CORE REQUIRED EXCEEDS THAT AVAILABLE
C           GO TO NEXT ROW OF CELLS
            MINNC = MIN0 (NCSAVE, MINNC)
            CELLCG(I,J) = 255
            GO TO 2010
C
 1920     CELLCG(I,J) = 0
          GO TO 2000
C
C           CORE IS AVAILABLE,  SAVE REQUIRED DIMENSIONS.
 2005       NCSAVE = NC
            CELLCG(I,J) = 222
 2000       CONTINUE
 2010 CONTINUE
C
      LASTPX = S1 + MINNC - 1
      IF (LASTPX.LT.NPIXLN) GO TO 2100
      LASTPX = NPIXLN
      MINNC = NPIXLN - S1 + 1
 2100 CONTINUE
C
C     NOW USE NUMBER OF INPUT COLUMNS AVAILABLE TO SET CELL SEGMENT
C     INDICATORS AND TO FIND ROW AND COLUMN INFORMATION
      MINS1 = 1000000
C
C     LOOP OVER CELL ROWS
      DO 2150 I=1,NHGL1
      IC1 = 100
      IC2 = 0
      RSTRT(I,NCGP) = 0
C
          DO 2130 J=1,NVGL1
C
C         CHECK WHETHER THE CELL REQUIRES INPUT OUTSIDE OF THE IMAGE
          IF (CELLCG(I,J).EQ.0) GO TO 2126
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
C
C          JUMP OUT IF THE CELL REQUIRES TOO MUCH CORE.
           IF (CELLCG(I,J).EQ.255) GO TO 2140
C
C          IF THE CELL USES INPUT DATA, CHECK FOR THE LAST PIXEL
           IF (CELLCG(I,J).EQ.222) GO TO 2125
           GO TO 2130
C
C          JUMP OUT IF THE CELL REQUIRES SAMPLES PAST LASTPX.
 2125      ROWNE = GRIDSL(2,I,J+1) - 1.0
           PIXNE = GRIDSL(1,I,J+1) + DELAY(ROWNE) + 3.0
           ROWSE = GRIDSL(2,I+1,J+1) + 2.0
           PIXSE = GRIDSL(1,I+1,J+1) + DELAY(ROWSE) + 3.0
           SE = MIN0 (MAX0(PIXNE,PIXSE), NPIXLN)
           IF (SE.GT.LASTPX) GO TO 2140
C
C          FIND FIRST AND LAST CELLS THAT USE INPUT DATA
           IC1 = MIN0 (J, IC1)
           IC2 = MAX0 (J, IC2)
C
C          ELSE SET COLUMN GROUP INDICATOR TO GROUP NUMBER.
           CELLCG(I,J) = NCGP
           GO TO 2130
C
 2126      CELLCG(I,J) = NCGP + 100
 2130      CONTINUE
C
C          FIND FIRST INPUT ROW FOR THIS ROW OF CELLS
 2140      IF (IC2.EQ.0) GO TO 2150
           ROWNW = GRIDSL(2,I,IC1) - 1.0
           ROWNE = GRIDSL(2,I,IC2+1) - 1.0
           RN = MAX0 (MIN0(ROWNW,ROWNE), 1)
           RSTRT(I,NCGP) = RN
C
C          FIND STARTING INPUT COLUMN FOR NEXT SEGMENT
           IF (J.EQ.NVGL) GO TO 2150
           NEXTCL = IC2 + 1
           ROWNW = GRIDSL(2,I,NEXTCL) - 1.0
           PIXNW = GRIDSL(1,I,NEXTCL) + DELAY(ROWNW) - 2.0
           ROWSW = GRIDSL(2,I+1,NEXTCL) + 2.0
           PIXSW = GRIDSL(1,I+1,NEXTCL) + DELAY(ROWSW) - 2.0
           SW = MAX0 (MIN0(PIXNW,PIXSW), 1)
           MINS1 = MIN0 (SW, MINS1)
C
C          SET INDICATOR FOR STARTING CELL IN NEXT SEGMENT
           CELLCG(I,NEXTCL) = 222
 2150 CONTINUE
C
C     SET PARAMETERS OF INPUT DATA SEGMENT
      MAXNR = MIN0 (CORE/MINNC, NLINPX)
      NBSAM(NCGP) = MINNC
      NBLIN(NCGP) = MAXNR
      NSTRT(NCGP) = S1
      S1 = MINS1
C
```

```
C     IF ALL CELLS NOT DONE, START A NEW SEGMENT
      IF (LASTPX.LT.NPIXLN) GO TO 1800
C
      WRITE (6,100)
      DO 2160 I=1,NHGL1
 2160 WRITE (6,101) (CELLCG(I,J), J=1,NVGL1)
      RETURN
C
  100 FORMAT (///5X,'COLUMN GROUP NUMBERS FOR THE GRID CELLS'/5X,
     ,39('*')/)
  101 FORMAT (29(1X,I2))
      END
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

REGISTRATION - 3
PERFORM DATA HANDLING AND INTERPOLATION

I. NAME

RECTFI

II. DESCRIPTION

This routine controls the reading of the input data, finds the equations for linear interpolation of the mapping functions within the cells, calls the interpolation routine, and assembles the segments of the output image if segmenting was required.

III. CALLING SEQUENCE

Call RECTFI (INPIX, PIXEL, PIXOUT)

where

INPIX is a buffer array for reading input scan lines,

PIXEL is the array for holding the input data segment, and

PIXOUT is the output buffer array.

IV. INPUT/OUTPUT

1. INPUT

   The input is a sequential data set on unit 10.

2. OUTPUT

   The output is a sequential data set on unit 11.

3. FILE STORAGE

   The output segments are written on sequential data sets having unit numbers 21, 22, 23 . . . (20 + number of segments).

V. DESCRIPTION OF SUBROUTINES

The subroutines required are listed in the following table.

DESCRIPTION OF SUBROUTINES

SUBROUTINE OR ENTRY	LENGTH (BYTES)	FUNCTION
RECTFI	3494	Read and write data, compute polynominal interpolation functions, call data interpolation routine.
LOADLN	4036	Load input image data into array. Set up pointer array to location of lines in the data block.
RESAMP	2664	Apply Earth rotation and sensor delay corrections. Compute interpolated data values.
READAR WRITAR	474	Read and write arrays of specified length in sequential unformatted files.

VI. PERFORMANCE SPECIFICATION

1. STORAGE

The storage requirement is primarily dictated by the arrays which are arguments to RECTFI. In addition, RECTFI uses common blocks of the following lengths in bytes:

PIXDAT	-	40
GRID	-	8461
SEGMNT	-	1284
PIXIN	-	2032
TRANS	-	24

2. EXECUTION TIME

The processing speed is highly dependent on the interpolation method. Using four band Landsat data, the speeds are:

Nearest Neighbor	2350 Pixels/Second
Bilinear	1100 Pixels/Second
Bicubic	450 Pixels/Second

VII. METHOD

For each segment and each row of grid cells, subroutine LOADLN is called to load the required input data into array PIXEL. For each cell in a row, the input image coordinates at the cell corners are obtained from the common block GRID (computed by GRIDMP). The intersections of the output line with the cell edges are computed and from this the linear equation for the input coordinates within the cell. The resampling routine is then called and the output line (or line segment) is written out. If the image was segmented, the segments are read, assembled, and written out.

VIII. COMMENTS

The routines GRIDMP, CELLMP, and RECTFI are all required for registration but were treated separately because it is useful to call them separately in designing computer runs. GRIDMP determines the size of the output image for a given tranformation and sampling rate; CELLMP determines the number of partitions required for a given core availability.

IX. TESTS

The program was tested by examining the registration of small test areas.

X. LISTINGS

Listings of the subroutines follow.

```
      SUBROUTINE RECTFI (INPIX, PIXEL, PIXOUT)
C
C      *CALL RESAMPLING ROUTINE TO COMPUTE INPUT PIXEL LOCATIONS AND
C       PERFORM SPECIFIED INTERPOLATION METHOD.
C      *READ LINE SEGMENTS AND WRITE ASSEMBLED RECORDS.
C
C     GRID IS COORDINATES (UTM & PIXEL) OF HORIZONTAL & VERTICAL GRID
C     INTERSECTIONS
C     UTMPIX IS THE FOLLOWING UTM TO PIXEL TRANSFORMATION PARAMETERS:
C     1. PIXEL COORDINATE OF FIRST SAMPLE IN SEGMENT
C     2. LINE COORDINATE OF FIRST SAMPLE IN SEGMENT
C     3. SLOPE OF INPUT PIXEL VS. OUTPUT PIXEL EQUATION
C     4. SLOPE OF INPUT  LINE VS. OUTPUT PIXEL EQUATION
C     5. NUMBER OF OUTPUT PIXELS IN THE CELL
C     6. OUTPUT PIXEL NUMBER.
C
      LOGICAL*1 INPIX(1), PIXEL(1), PIXOUT(NB,NSPL), CELLCG(29,29)
      DIMENSION GRIDEN(2,30), GRIDSL(2,30,30), NUMPXC(30), CPXVAL(30),
     .NUMLNC(30)
      INTEGER SAMPSG(29,10), RSTRT, PSTRT, SAMPO
      INTEGER*2 MNDEX(1000)
      REAL NORTH
      COMMON /PIXDAT/ NLINPX, NPIXLN, NHGL, NVGL, SRATE, NLPI, NSPL,
     .                KDUM, INTERP, NB
      COMMON /  GRID/ GRIDEN, GRIDSL, NUMPXC, CPXVAL, NUMLNC, CELLCG
      COMMON /SEGMNT/ NCGP, NBSAM(10), NBLIN(10), PSTRT(10),
     .                RSTRT(29,10)
      COMMON / PIXIN/ MAXSP, MAXLN, MNDEX, SAMPO, LINEO
      COMMON / TRANS/ UTMPIX(6)
      EQUIVALENCE (NPIX, UTMPIX(5)), (ISAMP, UTMPIX(6))
C
      NHGL1 = NHGL - 1
      NVGL1 = NVGL - 1
      IF (GRIDEN(2,2).GT.GRIDEN(2,1)) SRATEN =  SRATE
      IF (GRIDEN(2,2).LT.GRIDEN(2,1)) SRATEN = -SRATE
C
C     LOOP OVER COLUMN GROUPS
      DO 1020 ICGP=1,NCGP
      MAXSP = NBSAM(ICGP)
      MAXLN = NBLIN(ICGP)
      SAMPO = PSTRT(ICGP)
      LUNIT= ICGP + 20
      REWIND LUNIT
      CALL LOADLN (.TRUE., 1, MAXLN, INPIX, PIXEL)
C
C     LOOP OVER OUTPUT IMAGE CELLS
      DO 1010 IGY1=1,NHGL1
      IGY2 = IGY1 + 1
      LSTRT = NUMLNC(IGY1)
      LSTOP = NUMLNC(IGY2) - 1
C
C        LOAD LINES OF CCT NECESSARY FOR RESAMPLED LINE
         LMIN = RSTRT(IGY1,ICGP)
         IF (LMIN.EQ.0) GO TO 150
         LMAX = MINO (LMIN+MAXLN-1, NLINPX)
```

```
          CALL LOADLN (.FALSE., LMIN, LMAX, INPIX, PIXEL)
C
C     LOOP OVER OUTPUT IMAGE LINES IN THE CELL
  150 DO 1000 ILINE=LSTRT,LSTOP
      NORTH = GRIDEN(2,1) + (ILINE-1)*SRATEN
C
C     LOOP OVER GRID CELLS COVERING MAP AREA
      ISAMP=0
      DO 300 IGX1=1,NVGL1
      IGX2 = IGX1 + 1
C
C             FIND NUMBER OF OUTPUT PIXELS IN THE CELL
              NPIX = NUMPXC(IGX2) - NUMPXC(IGX1)
C
C             CHECK FOR CELLS NOT REQUIRING INPUT DATA
              IF (CELLCG(IGY1,IGX1).EQ.ICGP+100) GO TO 250
              IF (CELLCG(IGY1,IGX1).EQ.ICGP) GO TO 210
              GO TO 300
C
C     LOAD UTM COORDINATES OF GRID CELL CORNERS
  210 AX = GRIDEN(1,IGX1)
      BX = GRIDEN(1,IGX2)
      AY = GRIDEN(2,IGY1)
      CY = GRIDEN(2,IGY2)
C
C     LOAD PIXEL COORDINATES OF GRID CELL CORNERS
      AS = GRIDSL(1,IGY1,IGX1)
      AL = GRIDSL(2,IGY1,IGX1)
      BS = GRIDSL(1,IGY1,IGX2)
      BL = GRIDSL(2,IGY1,IGX2)
      CS = GRIDSL(1,IGY2,IGX1)
      CL = GRIDSL(2,IGY2,IGX1)
      DS = GRIDSL(1,IGY2,IGX2)
      DL = GRIDSL(2,IGY2,IGX2)
C
C     COMPUTE FRACTIONAL CELL DISTANCE ALONG NORTHING AXIS
      ACRATC = (AY-NORTH) / (AY-CY)
C
C     COMPUTE INTERPOLATED SAMPLE AND LINE VALUES AT CELL EDGES
      GS = AS + ACRATC*(CS-AS)
      GL = AL + ACRATC*(CL-AL)
      HS = BS + ACRATC*(DS-BS)
      HL = BL + ACRATC*(DL-BL)
C
C     COMPUTE SLOPE OF INPUT SAMPLES AND LINES WITHIN THE CELL
      DELTAE = BX - AX
      UTMPIX(3) = (HS-GS) / DELTAE
      UTMPIX(4) = (HL-GL) / DELTAE
C
C             FIND INPUT COORDINATES OF FIRST POINT IN THIS GRID CELL
              UX1 = CPXVAL(IGX1) - AX
              UTMPIX(1) = GS + UX1*UTMPIX(3)
              UTMPIX(2) = GL + UX1*UTMPIX(4)
C
C             INTERPOLATE OVER INPUT DATA
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
          CALL RESAMP (PIXEL, PIXOUT)
          GO TO 300
C
C         FILL IN THE CELL NOT REQUIRING INPUT DATA
  250     DO 255 I=1,NPIX
          ISAMP = ISAMP + 1
          DO 255 J=1,NB
  255     PIXOUT(J,ISAMP) = KDUM
  300     CONTINUE
C
C        WRITE THE LINE SEGMENT OUT
         IF (NCGP.EQ.1) GO TO 500
         IF (ISAMP.EQ.0) GO TO 1010
         CALL WRITAR (LUNIT, PIXOUT, NB*ISAMP)
         GO TO 1000
C
  500    CONTINUE
         WRITE (11) PIXOUT
 1000    CONTINUE
C
 1010  SAMPSG(IGY1,ICGP) = ISAMP
C
      REWIND LUNIT
      WRITE (6,1700) LSTOP, ICGP
 1020 CONTINUE
C
C     ASSEMBLE AND WRITE OUTPUT IMAGE
      IF (NCGP.EQ.1) RETURN
      DO 1050 IGY1=1,NMGL1
      LSTRT = NUMLNC(IGY1)
      LSTOP = NUMLNC(IGY1+1) - 1
C
C     ASSEMBLE AND WRITE OUT THE LINE SEGMENTS
      DO 1040 ILINE=LSTRT,LSTOP
      NS = 1
      DO 1030 ICGP=1,NCGP
      NSAMP = SAMPSG(IGY1,ICGP)
      IF (NSAMP.EQ.0) GO TO 1030
      LUNIT= ICGP + 20
      CALL READAR (LUNIT, PIXOUT(1,NS), NB*NSAMP)
      NS = NS + NSAMP
 1030 CONTINUE
 1040 WRITE (11) PIXOUT
 1050 CONTINUE
      RETURN
C
 1700 FORMAT(' FINISHED PROCESSING'I5,' RECORDS IN COLUMN GROUP'I3)
      END
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```
      SUBROUTINE LOADLN (FIRST, LMIN, LMAX, INPIX, PIXEL)
C
C     THIS SUBROUTINE LOADS ARRAY 'PIXEL' WITH INPUT IMAGE LINES
C     LMIN THROUGH LMAX, IT FIRST CHECKS WHICH LINES ARE ALREADY
C     LOADED (L1 THROUGH L2) TO DETERMINE WHICH LINES CAN BE LEFT
C     IN MEMORY AND WHICH MUST BE READ.
C     LNDEX(I) IS THE LINE NUMBER OF DATA STORED IN PIXEL(*,I)
C     MNDEX(L) IS STORAGE LOCATION OF LINE NUMBER 'L'
C     IPOSN IS THE LINE NUMBER TAPE IS POSITIONED AT
C
      LOGICAL*1 INPIX(NB,NPIXLN), PIXEL(NB,NBSAM,MAXLN)
      INTEGER*2 LNDEX(1000),MNDEX(1000), SAMPO*8
      LOGICAL EOF, FIRST
      COMMON /PIXDAT/ NLINPX, NPIXLN, NHGL, NVGL, SRATE, NLPI, NBPL,
     *                KDUM, INTERP, NB
      COMMON / PIXIN/ NBSAM, MAXLN, MNDEX, SAMPO, LINEO
C
C     INITIALIZE BY CALLING LAST LINE IN CORE LINE '0'
      IF (.NOT.FIRST) GO TO 200
      NBSAMB = NB*NBSAM
      REWIND 10
      EOF = .FALSE.
      IPOSN = 1
      L2 = MAXLN
      L1 = 1
      LNDEX(L1)=1-MAXLN
      LNDEX(L2)=0
      GO TO 1200
C
C     JFIRST, JLAST - FIRST AND LAST LINES ALREADY LOADED
  200 IF (EOF) RETURN
      JFIRST=LNDEX(L1)
      JLAST=LNDEX(L2)
      IF(LMIN.GE.JFIRST) GO TO 211
      WRITE(6,190) LMIN,LMAX,JFIRST,JLAST
      STOP 41
C
C     NLINRD - NUMBER OF LINES REQUIRED TO FILL IN FROM JLAST TO LMAX
C     JNEW   - NEW FIRST LINE AFTER LOADING NLINRD LINES FROM JFIRST
211   CONTINUE
      NLINRD=LMAX-JLAST
      JNEW=JFIRST+NLINRD
      IF(NLINRD.LE.0) GO TO 1200
      IF(NLINRD.GE.MAXLN) GO TO 300
C
C     FIND INDEX OF LINE JNEW (I)
C     LMINX1 - INDEX OF LAST LINE TO BE READ IN (I-1)
      DO 210 I=1,MAXLN
      IF(LNDEX(I).NE.JNEW) GO TO 210
      LMINX1 = I - 1
      IF(LMINX1.LE.0)LMINX1=MAXLN
      GO TO 220
  210 CONTINUE
  220 CONTINUE
      IF(L1.GT.LMINX1) GO TO 240
```

```
C
C *** CASE 1 - LOAD DATA FROM L1 TO LMINX1
      DO 230 I=L1,LMINX1
      READ (10,END=1300,ERR=1400) INPIX
      CALL MVL (INPIX(1,SAMPO), PIXEL(1,1,I), NBSAM4)
      LNDEX(I)=IPOSN
      IPOSN = IPOSN + 1
  230 CONTINUE
      L1 = LMINX1 + 1
      L2 = LMINX1
      IF(L1.GT.MAXLN) L1 =1
      GO TO 1000
C
C *** CASE 2 - LOAD DATA FROM L1 THROUGH MAXLN AND 1 THROUGH LMINX1
  240 DO 250 I = L1,MAXLN
      READ (10,END=1300,ERR=1400) INPIX
      CALL MVL (INPIX(1,SAMPO), PIXEL(1,1,I), NBSAM4)
      LNDEX(I) = IPOSN
      IPOSN = IPOSN + 1
  250 CONTINUE
      DO 260 I = 1,LMINX1
      READ (10,END=1300,ERR=1400) INPIX
      CALL MVL (INPIX(1,SAMPO), PIXEL(1,1,I), NBSAM4)
      LNDEX(I) = IPOSN
      IPOSN = IPOSN + 1
  260 CONTINUE
      L1 = LMINX1 + 1
      L2 = LMINX1
      GO TO 1000
C
C *** CASE 3 - NO OVERLAP OF OLD AND NEW DATA
C     POSITION TAPE
  300 IF(IPOSN.EQ.JNEW)GO TO 320
      READ (10,END=1300,ERR=1400)
      IPOSN =IPOSN + 1
      GO TO 300
  320 DO 330 I=1,MAXLN
      READ (10,END=1300,ERR=1400) INPIX
      CALL MVL (INPIX(1,SAMPO), PIXEL(1,1,I), NBSAM4)
      LNDEX(I) = IPOSN
      IPOSN = IPOSN + 1
  330 CONTINUE
      L1 = 1
      L2  = MAXLN
C
C     LOAD STORAGE LOCATIONS INTO ELEMENT OF MNDEX EQUAL TO LINE NUMBER
C     LINEO - FIRST IMAGE LINE HELD IN CORE
C     MX    - STORAGE LINE NUMBER AT IMAGE LINE LNDEX(I)
 1000 LINEO = LNDEX(L1)
      DO 1100 I = 1,MAXLN
      MX = LNDEX(I) - LINEO + 1
      MNDEX(MX) = I
 1100 CONTINUE
 1200 CONTINUE
      RETURN
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```
C
 1300 EOF = .TRUE.
      WRITE(6,1301) IPOSN
      RETURN
 1400 WRITE (6,1401) IPOSN
      STOP 43
C
190   FORMAT(1H0,'ERROR: BACKWARD READ REQUESTED',4I10)
1301  FORMAT(1H0,'EOF AT LINE ',I5)
1401  FORMAT(1H0,'READ ERROR AT LINE',I5)
      END
      SUBROUTINE RESAMP (LPIXEL, LPXOUT)
C
C     INTERP = 1, 2, 3  GIVES NEAREST, BILINEAR, BICUBIC INTERPOLATION
C
      LOGICAL*1 LPIXEL(NB,NBSAM,MAXLN), LPXOUT(NB,1)
      DIMENSION HK(4,4)
      INTEGER SAMPO, SLR
      INTEGER*2 MNDEX(1000)
C
C     UTMPIX IS THE FOLLOWING UTM TO PIXEL TRANSFORMATION PARAMETERS:
C     1. PIXEL COORDINATE OF FIRST SAMPLE IN SEGMENT
C     2. LINE COORDINATE OF FIRST SAMPLE IN SEGMENT
C     3. SLOPE OF INPUT PIXEL VS. OUTPUT PIXEL EQUATION
C     4. SLOPE OF INPUT  LINE VS. OUTPUT PIXEL EQUATION
C     5. NUMBER OF OUTPUT PIXELS IN THE CELL
C     6. OUTPUT PIXEL NUMBER.
      COMMON /PIXDAT/ NLINPX, NPIXLN, NHGL, NVGL, SRATE, NLPI, NSPL,
     *                KDUM, INTERP, NB
      COMMON / PIXIN/ NBSAM, MAXLN, MNDEX, SAMPO, LINEO
      COMMON / TRANS/ UTMPIX(6)
      EQUIVALENCE (NPIX, UTMPIX(5)), (ISAMP, UTMPIX(6))
C
      LBASE=LINEO-1
      PSI = UTMPIX(1) - SAMPO + 1
      PLI = UTMPIX(2) - LINEO + 1
      SPACE1 = UTMPIX(3)*SRATE
      SPACE2 = UTMPIX(4)*SRATE
C
C         LOOP OVER OUTPUT PIXELS
          DO 1050 I=1,NPIX
          ISAMP = ISAMP + 1
          DE=I-1
C
C         COMPUTE INPUT LINE AND SAMPLE NUMBERS
          PS=PSI+DE*SPACE1
          PL=PLI+DE*SPACE2
          GO TO (1000, 2000, 3000), INTERP
C
C     * * * * * NEAREST NEIGHBOR INTERPOLATION * * * * *
C
 1000     IPL = PL + 0.5
          IF (IPL.LT.1.OR.IPL.GT.MAXLN) GO TO 1030
C
C           ADD EARTH ROTATION AND SENSOR DELAY OFFSETS
            IPS = PS + DELAY(IPL+LBASE) + 0.5
            IF (IPS.LT.1.OR.IPS.GT.NBSAM) GO TO 1030
            SLR = MNDEX(IPL)
C
          DO 1021 IBAND=1,NB
 1021     LPXOUT(IBAND,ISAMP) = LPIXEL(IBAND,IPS,SLR)
          GO TO 1050
C
C     * * * * * BILINEAR INTERPOLATION * * * * *
C
 2000     IPL = PL
```

[redacted]

```
          IF (IPL.LT.1.OR.IPL.GT.MAXLN-1) GO TO 1030
          DP = PL - IPL
            DO 2010 LINE=1,2
            PSCCT = PS + DELAY(IPL+LBASE)
            IPS = PSCCT
            IF (IPS.LT.1.OR.IPS.GT.NBSAM-1) GO TO 1030
            D = PSCCT - IPS
            SLR = MNDEX(IPL)
              DO 2005 IBAND=1,NB
              K1 = LPIXEL(IBAND,IPS  ,SLR)
              K2 = LPIXEL(IBAND,IPS+1,SLR)
 2005         RR(LINE,IBAND) = K1 + D*(K2-K1)
            IPL = IPL + 1
 2010       CONTINUE
          DO 2021 IBAND=1,NB
          IX1 = RR(1,IBAND) + DP*(RR(2,IBAND)-RR(1,IBAND)) + 0.5
 2021     LPXOUT(IBAND,ISAMP) = IX1
          GO TO 1050
C
C      * * * * * BICUBIC INTERPOLATION * * * * *
C
 3000     IPL = PL
          IF (IPL.LT.2.OR.IPL.GT.MAXLN-2) GO TO 1030
          DP = PL - IPL
          IPL = IPL - 1
C
C           COMPUTE AN INTERPOLATED SAMPLE IN EACH OF 4 LINES
            DO 3010 LINE=1,4
            PSCCT = PS + DELAY(IPL+LBASE)
            IPS = PSCCT
            IF (IPS.LT.2.OR.IPS.GT.NBSAM-2) GO TO 1030
            D = PSCCT - IPS
            SLR = MNDEX(IPL)
              DO 3005 IBAND=1,NB
              K1 = LPIXEL(IBAND,IPS-1,SLR)
              K2 = LPIXEL(IBAND,IPS  ,SLR)
              K3 = LPIXEL(IBAND,IPS+1,SLR)
              K4 = LPIXEL(IBAND,IPS+2,SLR)
              K21 = K2 - K1
              K43 = K4 - K3
 3005         RR(LINE,IBAND) = D* (D* (D* (K43+K21) - (K43+2*K21))
     *                          + (K3-K1)) + K2
            IPL = IPL + 1
 3010       CONTINUE
C
C         INTERPOLATE OVER 4 LINES TO GET FINAL OUTPUT SAMPLE
          DO 3021 IBAND=1,NB
          R21 = RR(2,IBAND) - RR(1,IBAND)
          R43 = RR(4,IBAND) - RR(3,IBAND)
          IX1 = DP* (DP* (DP* (R43+R21) - (R43+2.0*R21)) + (RR(3,IBAND)-
     *          RR(1,IBAND))) + RR(2,IBAND) + 0.5
          IF (IX1.LT.0) IX1=0
          IF (IX1.GT.255) IX1=255
 3021     LPXOUT(IBAND,ISAMP) = IX1
          GO TO 1050
```

```
C
 1030 DO 1035 IBAND=1,NB
 1035 LPXOUT(IBAND,ISAMP) = KDUM
C
 1050      CONTINUE
      RETURN
      END
      SUBROUTINE EVPOLY (IFUN, X, Y, ANS)
C
C     EVALUATE POLYNOMIAL FIT FUNCTIONS
C
      DOUBLE PRECISION COEF(21,4), ANSD, XD, YD, XXX, YYY
      INTEGER NTERM(5)/3,6,10,15,21/
      INTEGER XP(21)/0,1,0,2,1,0,3,2,1,0,4,3,2,1,0,5,4,3,2,1,0/
      INTEGER YP(21)/0,0,1,0,1,2,0,1,2,3,0,1,2,3,4,0,1,2,3,4,5/
      COMMON /LSQCFC/ COEF, LSQDEG, IER1(2), TOL
      COMMON / MEANS/ GCPM(4)
C
C     COEF(21,I)= COEFFICIENTS FOR : 1= PIXEL
C                                    2= LINE
C                                    3= EASTING
C                                    4= NORTHING
      ISTOP = NTERM(LSQDEG)
      ANSD = 0.0
      IF (IFUN.EQ.3.OR.IFUN.EQ.4) GO TO 1
      XD = X - GCPM(3)
      YD = Y - GCPM(4)
      GO TO 5
    1 XD = X - GCPM(1)
      YD = Y - GCPM(2)
    5 CONTINUE
C
      DO 10 I=1,ISTOP
      XXX = 1.0
      IF (XP(I).NE.0) XXX = XD**XP(I)
      YYY = 1.0
      IF (YP(I).NE.0) YYY = YD**YP(I)
   10 ANSD = ANSD + COEF(I,IFUN)*XXX*YYY
C
      ANS = ANSD + GCPM(IFUN)
      RETURN
      END
      REAL FUNCTION MVPOFF (PS)
C
C     COMPUTE MIRROR VELOCITY PROFILE OFFSET
C
C     LLC = NUMBER OF PIXELS IN THE RAW SCAN LINE
C     SAMPOF = NUMBER OF PIXELS SKIPPED IN THE LANDSAT SCENE
C     AMPL, PHASE = AMPLITUDE, PHASE OF MIRROR VELOCITY PROFILE CURVE
C
      LOGICAL CCT
      COMMON /LANDST/ CCT, LLC, DEGCEN, SAMPOF, LINOFF, AMPL, PHASE
C
      IF (CCT) GO TO 12
      MVPOFF = 0.0
      RETURN
C
   12 CONTINUE
      PS1 = PS + SAMPOF
      MVPOFF = AMPL * SIN (6.2831853 * (PS1+PHASE-1.0) / (LLC-1))
      RETURN
      END
      REAL FUNCTION MVPINV (PS)
C
C     COMPUTE NEGATIVE MIRROR VELOCITY PROFILE CORRECTION
C
C     LLC = NUMBER OF PIXELS IN THE RAW SCAN LINE
C     SAMPOF = NUMBER OF PIXELS SKIPPED IN THE LANDSAT SCENE
C     AMPL, PHASE = AMPLITUDE, PHASE OF MIRROR VELOCITY PROFILE CURVE
C
      LOGICAL CCT
      COMMON /LANDST/ CCT, LLC, DEGCEN, SAMPOF, LINOFF, AMPL, PHASE
C
      IF (CCT) GO TO 10
      MVPINV = 0.0
      RETURN
C
   10 CONTINUE
      PS1 = PS + SAMPOF
      MVPINV = AMPL * SIN (6.2831853 * (PS1+PHASE-1.0) / (LLC-1))
      RETURN
      END
      FUNCTION ERCURV (PS)
C
C     COMPUTE EARTH CURVATURE CORRECTION
C
C     RE, RSAT = EARTH RADIUS, SATELLITE ORBIT RADIUS
C     TFOV = TOTAL FIELD OF VIEW OF THE SCANNER (11.56 DEGREES)
C     LLC  = NUMBER OF PIXELS IN THE RAW SCAN LINE
C     SAMPOF = NUMBER OF PIXELS SKIPPED IN THE LANDSAT SCENE
C
      LOGICAL CCT
      COMMON /LANDSY/ CCT, LLC, DEGCEN, SAMPOF, LINOFF, AMPL, PHASE
      DATA RE, RSAT, TFOV /6367.4, 7285.6, 0.20176/
C
      IF (CCT) GO TO 10
      ERCURV = 0.0
      RETURN
C
C     COMPUTE SCANNER ANGLE AT PIXEL NO. PS AND ANGLE SUBTENDED AT
C     THE CENTER OF THE EARTH
   10 CONTINUE
      TFOV2 = TFOV/2.0
      PS1 = PS + SAMPOF
      ANGSCN = (PS1-1.0)*TFOV/(LLC-1) - TFOV2
      ANGERT = ARSIN(SIN(ANGSCN)*RSAT/RE) - ANGSCN
      TOTERT = ARSIN (SIN(TFOV2)*RSAT/RE) - TFOV2
C
C     FIND SCANNER ANGLE BASED ON FRACTION OF TOTAL ARC ON THE EARTH'S
C     SURFACE AND CONVERT TO PIXEL NUMBER
      ANGSC1 = TFOV2 * ANGERT / TOTERT
      PS2 = 1.0 + (ANGSC1+TFOV2) * (LLC-1) / TFOV
      ERCURV = PS1 - PS2
      RETURN
      END
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```
      FUNCTION ERCURI (PS)
C
C     COMPUTE EARTH CURVATURE DE-CORRECTION
C
C     RE, RSAT - EARTH RADIUS, SATELLITE ORBIT RADIUS
C     TFOV - TOTAL FIELD OF VIEW OF THE SCANNER (11,56 DEGREES)
C     LLC  - NUMBER OF PIXELS IN THE RAW SCAN LINE
C     SAMPOF - NUMBER OF PIXELS SKIPPED IN THE LANDSAT SCENE
C
      LOGICAL CCT
      COMMON /LANDST/ CCT, LLC, DEGCEN, SAMPOF, LINCFF, AMPL, PHASE
      DATA RE, RSAT, TFOV /6367,4, 7285,6, 0,20176/
C
      IF (CCT) GO TO 10
      ERCURI = 0,0
      RETURN
C
C     EARTH ANGLE IS PROPORTIONAL TO CORRECTED PIXEL NUMBER,
   10 CONTINUE
      TFOV2 = TFOV/2,0
      CENTER = (LLC+1)/2,0
      PS1 = PS + SAMPOF
      TOTERT = ARSIN (SIN(TFOV2)*RSAT/RE) - TFOV2
      ANGERT = -TOTERT * (CENTER-PS1) / (CENTER-1,0)
C
C     COMPUTE ORIGINAL SCAN ANGLE BASED ON EARTH ANGLE,
C     C IS LINE FROM SATELLITE TO PIXEL LOCATION ON THE EARTH'S SURFACE,
      C = (RSAT**2 + RE**2 - 2,0*RSAT*RE*COS(ANGERT)) ** 0,5
      ANGSCN = ARSIN(RE/C*SIN(ANGERT))
      PS2 = 1,0 + (ANGSCN+TFOV2) * (LLC-1) / TFOV
      ERCURI = PS2 - PS1
      RETURN
      END
      SUBROUTINE READAR (NTAPE1, W, NSAMP)
C
C   READ NSAMP BYTES INTO ARRAY W FROM LOGICAL UNIT NTAPE1
C
      LOGICAL*1 W(NSAMP)
C
      READ (NTAPE1) W
      RETURN
C
C     ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
C     ****************************************************************
C
      ENTRY WRITAR (NTAPE1, W, NSAMP)
C
C   WRITE NSAMP BYTES FROM ARRAY W ONTO LOGICAL UNIT NTAPE1
C
      WRITE (NTAPE1) W
      RETURN
      END
```

MAP OVERLAY

A set of routines was developed to allow the registration of a ground truth map to Landsat CCT data. The subroutines are analagous to those previously described for registration, and so redundant descriptions will not be given. Differences in input and method are the following:

- o the CCT coordinates are to be read into the third and fourth columns of the GCP array, and the map coordinates into the first and second,
- o the output image which is divided into cells must be the offset image (Earth rotation and sensor delay removed), since this is the image which is used in the transformation equations,
- o since each CCT line has a different shift from the offset line, the number of pixels and the starting pixel coordinate in a cell must be computed for each output line, rather than kept in a table,
- o since map class numbers can not be interpolated, only nearest neighbor resampling (subroutine NN2) is used.

The routines which are modified and their storage requirements are given in the following table:

SUBROUTINE	STORAGE (BYTES)
GCPFT2	2884
GCPCR2	870
GRDMP2	2600
CELMP2	2496
RECTF2	3716
NN2	702

Listings of these subroutines follow.

```
      SUBROUTINE GCPFT2 (GCP, MGCP, MAXDEG)
C
C     TO FIND GEOMETRIC TRANSFORMATION NEEDED FOR GEOGRAPHIC REFERENCING
C
      DIMENSION GCP(4,MGCP)
      DOUBLE PRECISION COEF(21,4)
      LOGICAL CCT
      INTEGER NTERM(5)/3,6,10,15,21/
      COMMON /LANDST/ CCT, LLC, DEGCEN, SAMPOF, LINOFF, AMPL, PHASE
      COMMON /LSQCFC/ COEF, LSQDEG, IER1(2), TOL
      COMMON / MEANS/ GCPM(4)
C
C     GCP IS GROUND CONTROL POINT TABLE INPUT BY USER
C     1. PIXEL IN MAP DATA
C     2. LINE IN MAP DATA
C     3. PIXEL IN LANDSAT DATA
C     4. LINE IN LANDSAT DATA
C
      IF (CCT) CALL GCPCR2 (GCP, MGCP)
      AMGCP=MGCP
      RADDEG = 180.0/3.14159265
      TOL=1.E-20
      DO 1 I=1,4
    1 GCPM(I) = 0.0
      WRITE (6,101)
      WRITE (6,1020)
C
C     COMPUTE SUMS OF INPUT DATA
      DO 10 J=1,MGCP
      DO 5 I=1,4
    5 GCPM(I) = GCPM(I) + GCP(I,J)
      WRITE (6,102) J, (GCP(I,J), I=1,4)
   10 CONTINUE
C
C     COMPUTE MEANS OF INPUT DATA
      DO 11 I=1,4
   11 GCPM(I) = GCPM(I) / AMGCP
      WRITE (6,104) GCPM
C
C     SUBTRACT MEANS OF INPUT DATA
      WRITE (6,1019)
      WRITE (6,1020)
      DO 13 J=1,MGCP
      DO 12 I=1,4
   12 GCP(I,J) = GCP(I,J) - GCPM(I)
      WRITE (6,102) J, (GCP(I,J), I=1,4)
   13 CONTINUE
C
C     FIND POLYNOMIAL FITS FOR DEGREES 1 - INPUT VALUE OF LSQDEG
      DO 100 LSQDEG=1,MAXDEG
      ISTOP=NTERM(LSQDEG)
      IF (ISTOP.GT.MGCP) GO TO 400
      CALL LSQCF (0, GCP, MGCP)
      WRITE(6,383) LSQDEG,IER1,((COEF(J,I),I=1,4),J=1,ISTOP)
      WRITE(6,111)
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```
      RESPX=0.0
      RESUT=0.0
C
C     PERFORM ACCURACY ANALYSIS OF FIT
      DO 500 I=1,NGCP
      GP=GCP(1,I) + GCPM(1)
      GL=GCP(2,I) + GCPM(2)
      GE=GCP(3,I) + GCPM(3)
      GN=GCP(4,I) + GCPM(4)
C
      CALL EVPOLY(1,GE,GN,ANS)
      DP=GP-ANS
      CALL EVPOLY(2,GE,GN,ANS)
      DL=GL-ANS
      SQ = DP**2 + DL**2
      RESPX = RESPX + SQ
      XMAG = SQRT (SQ)
      XDIR = RADDEG * ATAN2(-DL,DP)
C
      CALL EVPOLY(3,GP,GL,ANS)
      DE=GE-ANS
      CALL EVPOLY(4,GP,GL,ANS)
      DN=GN-ANS
      SQ = DE**2 + DN**2
      RESUT = RESUT + SQ
      UMAG = SQRT (SQ)
      UDIR = RADDEG * ATAN2(DE,DN)
      WRITE (6,109) I, XMAG, XDIR, DP, DL, UMAG, UDIR, DE, DN
500   CONTINUE
C
      AGCP1=NGCP-1
      RESPX = SQRT (RESPX/AGCP1)
      RESUT = SQRT (RESUT/AGCP1)
      WRITE (6,390) RESPX, RESUT
  100 CONTINUE
      LSQDEG = MAXDEG
      RETURN
C
  900 WRITE (6,1100) LSQDEG, ISTOP, NGCP
      LSQDEG = LSQDEG - 1
      RETURN
C
C     FORMAT STATEMENTS.
  101 FORMAT ('1',20X,'GROUND CONTROL POINTS'/)
  102 FORMAT (1X,I4,4F15.3)
  104 FORMAT (///20X,'MEANS OF INPUT DATA ARE'//5X,4F15.3/
     .        '0THESE MEANS ARE FIRST SUBTRACTED')
  109 FORMAT (1X,I5,2(F10.3,F10.1,8X,2F10.3,8X))
  111 FORMAT (/35X,'COMPARISON OF OBSERVED AND PREDICTED VALUES'/
     .25X,'GEOGRAPHIC',45X,'LANDSAT'/15X,'ERROR',20X,'OBS - PRED',20X,
     .'ERROR',20X,'OBS - PRED'/5X,'MAGNITUDE  DIRECTION',10X,'P ERROR
     .L ERROR',10X,'MAGNITUDE  DIRECTION',10X,'E ERROR   N ERROR'/)
  595 FORMAT ('1  LEAST SQUARES FIT OF DEGREE',I5,25X,'ERROR CODES =',
     .2I5/40X,'COEFFICIENTS'/10X,'EAST PIXEL',10X,'NORTH LINE',10X,
     .'PIXEL',15X,'LINE'/(1X,1P4E20.6))
```

```
 390 FORMAT (/2(15X,'RMS ERROR =',E13.6,15X))
1019 FORMAT ('1',7X'CONTROL POINTS AFTER SUBTRACTING MEANS'/)
1020 FORMAT (18X,'GEOGRAPHIC',22X,'LANDSAT'/18X,'COORDINATES',18X,
    ,'COORDINATES'/)
1100 FORMAT (//20X,'FIT OF DEGREE',I2,' REQUIRES',I3,' GROUND CONTROL PO
    ,INTS,',I5,' WERE SUPPLIED,'/)
     END
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
      SUBROUTINE GCPCR2 (GCP, MGCP)
C
C     GCPCR2 REMOVES MIRROR VELOCITY PROFILE AND EARTH CURVATURE
C     DISTORTIONS PRIOR TO OBTAINING LEAST SQUARES FITS
C
C     GCP IS GROUND CONTROL POINT TABLE INPUT BY USER
C
      DIMENSION GCP(4,MGCP)
      REAL*4 MVPOFF
      COMMON /LANDST/ CCT, LLC, DEGCEN, SAMPOF, LINOFF, AMPL, PHASE
C
C     ADD BACK DELAY TO GET ORIGINAL CCT PIXEL NUMBER
      WRITE (6,100)
      DO 10 NGCP = 1,MGCP
      WRITE (6,101) GCP(4,NGCP), GCP(3,NGCP)
      LINE = GCP(4,NGCP) + 0.5
      SAMP = GCP(3,NGCP) + DELAY(LINE)
C
C     APPLY MIRROR VELOCITY PROFILE CORRECTION
      OFF = MVPOFF(SAMP)
      GCP(3,NGCP) = GCP(3,NGCP) - OFF
C
C     APPLY EARTH CURVATURE CORRECTION
      CURV = ERCURV(SAMP-OFF)
      GCP(3,NGCP) = GCP(3,NGCP) - CURV
C
      SAMPO = SAMP + SAMPOF
      SHIFT = OFF + CURV
      WRITE (6,102) SAMP, SAMPO, OFF, CURV, SHIFT, GCP(3,NGCP)
   10 CONTINUE
      RETURN
C
  100 FORMAT ('1',20X,'GROUND CONTROL POINT CORRECTIONS'//19X,'OFFSET',
     .5X,'ORIGINAL',3X,'CCT SCENE',7X,'MVP',9X,'CURV',8X,'NET',8X,
     .'CORR.'/7X,'RECORD',3(6X,'SAMPLE'),3(7X,'SHIFT'),6X,'OFFSET'/)
  101 FORMAT (1X,2F12.3)
  102 FORMAT ('+',24X,8F12.3)
      END
      SUBROUTINE GRDMP2
C
C     GRDMP2 COMPUTES THE INTERPOLATION GRID INTERSECTIONS AND THE
C     INTERPOLATION ERRORS IN INPUT IMAGE COORDINATES
C     THIS ROUTINE IS FOR TRANSFORMATION FROM A MAP (E & N COORDINATES)
C     TO AN IMAGE (P & L COORDINATES).
C     GRID IS COORDINATES (UTM & PIXEL) OF HORIZONTAL & VERTICAL GRID
C     INTERSECTIONS
C
      DIMENSION GRIDPL(2,30), GRIDEN(2,30,30), NUMPXC(30), CPXVAL(30),
     .NUMLNC(30), ERRCEL(29)
      LOGICAL*1 CELLCG(29,29)
      REAL LINE, LCEN, MVPOFF, MVP
      COMMON /PIXDAT/ NLINPX, NPIXLN, NHGL, NVGL, SRATE, NLPI, NSPL,
     .                KDUM, INTERP, NB
      COMMON /MAPDAT/ STRTL, STOPL, STRTP, STOPP
      COMMON /  GRID/ GRIDPL, GRIDEN, NUMPXC, CPXVAL, NUMLNC, CELLCG
C
C     COMPUTE NUMBER OF OUTPUT LINES AND SAMPLES
      NLPI = STOPL - STRTL + 1.5
      NSPL = STOPP - STRTP + 1.5
C
C     FIND MAXIMUM AND MINIMUM OFFSET COORDINATES
      DLYMIN = 10000.0
      DLYMAX = 0.0
      DO 55 NR=1,6
      DLYMIN = AMIN1 (DLYMIN, DELAY(NR))
   55 DLYMAX = AMAX1 (DLYMAX, DELAY(INT(STOPL)+1-NR))
      STRTP = STRTP - DLYMAX
      STOPP = STOPP - DLYMIN
C
C     COMPUTE GX AND GY GRID SPACING
      NHGL1=NHGL-1
      NVGL1=NVGL-1
      GX = (STOPP-STRTP) / NVGL1
      GY = (STOPL-STRTL) / NHGL1
C
C     CALCULATE GRID INTERSECTIONS BY STEPPING THROUGH IMAGE GRID AND
C     TRANSFORMING TO UTM MAP OR GEOGRAPHIC COORDINATES
        DO 1010 I=1,NVGL
        GRIDPL(1,I) = STRTP + (I-1)*GX
C
          DO 1000 J=1,NHGL
          GRIDPL(2,J) = STRTL + (J-1)*GY
C
C         APPLY MIRROR VELOCITY AND EARTH CURVATURE CORRECTIONS
C         TO GET PIXEL GROUND LOCATION
          LINE = GRIDPL(2,J)
          PIXCCT = GRIDPL(1,I) + DELAY(INT(LINE+0.5))
          MVP = MVPOFF(PIXCCT)
          CRV = ERCURV(PIXCCT-MVP)
          PIX = GRIDPL(1,I) - MVP - CRV
C
          CALL EVPOLY (1, PIX, LINE, PE)
          CALL EVPOLY (2, PIX, LINE, PN)
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
C
          GRIDEN(1,J,I) = PE
          GRIDEN(2,J,I) = PN
 1000     CONTINUE
 1010   CONTINUE
C
      WRITE (6,1050)
      DO 1011 J=1,NHGL
 1011 WRITE (6,1051) (GRIDEN(1,J,K), GRIDEN(2,J,K), GRIDPL(1,K),
     .GRIDPL(2,J), K=1,NVGL)
C
C     COMPUTE AND PRINT FINAL GRID ERRORS
      ERRMAX = 0.0
      WRITE (6,1065)
        DO 1030 I4=2,NHGL
        I41=I4-1
          DO 1020 J4=2,NVGL
          J41=J4-1
C
C         COMPUTE MAP COORDINATES BY BILINEAR INTERPOLATION IN A CELL
          PEB = (GRIDEN(1,I4,J4)+GRIDEN(1,I41,J4)+GRIDEN(1,I41,J41)
     .    +GRIDEN(1,I4,J41)) / 4.0
          PNB = (GRIDEN(2,I4,J4)+GRIDEN(2,I41,J4)+GRIDEN(2,I41,J41)
     .    +GRIDEN(2,I4,J41)) / 4.0
C
C         COMPUTE CENTER PIXEL AND LINE
          PCEN = (GRIDPL(1,J4)+GRIDPL(1,J41)) / 2.0
          LCEN = (GRIDPL(2,I4)+GRIDPL(2,I41)) / 2.0
C
C         APPLY MIRROR VELOCITY AND EARTH CURVATURE CORRECTIONS
C         TO GET PIXEL GROUND LOCATION
          PCNCCT = PCEN + DELAY(INT(LCEN+0.5))
          MVP = MVPOFF(PCNCCT)
          CRV = ERCURV(PCNCCT-MVP)
          PCEN = PCEN - MVP - CRV
C
C         COMPUTE MAP COORDINATES BY FUNCTION
          CALL EVPOLY (1, PCEN, LCEN, PE)
          CALL EVPOLY (2, PCEN, LCEN, PN)
C
C         COMPUTE ERROR
          ERR=SQRT((PEB-PE)**2+(PNB-PN)**2)
          ERRCEL(J41) = ERR
          ERRMAX = AMAX1 (ERRMAX, ERR)
 1020     CONTINUE
        WRITE (6,1070) (ERRCEL(J), J=1,NVGL1)
 1030   CONTINUE
      WRITE (6,1060) ERRMAX
C
C     COMPUTE TABLE OF FIRST OUTPUT LINE NUMBER IN EACH CELL
      NUMLNC(1) = 1
      DO 1039 J=2,NHGL1
 1039 NUMLNC(J) = GRIDPL(2,J) - GRIDPL(2,1) + 2.0
      NUMLNC(NVGL) = NLPI + 1
      RETURN
```

```
C
 1050 FORMAT ('1',30X,'INTERPOLATION GRID INTERSECTIONS'/31X,32('*')//
     ,10X,'EAST PIXEL',10X,'NORTH LINE',14X,'PIXEL',16X,'LINE')
 1051 FORMAT (/(4F20.3))
 1060 FORMAT (//21X,'MAXIMUM GRID ERROR =',1PE10.2,'  PIXELS')
 1065 FORMAT ('1',20X,'MAGNITUDE OF INTERPOLATION ERROR AT GRID CELL CEN
     ,TERS'/21X,53('*'))
 1070 FORMAT (/(1P13E10.2))
      END
      SUBROUTINE CELMP2 (CORE)
C
C     *COMPUTE DIMENSIONS OF INPUT ARRAY WHICH CAN BE HELD IN CORE AT A
C      TIME AND NUMBER OF PARTITIONS REQUIRED.
C     PIX AND ROW VARIABLES HAVE DIRECTIONS APPENDED, E.G. PIXNW IS
C     NORTHWEST CORNER PIXEL OF A CELL
C
      INTEGER ROWNW, PIXNW, ROWNE, PIXNE, ROWSE, PIXSE, ROWSW, PIXSW,
     .RN, RS, SW, SE, R1, R2, S1, S2, CORE, RSTRT
      DIMENSION GRIDPL(2,30), GRIDEN(2,30,30), NUMPXC(30), CPXVAL(30),
     .NUMLNC(30)
      LOGICAL*1 CELLCG(29,29)
      COMMON /PIXDAT/ NLINPX, NPIXLN, NHGL, NVGL, SRATE, NLPI, NSPL,
     .                KDUM, INTRP, NB
      COMMON /  GRID/ GRIDPL, GRIDEN, NUMPXC, CPXVAL, NUMLNC, CELLCG
      COMMON /SEGMNT/ NCGP, NBSAM(10), NBLIN(10), NSTRT(10),
     .                RSTRT(29,10)
C
C     INITIALIZE CELL COLUMN GROUP INDICATOR (CELLCG)
      NHGL1 = NHGL - 1
      NVGL1 = NVGL - 1
      DO 1100 I=1,NHGL1
      DO 1100 J=1,NVGL1
 1100 CELLCG(I,J) = 255
C
C     FIND NCGP, THE NUMBER OF COLUMN GROUPS,
C     S1 IS FIRST INPUT PIXEL FOR THIS COLUMN GROUP
      NCGP = 0
      S1 = 1
C
C     START A NEW COLUMN GROUP OR SEGMENT
 1800 CONTINUE
      NCGP = NCGP + 1
      MINNC = 1000000
C
C     LOOP OVER CELL ROWS
      DO 2100 I=1,NHGL1
      R1 = 1000000
      R2 = 0
      S2 = 0
C
C         LOOP OVER CELLS FOR THIS SEGMENT
          DO 2000 J=1,NVGL1
          IF (CELLCG(I,J).NE.255) GO TO 2000
C
C         FIND RANGE OF IMAGE COORDINATES REQUIRED FOR THE OUTPUT CELL
          ROWNW = GRIDEN(2,I,J) + 0.5
          PIXNW = GRIDEN(1,I,J) + 0.5
          ROWNE = GRIDEN(2,I,J+1) + 0.5
          PIXNE = GRIDEN(1,I,J+1) + 0.5
          ROWSE = GRIDEN(2,I+1,J+1) + 0.5
          PIXSE = GRIDEN(1,I+1,J+1) + 0.5
          ROWSW = GRIDEN(2,I+1,J) + 0.5
          PIXSW = GRIDEN(1,I+1,J) + 0.5
          RN = MAX0 (MIN0(ROWNW,ROWNE), 1)
```

```
                RS = MINO (MAXO(ROWSW,ROWSE), NLINPX)
                SW = MAXO (MINO(PIXNW,PIXSW), 1)
                SE = MINO (MAXO(PIXNE,PIXSE), NPIXLN)
C
C               CHECK FOR IMAGE COORDINATES OUTSIDE THE INPUT IMAGE
                IF (RN.GT.NLINPX.OR.RS.LT.1.OR.SW.GT.NPIXLN.OR.SE.LT.1)
     *          GO TO 1920
C
C                 FIND RANGE OF IMAGE COORDINATES FOR THIS ROW OF CELLS AND
C                 FIND NO. OF OUTPUT CELLS THAT CAN BE FILLED GIVEN CORE LIMIT
                  R1 = MINO (R1, RN)
                  R2 = MAXO (R2, RS)
                  S2 = MAXO (S2, SE)
                  NR = R2 - R1 + 1
                  NC = S2 - S1 + 1
                  IF (NR*NC.LE.CORE) GO TO 1950
C
C                 CORE REQUIRED EXCEEDS THAT AVAILABLE
C                 GO TO NEXT ROW OF CELLS
                  MINNC = MINO (NCSAVE, MINNC)
                  CELLCG(I,J) = 222
                  GO TO 2100
C
 1920           CELLCG(I,J) = 0
                GO TO 2000
C
C                 CORE IS AVAILABLE.  SAVE REQUIRED DIMENSIONS.
 1950             NCSAVE = NC
 2000           CONTINUE
 2100 CONTINUE
C
      LASTPX = S1 + MINNC - 1
      IF (LASTPX.LT.NPIXLN) GO TO 2110
      LASTPX = NPIXLN
      MINNC = NPIXLN - S1 + 1
 2110 CONTINUE
C
C     NOW USE NUMBER OF INPUT COLUMNS AVAILABLE TO SET CELL SEGMENT
C     INDICATORS AND TO FIND ROW AND COLUMN INFORMATION
      MINS1 = 1000000
C
C     LOOP OVER CELL ROWS
      DO 2150 I=1,NHGL1
      IC1 = 100
      IC2 = 0
      RSTRT(I,NCGP) = 0
C
C         LOOP OVER CELLS
          DO 2130 J=1,NVGL1
C
C         CHECK WHETHER THE CELL REQUIRES INPUT OUTSIDE OF THE IMAGE
          IF (CELLCG(I,J).EQ.0) GO TO 2126
C
C         JUMP OUT IF THE CELL REQUIRES TOO MUCH CORE.
          IF (CELLCG(I,J).EQ.222) GO TO 2140
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```
C
C           IF THE CELL USES INPUT DATA, CHECK FOR THE LAST PIXEL
            IF (CELLCG(I,J).EQ.255) GO TO 2125
            GO TO 2130
C
C           JUMP OUT IF THE CELL REQUIRES SAMPLES PAST LASTPX.
 2125       PIXNE = GRIDEN(1,I,J+1) + 0.5
            PIXSE = GRIDEN(1,I+1,J+1) + 0.5
            SE = MIN0 (MAX0(PIXNE,PIXSE), NPIXLN)
            IF (SE.GT.LASTPX) GO TO 2140
C
C           FIND FIRST AND LAST CELLS THAT USE INPUT DATA
            IC1 = MIN0 (J, IC1)
            IC2 = MAX0 (J, IC2)
C
C           ELSE SET COLUMN GROUP INDICATOR TO GROUP NUMBER.
            CELLCG(I,J) = NCGP
            GO TO 2130
C
 2126       CELLCG(I,J) = NCGP + 100
 2130       CONTINUE
C
C           FIND FIRST INPUT ROW FOR THIS ROW OF CELLS
 2140       IF (IC2.EQ.0) GO TO 2150
            ROWNW = GRIDEN(2,I,IC1) + 0.5
            ROWNE = GRIDEN(2,I,IC2+1) + 0.5
            RN = MAX0 (MIN0(ROWNW,ROWNE), 1)
            RSTRT(I,NCGP) = RN
C
C           FIND STARTING INPUT COLUMN FOR NEXT SEGMENT
            IF (J.EQ.NVGL) GO TO 2150
            NEXTCL = IC2 + 1
            PIXNW = GRIDEN(1,I,NEXTCL) + 0.5
            PIXSW = GRIDEN(1,I+1,NEXTCL) + 0.5
            SW = MAX0 (MIN0(PIXNW,PIXSW), 1)
            MINS1 = MIN0 (SW, MINS1)
C
C           SET INDICATOR FOR STARTING CELL IN NEXT SEGMENT
            DO 2145 J=NEXTCL,NVGL1
 2145       CELLCG(I,J) = 255
 2150 CONTINUE
C
C     SET PARAMETERS OF INPUT DATA SEGMENT
      MAXNR = MIN0 (CORE/MINNC, NLINPX)
      NBSAM(NCGP) = MINNC
      NBLIN(NCGP) = MAXNR
      NSTRT(NCGP) = S1
      S1 = MINS1
C
C     IF ALL CELLS NOT DONE, START A NEW SEGMENT
      IF (LASTPX.LT.NPIXLN) GO TO 1800
C
      WRITE (6,100)
      DO 2160 I=1,NHGL1
 2160 WRITE (6,101) (CELLCG(I,J), J=1,NVGL1)
```

```
      RETURN
C
  100 FORMAT (///5X,'COLUMN GROUP NUMBERS FOR THE GRID CELLS'/5X,
     ,39('*')/)
  101 FORMAT (5X,29I2)
      END
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
      SUBROUTINE RECTF2 (INPIX, PIXEL, PIXOUT)
C
C     HANDLE READING AND WRITING OF DATA RECORDS
C     CALL NEAREST NEIGHBOR RESAMPLING ROUTINE
C     EACH OUTPUT LINE IS SHIFTED TO THE OFFSET IMAGE COORDINATES
C     TO MATCH THE INTERPOLATION GRID.
C
      LOGICAL*1 INPIX(1), PIXEL(1), PIXOUT(NSPL), CELLCG(29,29)
      DIMENSION GRDOUT(2,30), GRIDIN(2,30,30), NUMPXC(30), CPXVAL(30),
     .NUMLNC(30)
      INTEGER*2 MNDEX(1000)
      INTEGER SAMPSG(29,10), RSTRT, PSTRT, SAMPO
      COMMON /PIXDAT/ NLINPX, NPIXLN, NHGL, NVGL, SRATE, NLPI, NSPL,
     .                KDUM, INTERP, NB
      COMMON /  GRID/ GRDOUT, GRIDIN, NUMPXC, CPXVAL, NUMLNC, CELLCG
      COMMON /SEGMNT/ NCGP, NBSAM(10), NBLIN(10), PSTRT(10),
     .                RSTRT(29,10)
      COMMON / PIXIN/ MAXSP, MAXLN, MNDEX, SAMPO, LINEO
      COMMON / TRANS/ TRANIO(6)
      EQUIVALENCE (NPIX,TRANIO(5)), (ISAMP,TRANIO(6))
C
      NHGL1 = NHGL - 1
      NVGL1 = NVGL - 1
C
C     LOOP OVER COLUMN GROUPS
      DO 1020 ICGP=1,NCGP
      MAXSP = NBSAM(ICGP)
      MAXLN = NBLIN(ICGP)
      SAMPO = PSTRT(ICGP)
      LUNIT = ICGP + 20
      REWIND LUNIT
      CALL LOADLN (.TRUE., 1, MAXLN, INPIX, PIXEL)
C
C       LOOP OVER OUTPUT IMAGE CELLS
      DO 1010 IGY1=1,NHGL1
      IGY2 = IGY1 + 1
      LSTRT = NUMLNC(IGY1)
      LSTOP = NUMLNC(IGY2) - 1
C
C     LOAD LINES OF INPUT NECESSARY FOR RESAMPLED LINE
      LMIN = RSTRT(IGY1,ICGP)
      IF (LMIN.EQ.0) GO TO 150
      LMAX = MIN0 (LMIN+MAXLN-1, NLINPX)
      CALL LOADLN (.FALSE., LMIN, LMAX, INPIX, PIXEL)
C
C         LOOP OVER OUTPUT IMAGE LINES IN THE CELL
  150 DO 1000 ILINE=LSTRT,LSTOP
      OUTLIN = GRDOUT(2,1) + ILINE - 1
C
C         COMPUTE OFFSET COORDINATE OF FIRST CCT PIXEL IN THE GRID
          OFF1 = 1.0 - DELAY(ILINE)
C
C         FIND STARTING SAMPLE NUMBERS AND COORDINATES FOR EACH CELL
          NUMPXC(1) = 1
          CPXVAL(1) = OFF1
```

```
            DO 200 J=2,NVGL1
            NUMPXC(J) = GRDOUT(1,J) - OFF1 + 2.0
            IF (NUMPXC(J).LT.1) NUMPXC(J) = 1
            IF (NUMPXC(J).GT.NSPL+1) NUMPXC(J) = NSPL + 1
  200       CPXVAL(J) = OFF1 + NUMPXC(J) - 1
            NUMPXC(NVGL) = NSPL + 1
C
C     LOOP OVER GRID CELLS COVERING OUTPUT AREA
      ISAMP = 0
      DO 300 IGX1 = 1,NVGL1
      IGX2 = IGX1 + 1
C
C     FIND NUMBER OF OUTPUT PIXELS IN THE CELL
      NPIX = NUMPXC(IGX2) - NUMPXC(IGX1)
      IF (NPIX.EQ.0) GO TO 300
C
C     CHECK FOR CELLS NOT REQUIRING INPUT DATA
      IF (CELLCG(IGY1,IGX1).EQ.ICGP+100) GO TO 250
      IF (CELLCG(IGY1,IGX1).EQ.ICGP) GO TO 210
      GO TO 300
C
C     LOAD OUTPUT COORDINATES OF GRID CELL CORNERS
  210 AX = GRDOUT(1,IGX1)
      AY = GRDOUT(2,IGY1)
      BX = GRDOUT(1,IGX2)
      CY = GRDOUT(2,IGY2)
C
C     LOAD INPUT COORDINATES OF GRID CELL CORNERS
      AS = GRIDIN(1,IGY1,IGX1)
      AL = GRIDIN(2,IGY1,IGX1)
      BS = GRIDIN(1,IGY1,IGX2)
      BL = GRIDIN(2,IGY1,IGX2)
      CS = GRIDIN(1,IGY2,IGX1)
      CL = GRIDIN(2,IGY2,IGX1)
      DS = GRIDIN(1,IGY2,IGX2)
      DL = GRIDIN(2,IGY2,IGX2)
C
C     COMPUTE FRACTIONAL CELL DISTANCE ALONG OUTPUT Y AXIS
            ACRATO = (AY-OUTLIN) / (AY-CY)
C
C     COMPUTE INTERPOLATED SAMPLE AND LINE VALUES AT CELL EDGES
      QS=AS+ACRATO*(CS-AS)
      QL=AL+ACRATO*(CL-AL)
      RS=BS+ACRATO*(DS-BS)
      RL=BL+ACRATO*(DL-BL)
C
C     COMPUTE SLOPE WITHIN THE CELL FOR SAMPLES AND LINES
      DELTAX = BX - AX
      TRANIO(3) = (RS-QS) / DELTAX
      TRANIO(4) = (RL-QL) / DELTAX
C
C     FIND INPUT COORDINATES OF FIRST POINT IN THIS GRID CELL
      UX1 = CPXVAL(IGX1) - AX
      TRANIO(1) = QS + UX1*TRANIO(3)
      TRANIO(2) = QL + UX1*TRANIO(4)
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
C
C      INTERPOLATE IMAGE DATA
             CALL NN2 (PIXEL, PIXOUT)
             GO TO 300
C
C      FILL IN THE CELL NOT REQUIRING INPUT DATA
  250 DO 255 I=1,NPIX
      ISAMP = ISAMP + 1
  255 PIXOUT(ISAMP) = KDUM
  300 CONTINUE
C
C      WRITE THE LINE SEGMENT OUT
      IF (NCGP.EQ.1) GO TO 500
      IF (ISAMP.EQ.0) GO TO 1005
      WRITE (LUNIT) ISAMP
      CALL WRITAR (LUNIT, PIXOUT, ISAMP)
      GO TO 1000
C
  500 WRITE (11) PIXOUT
 1000 CONTINUE
      GO TO 1010
C
 1005 SAMPSG(IGY1,ICGP) = 0
 1010 CONTINUE
C
      REWIND LUNIT
      WRITE (6,100) LSTOP, ICGP
 1020 CONTINUE
C
C      ASSEMBLE AND WRITE OUTPUT IMAGE
      IF (NCGP.EQ.1) RETURN
      DO 1050 IGY1=1,NMGL1
      LSTRT = NUMLNC(IGY1)
      LSTOP = NUMLNC(IGY1+1) - 1
C
C      ASSEMBLE AND WRITE OUT THE LINE SEGMENTS
      DO 1040 ILINE=LSTRT,LSTOP
      NS = 1
      DO 1030 ICGP=1,NCGP
      IF (SAMPSG(IGY1,ICGP).EQ.0) GO TO 1030
      LUNIT= ICGP + 20
      READ (LUNIT) NSAMP
      CALL READAR (LUNIT, PIXOUT(NS), NSAMP)
      NS = NS + NSAMP
 1030 CONTINUE
 1040 WRITE (11) PIXOUT
 1050 CONTINUE
      RETURN
C
  100 FORMAT ('    FINISHED RESAMPLING',I5,' RECORDS IN COLUMN GROUP',I5)
      END
      SUBROUTINE NN2 (PIXEL, PIXOUT)
C
C     PERFORM NEAREST NEIGHBOR INTERPOLATION
C
      LOGICAL*1 PIXEL(NBSAM,MAXLN), PIXOUT(1)
      INTEGER SAMPO, SLR
      INTEGER*2 MNDEX(1000)
      COMMON /PIXDAT/ NLINPX, NPIXLN, NHGL, NVGL, SRATE, NLPI, NSPL,
     *                KDUM, INTERP, NB
      COMMON / PIXIN/ NBSAM, MAXLN, MNDEX, SAMPO, LINEO
      COMMON / TRANS/ TRANIO(6)
      EQUIVALENCE (NPIX,TRANIO(5)), (ISAMP,TRANIO(6))
C
      PSI = TRANIO(1) - SAMPO + 1
      PLI = TRANIO(2) - LINEO + 1
      SPACE1 = TRANIO(3)
      SPACE2 = TRANIO(4)
C
        DO 1050 I=1,NPIX
        ISAMP = ISAMP + 1
        DE = I-1
        IPL = PLI + DE*SPACE2 + 0.5
        IF (IPL.LT.1.OR.IPL.GT.MAXLN) GO TO 1030
C
        IPS = PSI + DE*SPACE1 + 0.5
        IF (IPS.LT.1.OR.IPS.GT.NBSAM) GO TO 1030
C
        SLR = MNDEX(IPL)
        PIXCUT(ISAMP) = PIXEL(IPS,SLR)
        GO TO 1050
C
 1030 PIXOUT(ISAMP) = KDUM
 1050   CONTINUE
      RETURN
```

CHAPTER IV

DATA COMPRESSION

COMPRESSION BY ADAPTIVE DIFFERENTIAL PULSE CODE MODULATION

I. NAME

ADPCM (ADAPTIVE DIFFERENTIAL PULSE CODE MODULATION)

II. DESCRIPTION

ADPCM uses a predictive coding technique for image data compression. Employing a third order predictor, this method performs an adaptive DPCM on blocks of data 16 pixels wide. A constant bit rate is used for the entire image. A restriction for this routine is that only a 16*n pixel wide portion of the image will be processed.

III. CALLING SEQUENCE

CALL ADPCM (IV,X,IOUT)

where IV, X, IOUT are arrays with variable dimensions used in processing the data. The array dimensions required are:

IV (4, NPL)	bytes
X (2, NPL)	words
IOUT (NPL)	bytes

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

where NPL is the number of pixels per scan line.

IV. INPUT/OUTPUT

1. INPUT

The input data should be on logical unit 10 as a data set consisting of NLINE records and NPL pixels per record with four channels per pixel. Each data value has a length of one byte.

Initial parameters needed to process the image are transferred from a driver program through the following common statement:

COMMON NB, NLINE, IBIT, NPL

where

NB is the band currently being processed (integer)

NLINE is the number of records in the input image (integer)

IBIT is the approximate bit rate desired (integer)

NPL is the number of pixels per record which will be processed. It must be an integer multiple of 16.

2. OUTPUT

The input parameters and the mean values of the original and reconstructed images are printed for each band. The reconstructed image for band NB is written onto unit NB.

V. DESCRIPTION OF SUBROUTINES

The storage requirements and the functions of the non-system subroutines used are given in the following table.

DESCRIPTION OF SUBROUTINES FOR ADPCM DATA COMPRESSION

SUBROUTINE NAME (Entry Points)	STORAGE (Bytes)	FUNCTION
ADPCM	2850	Read data from input data set, call mapping and quantizing routines, call predictor routines, reconstruct and write image by calling for inverse transformations.
VARVI (VARV)	1152	Predictors for incoming data values. VARVI is used for first line only.
DSQ (QUAN)	908	Perform mapping to obtain uniform distribution of coefficients, quantize to specified number of levels, do inverse to reconstruct values.

The linkages of the subroutines are given in the following table.

LINKAGES OF SUBROUTINES FOR ADPCM DATA COMPRESSION

ADPCM	VARVI (VARV) DSQ (QUAN)

VI. <u>PERFORMANCE SPECIFICATIONS</u>

1. STORAGE REQUIREMENTS

The program requires 44K bytes of storage when operating on a LACIE sample segment.

2. EXECUTION TIME

The average speed obtained from several computer runs is 628 pixels/second.

VII. <u>METHOD</u>

ADPCM employs a block adaptive DPCM for data compression. Each row is divided into blocks 16 pixels wide and a DPCM performed on the blocks. A third order predictor is used except for the first row and the first element of each row.

The elements $X_{i-1, j+1}$, $X_{i-1, j}$ and $X_{i, j}$ are used along with weighting functions to predict $X_{i,j+1}$. The predictor equation is given by $X_{i, j+1} = A_3 \; X_{i-1, j} + A_2 X_{i, j} + A_1 X_{i-1,j+1}$

where $A_1 = A_2 = 3/4$, A3 =-1/2 and the configuration is:

$$\begin{tikzcd}
X_i\text{-1,j} & X_i\text{,j} \\
X_i\text{-1,j+1} & X_i\text{,j+1} \arrow[l, "A1"'] \arrow[u, "A2"'] \arrow[ul, "A3"]
\end{tikzcd}$$

The variance of a block is calculated and used to compute the scaling factors necessary for quantization of the block. The next point in the block is predicted and the difference between the actual and the predicted value is quantized using the scaling factors for the block. The quantized value is then used to reconstruct the point.

A constant bit rate is used throughout the image. A flow chart is given in the following figure.

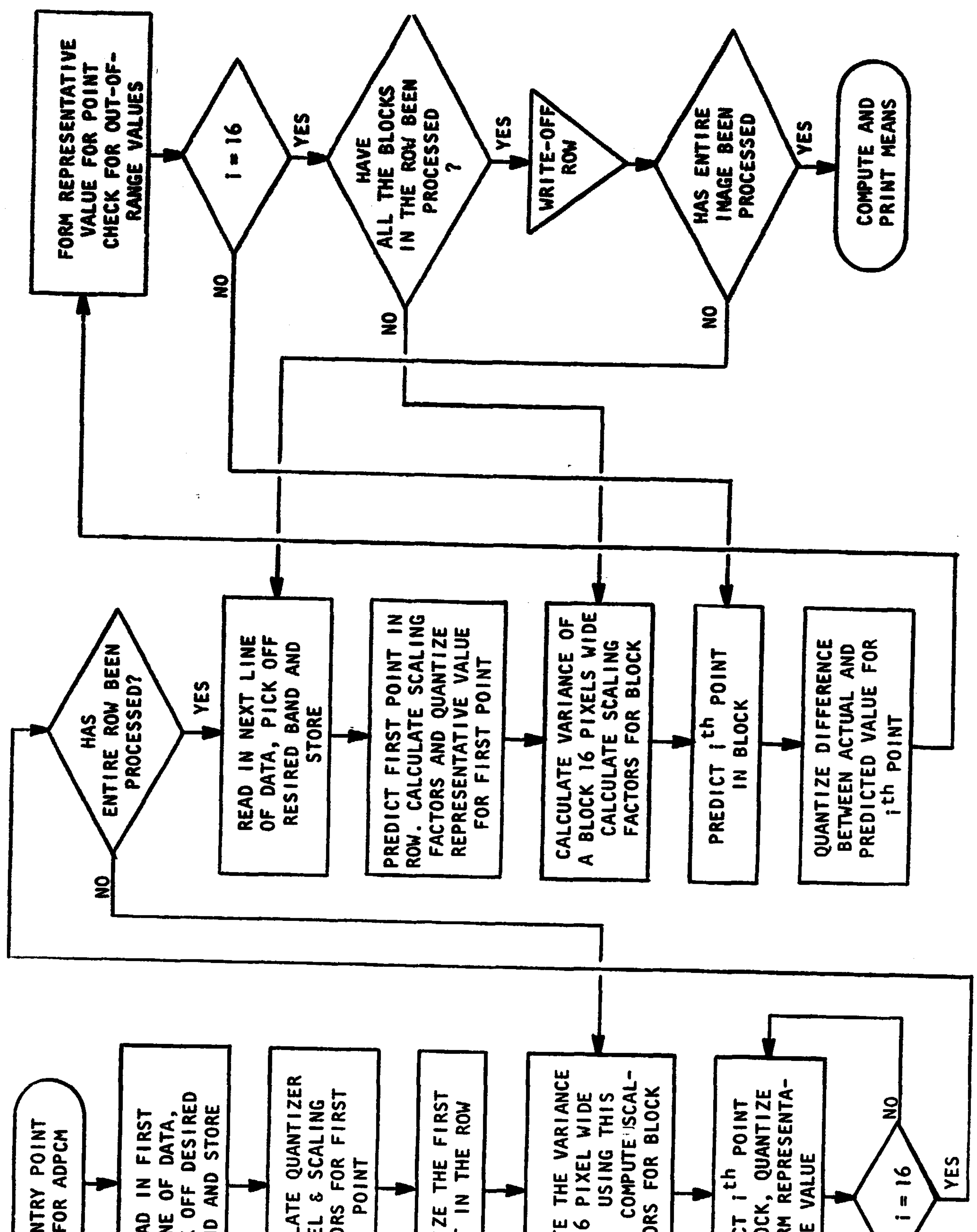

ENTRY POINT FOR ADPCM
READ IN FIRST LINE OF DATA, PICK OFF DESIRED BAND AND STORE
CALCULATE QUANTIZER LEVEL & SCALING FACTORS FOR FIRST POINT
QUANTIZE THE FIRST POINT IN THE ROW
CALCULATE THE VARIANCE OF A 16 PIXEL WIDE BLOCK. USING THIS VARIANCE, COMPUTE SCALING FACTORS FOR BLOCK
PREDICT i th POINT IN BLOCK, QUANTIZE AND FORM REPRESENTATIVE VALUE
i = 16
NO
YES
HAS ENTIRE ROW BEEN PROCESSED?
NO
YES
READ IN NEXT LINE OF DATA, PICK OFF RESIRED BAND AND STORE
PREDICT FIRST POINT IN ROW. CALCULATE SCALING FACTORS AND QUANTIZE REPRESENTATIVE VALUE FOR FIRST POINT
CALCULATE VARIANCE OF A BLOCK 16 PIXELS WIDE CALCULATE SCALING FACTORS FOR BLOCK
PREDICT i th POINT IN BLOCK
QUANTIZE DIFFERENCE BETWEEN ACTUAL AND PREDICTED VALUE FOR i th POINT
FORM REPRESENTATIVE VALUE FOR POINT CHECK FOR OUT-OF-RANGE VALUES
i = 16
YES
NO
HAVE ALL THE BLOCKS IN THE ROW BEEN PROCESSED ?
YES
NO
WRITE-OFF ROW
HAS ENTIRE IMAGE BEEN PROCESSED
YES
NO
COMPUTE AND PRINT MEANS

Figure 9. Flow Chart for ADPCM Compression

VIII. COMMENTS

There is no restriction on the number of records in the input image but only a 16*n pixels wide segment will be processed.

IX. TESTS

The quality of the reconstructed images has been examined by use of mean values, plots and histograms of the reconstructed images, and plots and histograms of difference images. An example of difference image histograms follows.

X. LISTINGS

Listings of the subroutines follow.

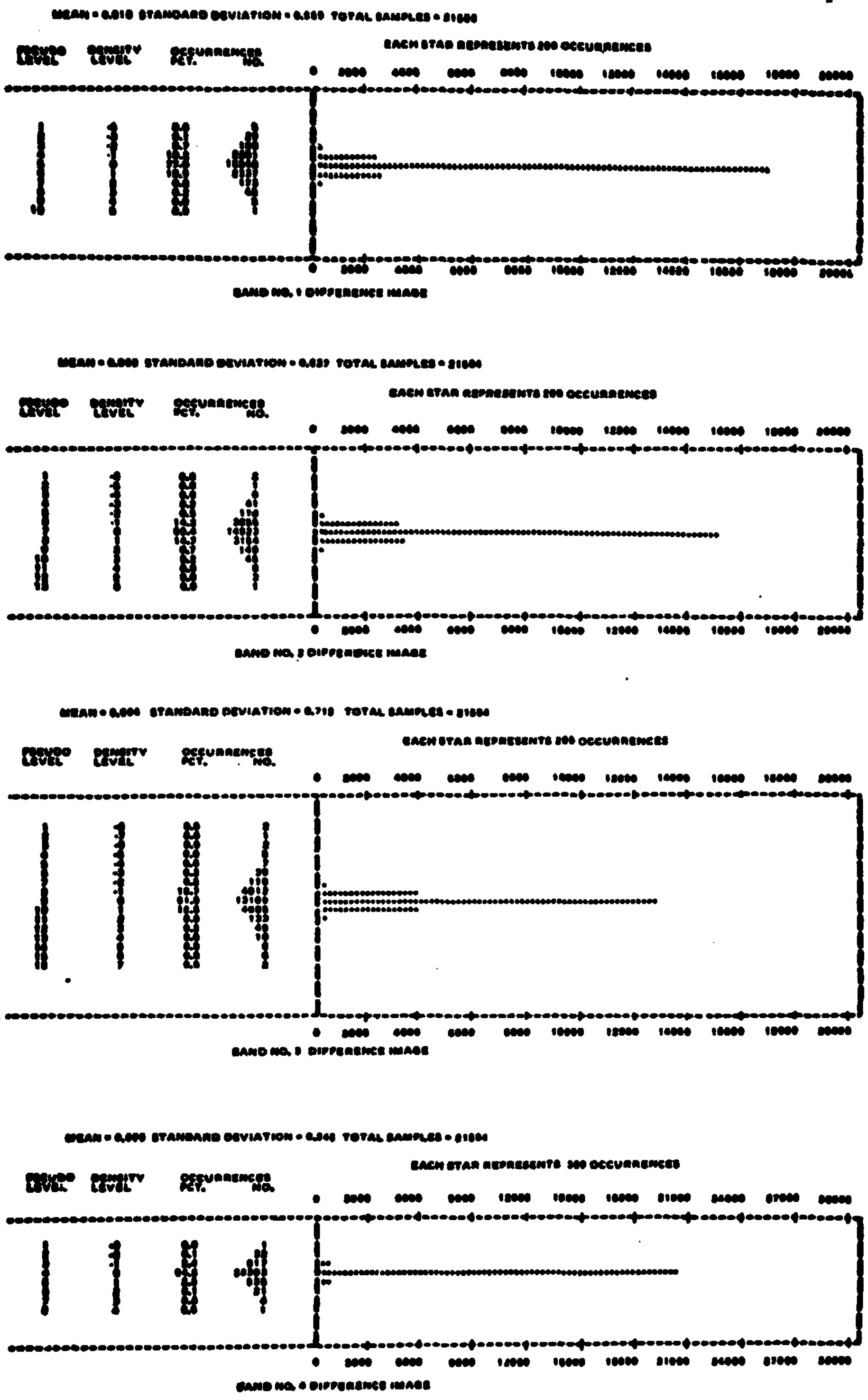


Figure 10. Histograms of Difference Images for Four Bands (original - ADPCM Reconstruction)

```
      SUBROUTINE ADPCM (IV, X, IOUT)
C
C     ADPCM PERFORMS AN ADAPTIVE 2-D DPCM ON AN IMAGE 16*N PIXELS WIDE.
C     THERE IS NO RESTRICTION ON THE NUMBER OF LINES IN THE IMAGE.
C     INPUT IMAGE IS PROCESSED ONE BAND AT A TIME
C
C        NPL   = NUMBER OF PIXELS PER LINE (16*N)
C        NLINE = NUMBER OF LINES
C        NBPL  = NUMBER OF BLOCKS PER LINE
C        BLK   = NUMBER OF PIXELS PER BLOCK
C        IBIT  = APPROXIMATE BIT RATE
C
C     LOGICAL UNITS
C       10 = INPUT PICTURE (BYTES)
C       NB = RECONSTRUCTED BAND NO. NB (BYTES)
C '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
C **********************************************************************
C
      DIMENSION IV(4,NPL), X(2,NPL), IOUT(NPL), A(3)
      LOGICAL*1 IV, IOUT
      INTEGER BLK /16/
      REAL MAXP,MINP,MAX(4)/3*127.0,63.0/,MIN(4)/4*0.0/
      COMMON NB, NLINE, IBIT, NPL
      DATA GAIN /2.5/, CONH, CONV /2*1.0/, A /0.75, 0.75, -0.5/
C
C     INITIALIZE COUNTERS, SET IMAGE PARAMETERS
      ISUM1=0
      ISUM2=0
      NBPL = NPL/BLK
      MAXP = MAX(NB)
      MINP = MIN(NB)
C
C     THE FIRST LINE OF THE DATA IS HANDLED SEPARATELY.
C     READ THE FIRST LINE OF DATA INTO ARRAY IV, PICK OFF THE DESIRED
C     BAND AND STORE IT IN X(1,J)
      READ (10) IV
      DO 120 J=1,NPL
      IIV = IV(NB,J)
      X(1,J) = IIV
      ISUM1 = ISUM1 + IIV
  120 CONTINUE
C
C     CALCULATE QUANTIZER LEVEL AND SCALING FOR THE FIRST POINT.
      LEVEL=2**(IBIT-1)
      SDE=MAXP/GAIN
      CALL DSQ (GAIN, SDE)
C
C     CALCULATE THE REPRESENTATIVE VALUE FOR THE FIRST PIXEL
      F = X(1,1)
      CALL QUAN (F, LEVEL, EQ)
      X(1,1) = EQ
C
C     CALCULATE VARIANCE, S, OF A BLOCK.  USING VARIANCE COMPUTE THE
C     SCALING FACTOR FOR THE REST OF THE BLOCK.
      PAR=MAXP/8.
      IF(PAR.LT.8.)PAR=8.
      DO 199 JJ=1,NBPL
      CALL VARV1 (X, CONV, JJ, S, PAR, NPL)
      IF (JJ.EQ.1) SLAG = S
      CALL DSQ (2,5, S)
      JL=(JJ-1)*BLK+1
      JH=JJ*BLK
      IF(JH.EQ.NPL) JH=NPL-1
C
C     PREDICT NEXT POINT IN BLOCK, QUANTIZE AND FORM REPRESENTATIVE
C     VALUE.  CONTINUE UNTIL ALL POINTS IN BLOCK ARE PROCESSED.
      DO 190 J=JL,JH
      J1=J+1
      EX = CONH*X(1,J)
      F = X(1,J1) - EX
      CALL QUAN (F, LEVEL, EQ)
      X(1,J1) = EX + EQ
  190 CONTINUE
  199 CONTINUE
C
C     READ IN NEXT LINE OF DATA, PICK OFF APPROPRIATE BAND AND STORE
      DO 299 II=2,NLINE
      READ (10) IV
      DO 240 J=1,NPL
      IIV = IV(NB,J)
      X(2,J) = IIV
      ISUM1 = ISUM1 + IIV
  240 CONTINUE
C
C     PREDICT THE FIRST POINT, CALCULATE THE SCALING FACTOR FOR
C     VARIANCE OF NEW ROW WITH RESPECT TO THE PREVIOUS ROW, QUANTIZE
C     AND FORM REPRESENTATIVE VALUE FOR FIRST POINT OF THE ROW.
      CALL DSQ (2,5, SLAG)
      EX = CONV*X(1,1)
      F = X(2,1) - EX
      CALL QUAN (F, LEVEL, EQ)
      X(2,1) = EX + EQ
C
C     CALCULATE VARIANCE, S, FOR A BLOCK.  USING THIS VARIANCE COMPUTE
C     THE SCALING FACTOR FOR THE REST OF THE BLOCK.
      DO 260 JJ=1,NBPL
      CALL VARV (X, A, JJ, S, PAR, NPL)
      CALL DSQ (2,5, S)
      JL=(JJ-1)*BLK+1
      JH=JJ*BLK
      IF(JH.EQ.NPL) JH=NPL-1
C
C     COMPUTE THE REPRESENTATIVE VALUE OF THE NEXT POINT IN THE BLOCK
C     BY QUANTIZING THE DIFFERENCE BETWEEN THE NEXT POINT AND THE
C     PREDICTED VALUE AND THEN ADDING THE QUANTIZED VALUE TO THE
C     PREDICTED VALUE.
C     ROTATE THE PREDICTOR VALUES FROM ROW 2 TO ROW 1
      DO 250 J=JL,JH
      J1=J+1
      EX = A(1)*X(2,J) + A(3)*X(1,J) + A(2)*X(1,J1)
      F = X(2,J1) - EX
      CALL QUAN (F, LEVEL, EQ)
      X(2,J1) = EX + EQ
      IF (X(1,J).GT.MAXP) X(1,J) = MAXP
      IF (X(1,J).LT.MINP) X(1,J) = MINP
      IOUT(J) = X(1,J) + 0.5
      ISUM2 = ISUM2 + IOUT(J)
      X(1,J) = X(2,J)
  250 CONTINUE
  260 CONTINUE
C
C     LOAD OUTPUT ARRAY AND MOVE PREDICTOR VALUE FOR LAST SAMPLE
      IF (X(1,NPL).GT.MAXP) X(1,NPL) = MAXP
      IF (X(1,NPL).LT.MINP) X(1,NPL) = MINP
      IOUT(NPL) = X(1,NPL) + 0.5
      ISUM2 = ISUM2 + IOUT(NPL)
      X(1,NPL) = X(2,NPL)
C
C     WRITE RECONSTRUCTED IMAGE FOR A LINE OF DATA ONTO UNIT # NB.
      WRITE (NB) IOUT
  299 CONTINUE
C
C     WRITE LAST RECONSTRUCTED IMAGE LINE ON UNIT NB.
      DO 300 J=1,NPL
      IF (X(1,J).LT.MINP) X(1,J) = MINP
      IF (X(1,J).GT.MAXP) X(1,J) = MAXP
      IOUT(J) = X(1,J) + 0.5
  300 ISUM2 = ISUM2 + IOUT(J)
      WRITE (NB) IOUT
C
C     COMPUTE MEANS FOR ORIGINAL AND RECONSTRUCTED IMAGE.
      PIX=NLINE*NPL
      AMEAN1 = ISUM1 / PIX
      AMEAN2 = ISUM2 / PIX
C
      WRITE(6,38)
      WRITE(6,35) NPL,NLINE,NB,IBIT,MAXP
      WRITE(6,41) AMEAN1, AMEAN2
      RETURN
C
   35 FORMAT(' NPL = ',I4,10X,'NLINE = ',I4,10X,'BAND NO. = ',I2,13X,
     ,'IBIT = ',I5,15X,'MAXP = ',F6.2)
   38 FORMAT(50X,'ADAPTIVE TWO-DIMENSIONAL DPCM',//)
   41 FORMAT (/' MEAN OF ORIGINAL IMAGE =',F8.3,24X,'MEAN OF RECONSTRUCT
     ,ED IMAGE =',F8.3///)
      END
      SUBROUTINE VARV1 (X, COVH, K, S, PAR, NPL)
C
C     SUBROUTINE VARV1 COMPUTES THE VARIANCES FOR A BLOCK OF DATA IN THE
C     FIRST ROW.  PREDICTED VALUE IS THE VALUE OF THE PREVIOUS SAMPLE
C     TIMES A WEIGHTING FACTOR.
C ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C **********************************************************************
C
      DIMENSION X(2,1),A(3)
      M1 = NPL - 1
      S=0.0
      N1=(K-1)*16+1
      N2=N1+15
      IF (N2.EQ.NPL) N2=M1
      DO 100 J=N1,N2
      J1=J+1
      EX=COVH*X(1,J)
      E=X(1,J1)-EX
      S=S+E*E
  100 CONTINUE
      S=S/16.
      IF(N2.EQ.M1)S=S*1.0606
      S=SQRT(S)
      IS=S*8/PAR
      IF(IS.GE.8) IS = 7
      S=(IS+0.5)*PAR/8.0
      RETURN
C
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C***********************************************************************
      ENTRY VARV (X, A, K, S, PAR, NPL)
C
C     SUBROUTINE VARV CALCULATES THE VARIANCE FOR A BLOCK OF 16 PIXELS.
C     VARV USES A THIRD ORDER PREDICTOR THAT UTILIZES THE ADJACENT
C     ELEMENT IN THE SAME LINE, THE ADJACENT ELEMENT IN THE SAME COLUMN,
C     AND THE DIAGONAL ELEMENT TO PREDICT EACH SAMPLE.  THE FIRST ROW OF
C     DATA CANNOT BE HANDLED BY THIS ROUTINE.
C ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C **********************************************************************
C
      M1 = NPL - 1
      S=0.0
      N1=(K-1)*16+1
      N2=N1+15
      IF (N2.EQ.NPL) N2=M1
      DO 1000 J=N1,N2
      J1=J+1
      EX=A(1)*X(2,J)+A(3)*X(1,J)+A(2)*X(1,J1)
      E=X(2,J1)-EX
      S=S+E*E
 1000 CONTINUE
      S=S/16.
      IF(N2.EQ.M1) S=S*1.0606
      S=SQRT(S)
      IS=S*8/PAR
      IF(IS.GE.8) IS=7
      S=(IS+0.5)*PAR/8.0
      RETURN
      END
      SUBROUTINE DSQ (XMULT, SIGMA)
C
C     AN ASSUMED LAPLACIAN DISTRIBUTION OF THE COEFFICIENTS IS TRANS-
C     FORMED TO A UNIFORM DISTRIBUTION BEFORE QUANTIZATION.
C     DSQ COMPUTES THE CONSTANTS OF THE MAPPING FUNCTION.
C        XMULT = ESTIMATED NUMBER OF SIGMAS IN THE DATA RANGE
C        SIGMA = ESTIMATED VARIANCE OF THE INPUT VALUE
C ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C **********************************************************************
C
      EMAX=XMULT*SIGMA
      EM = SQRT(2.0*EMAX) / (3.0*SIGMA)
      EXPM=EXP(-EM)
      EM1 = -EM/EMAX
      RETURN
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C***********************************************************************
C
      ENTRY QUAN (F, LEVEL, EQ)
C
C      QUANTIZATION OF AC COEFFICIENTS.
C        F     = VALUE TO BE QUANTIZED
C        LEVEL = NUMBER OF LEVELS IN THE QUANTIZER
C        EQ    = REPRESENTATIVE VALUE FOR F
C ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C **********************************************************************
C
      II=0
      IF(F.GE.0.0)GO TO 2
      F=-F
      II=1
 2    CONTINUE
C
C     FORWARD MAPPING
      EW = F*EM1
      EXPE=EXP(EW)
      Z=EMAX*(1.-EXPE)/(1.-EXPM)
C
      LEVEL1=LEVEL-1
      IZ=(Z/EMAX)*LEVEL
      IF(IZ.LT.0)IZ=0
      IF(IZ.GT.LEVEL1)IZ=LEVEL1
C
C     INVERSE MAPPING
      ZQ=(FLOAT(IZ)+0.5)/FLOAT(LEVEL)
      ZQW=1.-ZQ*(1.-EXPM)
      EQ=-EMAX/EM*ALOG(ZQW)
      IF(II.EQ.1)EQ=-EQ
      RETURN
      END
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

COMPRESSION BY TWO DIMENSIONAL TRANSFORMS

I. <u>NAME</u>

TRANC

II. <u>DESCRIPTION</u>

TRANC employs an adaptive transform coding method for image data compression. Using a fixed block size and a fixed transformation (Hadamard or Cosine), this method performs a blocked, two-dimensional orthogonal transformation on a 16*m row by 16*n pixel image, where m and n are integers. The variances of the transformed coefficients are estimated using a recursive formula. A bit assignment is made proportional to the logarithm of the estimated variance; therefore, the number of bits assigned to each coefficient varies from block to block. The image is reconstructed by using inverse transformations.

III. <u>CALLING SEQUENCE</u>

```
CALL TRANC (IV,IOUT,D)     - Cosine Transformation
CALL TRANH (IV,OUT,D)      - Hadamard Transformation
```

where IV,IOUT and D are arrays with variable dimensions. The array dimensions required are:

$$\left.\begin{array}{r}\text{IV (4, NPL)}\\ \text{IOUT (NPL)}\end{array}\right\}\text{bytes}$$
$$\text{D (16, NPL) words}$$

where NPL is the number of pixels per scan line.

IV. <u>INPUT/OUTPUT</u>

1. INPUT

The input data should be on logical unit 10 as a data set consisting of NLINE records and NPL pixels per record with four channels per pixel. Each data value has a length of one byte.

Initial parameters needed to process the image are transferred from a driver program through the following common statement:
COMMON NB, NLINE, FIXB, NPL
where

- o NB is the band currently being processed (integer)
- o NLINE is the number of records in the input image which will be processed. It must be an integer multiple of 16. (integer)
- o FIXB is the approximate bit rate desired (floating point)
- o NPL is the number of pixels per record which will be processed. It must be an integer multiple of 16. (integer)

2. OUTPUT

The input parameters, the average bit rate, and the mean values of the original and reconstructed images are printed for each band.

The reconstructed image for band NB is written onto unit NB.

V. DESCRIPTION OF SUBROUTINES

The storage requirements and the functions of the non-system subroutines used are given in the following table.

DESCRIPTION OF SUBROUTINES FOR 2-D TRANSFORM DATA COMPRESSION

SUBROUTINE NAME (Entry Points)	STORAGE (Bytes)	FUNCTION
TRANC (TRANH)	6202	Read 16 line blocks from input data set, set up arrays to do 2-D Cosine or Hadamard transforms, map 2-D arrays into 1-D, estimate recursive variance, calculate bit rate, quantize, reconstruct and write image by calling for inverse transformations.
MAP (UNMAP)	1524	Map 16 x 16 array of transformed coefficients into 1-D array, and do inverse.

DESCRIPTION OF SUBROUTINES FOR 2-D

TRANSFORM DATA COMPRESSION (CONT.)

SUBROUTINE NAME (Entry Points)	STORAGE (Bytes)	FUNCTION
DSQ (QUAN)	908	Perform mapping to obtain uniform distribution of coefficients, quantize to specified number of levels, do inverse to reconstruct coefficient values.
HADD	1110	Hadamard transformation of a string of 16 pixels.
COST	1844	Cosine transformation of a string of 16 pixels.
MDFT	1826	
MFORT	1870	

The linkages of the subroutines are given in the following table.

LINKAGES OF SUBROUTINES FOR 2-D

TRANSFORM DATA COMPRESSION

TRANC (TRA[illegible]H)	HADD COST MAP UNMAP DSQ QUAN
COST	MDFT
MDFT	MFORT

VI. PERFORMANCE SPECIFICATIONS

1. STORAGE REQUIREMENTS

The program requires 66K bytes of storage when operating on a LACIE sample segment.

2. EXECUTION TIME

The speed of the program is highly dependent on the type of transform applied, and somewhat dependent on the bit rate, being slower at higher bit rates due to the quantization of a greater number of coefficients. Average speeds from several computer runs are given in the following table.

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

PROCESSING SPEEDS FOR 2-D TRANSFORM COMPRESSION

(4 band pixels/second)

BIT RATE	COSINE SPEED	HADAMARD SPEED
1/2	150	488
1	141	421
2	127	314
3	118	268
4	122	259

VII. <u>METHOD</u>

The adaptive transform method uses a two-dimensional orthogonal transformation followed by recursive quantization to compress an image. Initially the input image is divided into blocks of 16 x 16 pixels. After a two-dimensional Hadamard or Cosine transformation is performed on a block of data, a scanning method is used to convert the two-dimensional array into a one-dimensional array. The variance of the transformed coefficients are estimated using the first-order recursive relation

$$\sigma_{i+1}^2 = A\sigma_i^2 + (1-A)\,X_i^2$$

where X_i is the i-th transformed coefficient after quantization and σ_i^2 is the estimated variance of X_i. A is a weighting coefficient which is set to 0.75. The variance, σ_2^2, needed to start this recursive relation is obtained by averaging the sum of the squares of the first four coefficients and then quantizing this average using 32 levels. For the actual quantization a nonuniform quantizer designed for a Laplacian distribution is used. Experimental results demonstrate that for most images, the probability density function of the variance is a Laplacian probability density function. The mapping, g(.), of the transformed coefficients yields a signal, Z, with a uniform probability density function. Z is then quantized and the result undergoes the inverse mapping, $g^{-1}(.)$. The mapping, g(.) for an exponential probability density function is given by

$$Z = g(x) = \frac{X_o\left[1-\exp\left(-Mx/X_o\right)\right]}{1-\exp\left(-M\right)}$$

where

$$X_o = 3\sigma$$

$$M = \frac{\sqrt{2}X_o}{3\sigma}$$

The inverse mapping is given by $g^{-1}(x) = \frac{-X_o}{M} \ln\left[1 - \frac{Z^*}{M}\left(1-\exp(-M)\right)\right]$.

The number of bits, m_i, assigned to each coefficient is obtained from

$$m_i = \text{Integer}\left[\tfrac{1}{2}\log_2 \sigma_i^2 + C\right]$$

where C is a constant which is adjusted to correct for variations from the desired bit rate. If the number of bits assigned to a coefficient is less than one, the remaining coefficients in the block are not transmitted. With this type of bit assignment, coefficients with large variances are assigned a greater number of bits.

In order to reconstruct the image, the one-dimensional array is mapped back into a two-dimensional format. The inverse of the two-dimensional orthogonal transformation is taken, a check is made for out-of-range values, the data is rounded off to pack into bytes, and is written out.

A flow chart of the algorithm is given in the following figure.

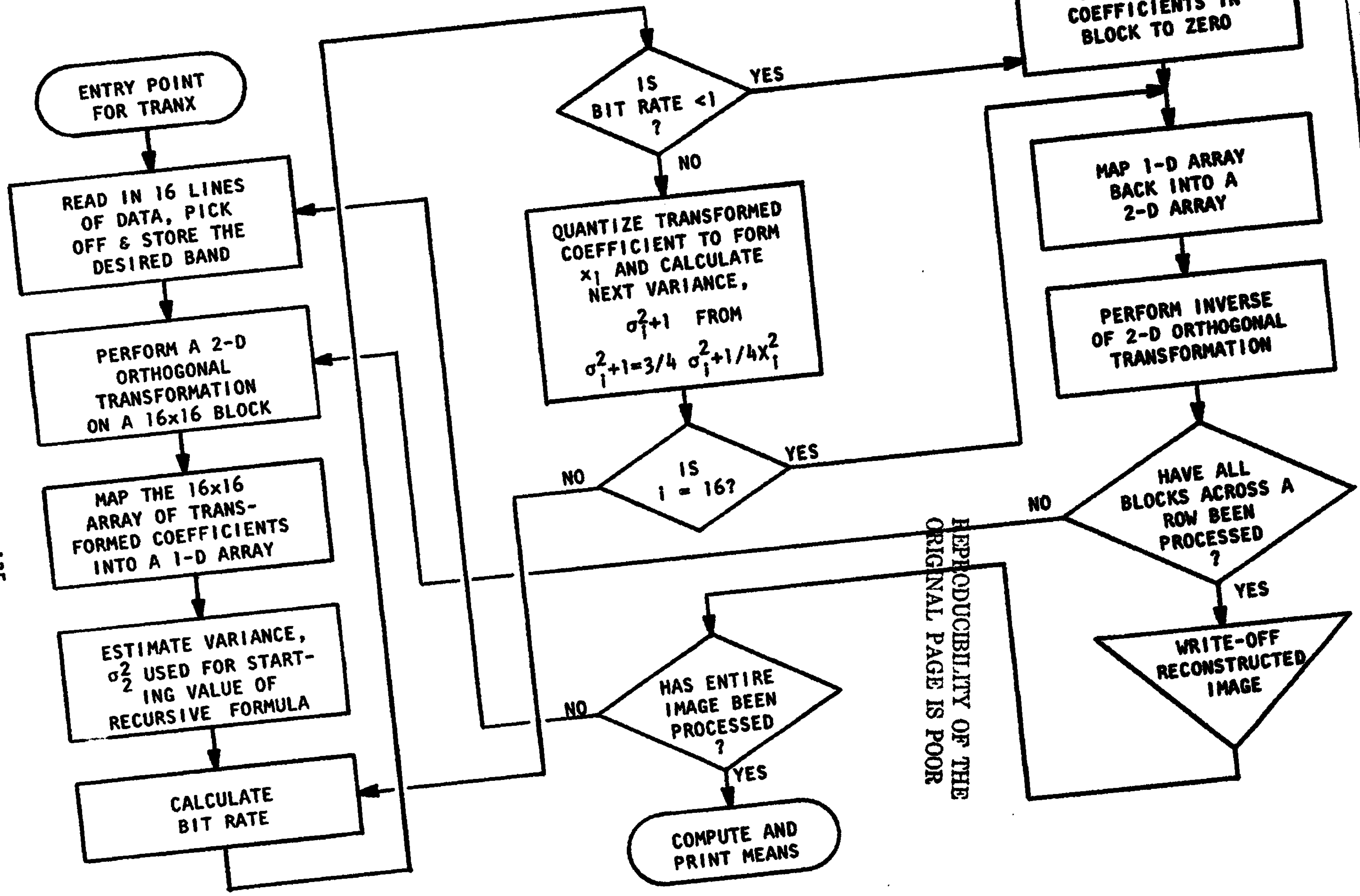

ENTRY POINT FOR TRANX
READ IN 16 LINES OF DATA, PICK OFF & STORE THE DESIRED BAND
PERFORM A 2-D ORTHOGONAL TRANSFORMATION ON A 16x16 BLOCK
MAP THE 16x16 ARRAY OF TRANS-FORMED COEFFICIENTS INTO A 1-D ARRAY
ESTIMATE VARIANCE, σ_2^2 USED FOR START-ING VALUE OF RECURSIVE FORMULA
CALCULATE BIT RATE
IS BIT RATE <1 ?
YES
NO
QUANTIZE TRANSFORMED COEFFICIENT TO FORM x_i AND CALCULATE NEXT VARIANCE, σ_i^2+1 FROM $\sigma_i^2+1=3/4\ \sigma_i^2+1/4x_i^2$
IS i = 16?
YES
NO
SET REMAINING COEFFICIENTS IN BLOCK TO ZERO
MAP 1-D ARRAY BACK INTO A 2-D ARRAY
PERFORM INVERSE OF 2-D ORTHOGONAL TRANSFORMATION
HAVE ALL BLOCKS ACROSS A ROW BEEN PROCESSED ?
YES
NO
WRITE-OFF RECONSTRUCTED IMAGE
HAS ENTIRE IMAGE BEEN PROCESSED ?
YES
NO
COMPUTE AND PRINT MEANS

Figure 11. Flow Chart for 2-D Transform Compression Method

VIII. COMMENTS

Only a 16*m row by 16*n pixel portion of the input image can be processed.

IX. TESTS

The quality of the reconstructed images has been examined by use of mean values, plots and histograms of the reconstructed images, and plots and histograms of difference images. An example of output from the program follows, showing the mean values of the original and reconstructed images.

X. LISTINGS

Listings of the subroutines follow.

```
                                LACIE DATA

          ADAPTIVE TRANSFORM CODING USING TWO-DIMENSIONAL COSINE TRANSFORM

NPL =  192                NLINE =  112              BAND NO. 1

MAXP =127.0               MINP =  0.0               FIXB =  3.0

AVERAGE BIT RATE =     3.170

MEAN OF ORIGINAL IMAGE =  14.754                    MEAN OF RECONSTRUCTED IMAGE =  14.752

          ADAPTIVE TRANSFORM CODING USING TWO-DIMENSIONAL COSINE TRANSFORM

NPL =  192                NLINE =  112              BAND NO. 2

MAXP =127.0               MINP =  0.0               FIXB =  3.0

AVERAGE BIT RATE =     3.110

MEAN OF ORIGINAL IMAGE =  18.249                    MEAN OF RECONSTRUCTED IMAGE =  18.249

          ADAPTIVE TRANSFORM CODING USING TWO-DIMENSIONAL COSINE TRANSFORM

NPL =  192                NLINE =  112              BAND NO. 3

MAXP =127.0               MINP =  0.0               FIXB =  3.0

AVERAGE BIT RATE =     3.183

MEAN OF ORIGINAL IMAGE =  18.738                    MEAN OF RECONSTRUCTED IMAGE =  18.741

          ADAPTIVE TRANSFORM CODING USING TWO-DIMENSIONAL COSINE TRANSFORM

NPL =  192                NLINE =  112              BAND NO. 4

MAXP = 63.0               MINP =  0.0               FIXB =  3.0

AVERAGE BIT RATE =     3.112

MEAN OF ORIGINAL IMAGE =   8.953                    MEAN OF RECONSTRUCTED IMAGE =   8.952
```

Figure 12. Printed Output of the 2D Transform Compression Program

```
      SUBROUTINE TRANC (IV, IOUT, D)
C
C     TRANX IS AN ADAPTIVE TRANSFORM CODING PROGRAM WHICH PERFORMS
C     A 16 X 16 BLOCKED HADAMARD OR COSINE TRANSFORM ON A 16*N PIXELS
C     BY 16*M ROW IMAGE.  THE TWO-DIMENSIONAL ARRAY IS MAPPED INTO A
C     ONE-DIMENSIONAL ARRAY IN A ZIGZAG MANNER.
C     A FIRST ORDER RECURSIVE RELATION IS USED TO ESTIMATE VARIANCE
C     OF EACH TRANSFORMED COEFFICIENT = WHT*VA+(1-WHT)*CURRENT REP VALUE
C     A LAPLACIAN FUNCTION IS USED TO MODEL THE PROBABILITY DENSITY
C     FUNCTION OF THE AC TRANSFORMED COEFFICIENTS.
C     THE IMAGE IS RECONSTRUCTED BY MAPPING
C     BACK INTO A 16 X 16 ARRAY AND PERFORMING A 2-D INVERSE MAPPING.
C     TRANC PERFORMS THE PROCESS FOR ONE BAND AT A TIME.
C     INPUT AND RECONSTRUCTED IMAGES ARE IN BYTE ARRAYS.
C     THIS METHOD USES A FIXED TRANSFORMATION AND A FIXED 16 X 16
C     BLOCK SIZE.  THE NUMBER OF BITS ASSIGNED TO EACH COEFFICIENT
C     VARIES FROM BLOCK TO BLOCK.
C
C     INPUT PARAMETERS
C        NB     = BAND NUMBER TO BE PROCESSED
C        NPL    = NUMBER OF PIXELS PER LINE
C        NLINE  = NUMBER OF LINES TO BE READ
C        FIXB   = APPROXIMATE BIT RATE
C
C        ITT    = TRANSFORM TYPE
C                 1 = HADAMARD (TRANH)  2 = COSINE (TRANC)
C        NBPL   = NUMBER OF BLOCKS PER LINE
C        NBPV   = NUMBER OF BLOCKS IN VERTICAL DIRECTION
C        NTB    = NUMBER OF TOTAL 16 X 16 BLOCKS IN IMAGE
C        MAXP   = MAXIMUM VALUE OF PICTURE
C        MINP   = MINIMUM VALUE OF PICTURE
C        WHT    = WEIGHTING COEFFICIENT FOR RECURSIVE RELATION USED
C                 TO ESTIMATE VARIANCE
C        INITB  = NUMBER OF BITS REQUIRED FOR FIRST VARIANCE (5)
C        IBIT   = NUMBER OF BITS ASSIGNED TO EACH COEFFICIENT
C        ITCNT  = COUNTER FOR THE NUMBER OF BITS USED FOR ENTIRE IMAGE
C        ICCNT  = COUNTER FOR NUMBER OF BITS USED FOR EACH 16 X 16 BLOCK
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      DIMENSION IV(4,NPL), IOUT(NPL), D(16,NPL), A(16), B(16), V(256),
     , C(16,16)
      REAL ITCNT, MAXP, MINP, MAX(4) /3*127.0,63.0/, MIN(4) /4*0.0/
      LOGICAL*1 IV, IOUT
      COMMON NB, NLINE, FIXB, NPL
      DATA M, N /16, 256/
C
C     ENTRY FOR COSINE TRANSFORM
      ITT = 2
      WRITE (6,35)
      GO TO 1
C
C     ENTRY FOR HADAMARD TRANSFORM
      ENTRY TRANH (IV, ICUT, D)
      ITT = 1
      WRITE (6,34)
```

```
C
C     INPUT PARAMETERS
    1 NBPL = NPL/M
      NBPV=NLINE/M
      NTB=NBPV*NBPL
      ALN2 = ALOG(2.0)
      WHT=0.75
      WHTC=1.-WHT
      INITB=5
C
C     INITIALIZE COUNTERS, SET IMAGE PARAMETERS
      ITCNT=0.0
      ISUM1=0
      ISUM2=0
      MAXP = MAX(NB)
      MINP = MIN(NB)
      CCC=MAXP+1.0
      DCONS=FIXB-ALOG(CCC/128.)/ALOG(2.)
C
C     A 16 X NPL BLOCK OF DATA IS INPUT FOR ONE BAND.  AFTER PROCESSING
C     ENTIRE BLOCK THE RECONSTRUCTED IMAGE IS WRITTEN OFF AND THE
C     NEXT 16 X NPL BLOCK IS INPUT.  THIS CONTINUES UNTIL ALL THE DATA
C     FOR ONE BAND IS PROCESSED.
      DO 950 L=1,NBPV
C
C     READ DATA ONE LINE AT A TIME, PICK OFF DESIRED BAND AND STORE
C     IN ARRAY D.  CONTINUE UNTIL 16 LINES HAVE BEEN READ.  AFTER
C     COMPLETION, D CONTAINS A 16 X NPL BLOCK OF DATA.
      DO 100 I=1,M
      READ (10) IV
      DO 100 J=1,NPL
      IIV = IV(NB,J)
      D(I,J) = IIV
      ISUM1 = ISUM1 + IIV
  100 CONTINUE
C
C     A 16 X 16 BLOCK OF DATA IS TRANSFORMED, ENCODED AND
C     RECONSTRUCTED.  THIS CONTINUES UNTIL THE ENTIRE 16 X NPL BLOCK
C     OF DATA IS PROCESSED.
      DO 900 NH=1,NBPL
C
C     PERFORM A 2-D ORTHOGONAL TRANSFORM (HADAMARD OR COSINE)
C     ON A 16 X 16 BLOCK OF DATA
      DO 230 J=1,M
      JJ=(NH-1)*M+J
      DO 220 I=1,M
  220 A(I) = D(I,JJ)
      GO TO (221,222), ITT
  221 CALL HADD (A, B)
      GO TO 225
  222 CALL COST (A, B, 1)
  225 CONTINUE
      DO 230 I=1,M
  230 C(I,J)=B(I)
C
```

```
      DO 250 I=1,M
      DO 240 J=1,M
  240 A(J) = C(I,J)
      GO TO (241,242), ITT
  241 CALL HADD (A, B)
      GO TO 245
  242 CALL COST (A, B, 1)
  245 CONTINUE
      DO 250 J=1,M
  250 C(I,J)=B(J)
C
C     A 16 X 16 BLOCK OF 2-D TRANSFORMED DATA IS CONVERTED TO A
C     1-D FORMAT
      CALL MAP(C,V)
C
C     AVERAGE THE SUM OF THE SQUARES OF THE FIRST 4 AC COEFFICIENTS AND
C     QUANTIZE TO FORM THE FIRST VARIANCE.
      S=0.0
      DO 310 J=2,5
      S=S+V(J)*V(J)
  310 CONTINUE
      VA=S/4.0
      S=SQRT(VA)
      XMULT=3.0
      CALL DSQ (XMULT, S)
      LEVEL=2**INITB
      CALL QUAN (S, LEVEL, EQ)
      VA=EQ*EQ
C
C     QUANTIZATION OF COEFFICIENTS
C
C     ASSIGN THE NUMBER OF BITS TO EACH COEFFICIENT.  INCREMENT
C     COUNTERS.  IF THE NUMBER OF BINARY DIGITS ASSIGNED FOR THE
C     QUANTIZATION IS LESS THAN ONE, THE REMAINING CCEFFICIENTS IN THE
C     BLOCK ARE NOT TRANSMITTED.
      ICONT=0
      V(1) = AINT (V(1)+0.5)
      DO 350 I=2,N
      IBIT = 0.5*ALOG(VA)/ALN2 + DCONS
      IF(IBIT.EQ.1)XMULT=1.8
      IF(IBIT.LT.1) GO TO 360
      ICONT=ICONT+IBIT
      S=SQRT(VA)
      CALL DSQ (XMULT, S)
      LEVEL=2**(IBIT-1)
      CALL QUAN (V(I), LEVEL, EQ)
      V(I)=EQ
      VA=WHT*VA+WHTC*EQ*EQ
  350 CONTINUE
      GO TO 375
C
  360 DO 370 J=I,N
  370 V(J)=0.0
C
C     ADD OVERHEAD BITS AND ADJUST 'DCONS' BASED ON DESIRED BIT RATE
```

```
  375 CONTINUE
      IBITDC = ALOG(V(1))/ALN2 + 1.0
      ICONT = ICONT + INITB + IBITDC
      XICONT=ICONT
      II=NH+(L-1)*NBPL
      DCONS=DCONS+1./FLOAT(II)*(FIXB-XICONT/256.)
      ITCNT=ITCNT+FLOAT(ICONT)/256.
C
C     MAP 1-D ARRAY OF COEFFICIENTS BACK INTO A 16 X 16 BLOCK.
      CALL UNMAP(C,V)
C
C     PERFORM INVERSE OF 2-D TRANSFORMATION ON A 16 X 16 BLOCK
C     CHECK FOR POINTS WHICH ARE OUT-OF-RANGE.
      DO 440 J=1,M
      DO 430 I=1,M
  430 A(I) = C(I,J)
      GO TO (431,432), ITT
  431 CALL HADD (A, B)
      GO TO 435
  432 CALL COST (A, B, -1)
  435 CONTINUE
      DO 440 I=1,M
  440 C(I,J)=B(I)
C
      DO 460 I=1,M
      DO 450 J=1,M
  450 A(J) = C(I,J)
      GO TO (451,452), ITT
  451 CALL HADD (A, B)
      GO TO 455
  452 CALL COST (A, B, -1)
  455 CONTINUE
      DO 460 J=1,M
      IF (B(J).LT.MINP) B(J) = MINP
      IF (B(J).GT.MAXP) B(J) = MAXP
      K=(NH-1)*M+J
  460 D(I,K) = B(J)
  900 CONTINUE
C
C     AFTER A 16 X NPL BLOCK OF DATA HAS BEEN RECONSTRUCTED, PUT IN
C     BYTE ARRAY AND WRITE ON UNIT NO. NB ONE LINE AT A TIME.
      DO 920 I=1,M
      DO 910 J=1,NPL
      IOUT(J)=D(I,J)+0.5
      ISUM2 = ISUM2 + IOUT(J)
  910 CONTINUE
      WRITE(NB) IOUT
  920 CONTINUE
  950 CONTINUE
C
C     COMPUTE AVERAGE BIT RATE, MEANS OF ORIGINAL, RECONSTRUCTED IMAGES
      ITCNT=ITCNT/FLOAT(NTB)
      PIX=NLINE*NPL
      AMEAN1 = ISUM1 / PIX
      AMEAN2 = ISUM2 / PIX
```

```
C
      WRITE (6,40) NPL, NLINE, NB, MAXP, MINP, FIXB
      WRITE(6,60) ITCNT
      WRITE(6,61) AMEAN1, AMEAN2
      RETURN
C
   34 FORMAT(22X,'ADAPTIVE TRANSFORM CODING USING TWO-DIMENSIONAL HADAMA
     .RD TRANSFORM'//)
   35 FORMAT(23X,'ADAPTIVE TRANSFORM CODING USING TWO-DIMENSIONAL COSINE
     . TRANSFORM'//)
   40 FORMAT(' NPL = ',I4,20X,'NLINE = ',I4,10X,'BAND NO.',I2//
     .' MAXP =',F5.1,19X,'MINP =',F5.1,17X,'FIXB =',F5.1/)
   60 FORMAT(' AVERAGE BIT RATE = ',F8.3,/)
   61 FORMAT(' MEAN OF ORIGINAL IMAGE =',F8.3,26X,'MEAN OF RECONSTRUCTED
     . IMAGE =',F8.3//)
      END
      SUBROUTINE MAP(A,B)
C
C     MAP CONVERTS A 16 X 16 COEFFICIENT MATRIX INTO A 1-D ARRAY IN A
C     ZIGZAG SEQUENCE.
C
C        A - INPUT 16 X 16 BLOCK MATRIX
C        B - OUTPUT 256 X 1 ARRAY IN ZIGZAG FORMAT
C*************************************************************************
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      DIMENSION A(16,16), B(256)
      B(1)=A(1,1)
      B(2)=A(1,2)
      B(3)=A(2,1)
      K=4
      L=3
C
C     THIS PART CONVERTS THE UPPER TRIANGLE OF MATRIX
  4   CONTINUE
      J=1
      I=L
   10 CONTINUE
      B(K)=A(I,J)
      K=K+1
        IF(K.GT.136) GO TO 30
      I=I-1
      J=J+1
        IF(I.GE.1) GO TO 10
      J=L+1
      I=1
   20 CONTINUE
      B(K)=A(I,J)
      I=I+1
      J=J-1
      K=K+1
        IF(K.GT.136) GO TO 30
        IF(J.GE.1) GO TO 20
      L=L+2
      GO TO 4
   30 CONTINUE
C
C     THIS PART CONVERTS THE LOWER TRIANGLE OF MATRIX
      L=2
   34 CONTINUE
      J=L
      I=16
   40 CONTINUE
      B(K)=A(I,J)
      K=K+1
      IF(K.GT.256) GO TO 60
      I=I-1
      J=J+1
        IF(J.LE.16) GO TO 40
      I=L+1
      J=16
   50 CONTINUE
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
      B(K)=A(I,J)
      I=I+1
      J=J-1
      K=K+1
        IF(K.GT.256) GO TO 60
        IF(I.LE.16) GO TO 50
      L=L+2
      GO TO 34
   60 CONTINUE
      RETURN
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      ENTRY UNMAP (A, B)
C
C     UNMAP CONVERTS THE 1-D COEFFICIENT ARRAY IN A ZIGZAG FORMAT TC
C     A 16 X 16 BLOCK MATRIX
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      A(1,1)=B(1)
      A(1,2)=B(2)
      A(2,1)=B(3)
      K=4
      L=3
C
C     THIS PART RECONSTRUCTS THE UPPER TRIANGLE OF CCEFFICIENTS
  100 CONTINUE
      J=1
      I=L
  110 CONTINUE
      A(I,J)=B(K)
      K=K+1
        IF (K.GT.136) GO TO 130
      I=I-1
      J=J+1
        IF (I.GE.1) GO TC 110
      J=L+1
      I=1
  120 CONTINUE
      A(I,J)=B(K)
      I=I+1
      J=J-1
      K=K+1
        IF (K.GT.136) GO TO 130
        IF (J.GE.1) GC TO 120
      L=L+2
      GO TO 100
C
C     THIS PART RECONSTRUCTS THE LOWER TRIANGLE OF CCEFFICIENTS
  130 CONTINUE
      L=2
  134 CONTINUE
      J=L
      I=16
  140 CONTINUE
      A(I,J)=B(K)
      K=K+1
        IF (K.GT.256) GO TO 160
      I=I-1
      J=J+1
        IF (J.LE.16) GO TO 140
      I=L+1
      J=16
  150 CONTINUE
      A(I,J)=B(K)
      I=I+1
      J=J-1
      K=K+1
        IF (K.GT.256) GO TO 160
        IF (I.LE.16) GO TO 150
      L=L+2
      GO TO 134
  160 CONTINUE
      RETURN
      END
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```
      SUBROUTINE HADD(SPACE,HAD)
C
C     HADD DOES A FAST HADAMARD TRANSFORMATION ON A 16 X 1 ARRAY
C     THE MATRIX IS NORMALIZED SO THAT THE FORWARD AND INVERSE
C     MAPPINGS ARE THE SAME.
C
C        SPACE = INPUT ARRAY
C        HAD   = TRANSFORMED OUTPUT ARRAY
C.......................................................................
C***********************************************************************
C
      REAL      SPACE(16),BLOCK5(8),BLOCK6(4),BLOCK7(2),IPGM(8),HAD(16)
C
      XN=4.
      IPGM(1)=1
        DO 30 N=2,4
        IDISPL=2**(N-2)+1
        IDELT=0
   20   IPGM(IDISPL+IDELT)=N
        IDELT=IDELT+2**(N-1)
        IF(IDELT+IDISPL-8) 20,20,30
   30   CONTINUE
C
        DO 150 J=1,8
        L=J+J
        I=IPGM(J)
        GO TO (50,200,220,240),I
   50     DO 60 I=1,8
          K=I+I
   60     BLOCK5(I)=SPACE(K-1)+SPACE(K)
   70       DO 80 I=1,4
            K=I+I
   80       BLOCK6(I)=BLOCK5(K-1)+BLOCK5(K)
   90         DO 100 I=1,2
              K=I+I
  100         BLOCK7(I)=BLOCK6(K-1)+BLOCK6(K)
  120   HAD(L-1)=(BLOCK7(1)+BLOCK7(2))/XN
  150   HAD(L) = (BLOCK7(1)-BLOCK7(2))/XN
      RETURN
C
  200 BLOCK7(1)=BLOCK6(1)-BLOCK6(2)
      BLOCK7(2)=BLOCK6(4)-BLOCK6(3)
      GO TO 120
  220   DO 230 I=1,4,2
        K=I+I
        BLOCK6(I)=BLOCK5(K-1)-BLOCK5(K)
  230   BLOCK6(I+1)=BLOCK5(K+2)-BLOCK5(K+1)
      GO TO 90
  240   DO 250 I=1,8,2
        K=I+I
        BLOCK5(I)=SPACE(K-1)-SPACE(K)
  250   BLOCK5(I+1)=SPACE(K+2)-SPACE(K+1)
      GO TO 70
      END
      SUBROUTINE COST (INPUT, COSOUT, ITYPE)
C
C GENERALIZED FAST COSINE TRANSFORM ROUTINE
C THERE IS A CHECK MADE SO THE TABLE IS ONLY CALCULATED THE FIRST TIME
C MAXIMUM SIZE ARRAY = 256 ELEMENTS
C
C INPUT = INPUT ARRAY
C COSOUT= COSINE TRANSFORMED ARRAY
C ITYPE:   1= FORWARD TRANSFORM
C         -1= INVERSE TRANSFORM
C ISIZE = NUMBER OF ELEMENTS IN INPUT/OUTPUT ARRAY
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C*****************************************************************************
C
      REAL INPUT(1)
      DIMENSION COSOUT(1), COSINE(15), SINE(15), OUTPUT(64), S(15)
      LOGICAL FIRST /.TRUE./
      DATA ISIZE, MM /16, 5/
C
      IF (.NOT.FIRST) GO TO 951
      ISIZEF=ISIZE*4
      SQRT2=SQRT(2.0)
C
C COSINE TRANSFORM TABLE GENERATION
      Y=3.1415927/2.0/FLOAT(ISIZE)
      CC=COS(Y)
      SC=SIN(Y)
      COSINE(1)=CC
      SINE(1)=SC
      JJ=ISIZE-2
      DO 950 I=1,JJ
      COSINE(I+1)=COSINE(I)*CC - SINE(I)*SC
950   SINE(I+1)=SINE(I)*CC + COSINE(I)*SC
      CALL MDFT (OUTPUT, MM, 1, S, 0, IFERR)
      FIRST = .FALSE.
C
C BRANCH TO DO FORWARC OR INVERSE TRANSFORM
  951 IF (ITYPE.EQ.-1) GO TO 888
C
C FORWARD TRANSFORM
      OUTPUT(1)=INPUT(1)
      OUTPUT(2)=0.0
      DO 900 I=2,ISIZE
      OUTPUT(I*2-1)=INPUT(I)
      OUTPUT(ISIZEF+3-I*2)=0.0
      OUTPUT(ISIZEF+4-I*2)=0.0
900   OUTPUT(I*2)=0.0
      OUTPUT(2*ISIZE+1)=0.
      OUTPUT(2*ISIZE+2)=0.
C
      CALL MDFT (OUTPUT, MM, 1, S, 2*ITYPE, IFERR)
C
      COSOUT(1) = OUTPUT(1)*SQRT2
      DO 810 I=2,ISIZE
810   COSOUT(I) = 2.0*(OUTPUT(2*I-1)*COSINE(I-1)-OUTPUT(2*I)*SINE(I-1))
      RETURN
```

```
C
C INVERSE TRANSFORM
888    SUM=INPUT(1)/SQRT2
       CT=SUM*(1.0-1.0/SQRT2)/FLOAT(ISIZE)
C
       OUTPUT(1)=INPUT(1)
       OUTPUT(2)=0.0
       DO 910 I=2,ISIZE
       OUTPUT(2*I-1)=          INPUT(I)*COSINE(I-1)
       OUTPUT(ISIZEF+3-I*2)=INPUT(I)*COSINE(I-1)
       OUTPUT(ISIZEF+4-I*2)=INPUT(I)*SINE(I-1)
910    OUTPUT(2*I)=         -(INPUT(I)*SINE(I-1))
       OUTPUT(2*ISIZE+1)=0.
       OUTPUT(2*ISIZE+2)=0.
C
       CALL MDFT (OUTPUT, MM, 1, S, 2*ITYPE, IFERR)
C
       DO 811 I=1,ISIZE
811    COSOUT(I) = OUTPUT(2*I-1) + CT
       RETURN
       END
```

```
      SUBROUTINE MDFT(A,MC,ND,S,IFS,IFERR)
C
C          MDFT, MULTI-DIMENSIONAL FINITE FOURIER TRANSFORM
C
C     MDFT TOGETHER WITH MFORT IS A MODIFICATION OF FORT
C     A SUBROUTINE SUPPLIED BY J.W. COOLEY OF IBM. FORT
C     COMPUTES ONE-DIMENSIONAL FOURIER TRANSFORMS, MDFT-
C     MFORT GIVES TRANSFORMS IN UP TO SIX DIMENSIONS
C
C     A IS A COMPLEX ARRAY WITH DIMENSION A(NN(1),....,NN(ND)),
C     WHERE NN(K)=2**MC(K).  A IS TO BE SET BY THE USER.
C
C     MC IS A VECTOR SET BY USER. 0.LT.MC(K).LE.13 FOR
C     K=1,2,....,ND.  IN ADDITION MC(1)+MC(2)+...+MC(ND).LE.13.
C     (IF IFS=0 THE 13 IN THE TWO ABOVE COMMENTS CAN BE REPLACED BY 14.)
C     IN THE COMMENTS WHICH FOLLOW, M=MAX(MC(K)).
C
C     ND IS THE DIMENSION OF THE COMPLEX ARRAY A.  ND IS
C     SUPPLIED BY THE USER.
C
C     S IS A VECTOR S(J)= SIN(2*PI*J/NP ), J=1,2,.....,NP/4-1,
C     WHERE NP=MAX(NN(K)).  S IS COMPUTED BY THE PROGRAM.
C
C     IFS IS A PARAMETER TO BE SET BY USER AS FOLLOWS-
C     IFS=0 TO SET NP=2**M AND SET UP SINE TABLE.
C
C     IFS=1 TO SET N=NP=2**M, SET UP SIN TABLE, AND DO FOURIER
C     SYNTHESIS.  THE ARRAY A(J(1),J(2),....,J(ND)) IS REPLACED BY
C     X(J(1),J(2),....,J(ND))= SUM OVER 0.LE.K(1).LE.NN(1)-1,
C     0.LE.K(2).LE.NN(2)-1,....,0.LE.K(ND).LE.NN(ND)-1  OF
C     A(K(1),K(2),....,K(ND))*(EXP(2*PI*J(1)*K(1)*I/NN(1)))*
C     (EXP(2*PI*J(2)*K(2)*I/NN(2)))*....*(EXP(2*PI*J(ND)*K(ND)*I/NN(ND)))
C     0.LE.J(1).LE.NN(1)-1,0.LE.J(2).LE.NN(2)-1,....,
C     0.LE.J(ND).LE.NN(ND)-1, WHERE I=SQRT(-1).
C     THE X'S ARE STORED WITH RE X(J(1),J(2),....,J(ND)) IN CELL
C     1+2*(J(1)+J(2)*NN(1)+J(3)*NN(1)*NN(2)+...
C     +J(ND)*NN(1)*NN(2)*...*NN(ND-1)),  AND
C     IM X(J(1),....,J(ND)) IN THE CELL FOLLOWING
C     RE X(J(1),....,J(ND)).
C     THE  A'S  ARE STORED IN THE SAME MANNER.
C
C     IFS=-1   TO SET  N=NP=2**M,SET UP SIN TABLE, AND DO FOURIER
C     ANALYSIS,TAKING THE INPUT ARRAY A AS X AND
C     REPLACING IT BY THE A  SATISFYING THE ABOVE FOURIER SERIES.
C
C     IFS=+2 TO DO FOURIER SYNTHESIS ONLY, WITH A PRE-COMPUTED S.
C
C     IFS=-2 TO DO FOURIER ANALYSIS ONLY, WITH A PRE-COMPUTED S.
C
C     IFERR   IS SET BY PROGRAM TO-
C     =0 IF NO ERROR DETECTED.
C     =1 IF THE MC(K)'S DO NOT SATISFY THE CONDITIONS ABOVE,
C        OR ND DOES NOT SATISFY 1.LE.ND.LE.6.
C     =-1 WHEN IFS=1,0,OR -1 AND THE S TABLE NEED NOT BE COMPUTED.
C     =-2 WHEN IFS=2 OR -2 AND THE S TABLE NEED BE COMPUTED.
```

```
C
C     AS STATED ABOVE, MC(1)+MC(2)+...+MC(ND).LE.13.IF THE
C     COMPUTER USED HAS A GREATER STORAGE CAPACITY THAN THE
C     IBM 7094 THIS MAXIMUM MAY BE INCREASED BY REPLACING
C     13 IN STATEMENT 6 BELOW WITH M=LOG2 N,WHERE N IS
C     THE MAXIMUM NUMBER OF COMPLEX NUMBERS ONE CAN STORE
C     IN HIGH SPEED CORE.  THE DIMENSION OF KE MUST BE SET
C     EQUAL TO M+1 IN BOTH MDFT AND MFORT. THE DO LOOP
C     JUST BEFORE STATEMENT 40 MUST EXTEND TO M-1 INSTEAD OF 12.
C     THE 14 IN STATEMENT 105 MUST BE CHANGED TO M+1.
C     IN MFORT ONE MUST CHANGE THE EQUIVALENCE STATEMENTS FOR
C     THE KE'S AND ADD MORE DO STATEMENTS TO THE BINARY SORT
C     JUST ABOVE STATEMENT 28.
C
      DIMENSION A(1),B(1),MC(1)
      DIMENSION NF(7),NN(6)
      DIMENSION KE(14)
      EQUIVALENCE (KE(1),JC)
      DATA NF(1)/2/, NPD/0/
C
C     NOTE THAT THE NAMED COMMON,CFORTC, IS USED FOR
C     COMMUNICATION BETWEEN MDFT AND MFORT.
      COMMON/CFORTC/M,N,NT,KS,KS2,KST,KE
C
      NDD=ND
      IF (NDD) 110,110,2
 2    IF(ND.GT.6) GO TO 110
      IFSS=IFS
      MS=0
      KM=1
      DO 10 K=1,NDD
      M=MC(K)
      IF(M) 110,110,4
 4    MS=MS+M
      IF(MS.GT.13) GO TO 105
 8    NN(K)=2**M
      NF(K+1)=NF(K)*NN(K)
 10   IF(NN(K).GT.NN(KM)) KM=K
      IFERR=0
      IF (IABS(IFSS).LT.2) GO TO 160
      IF(NPD.LT.NN(KM)) GO TO 150
 15   NTOT2=NF(NDD+1)
      IF(IFSS) 20,110,30
C
C     DOING FOURIER ANALYSIS SO DIVIDE BY NN(1)*NN(2)*...*NN(ND) AND
C     CONJUGATE.
 20   FN=NTOT2/2
      DO 25 I=1,NTOT2,2
      A(I)=A(I)/FN
 25   A(I+1)=-A(I+1)/FN
C
C     BEGINNING OF LOOP FOR COMPUTING MULTIPLE SUM
 30   DO 50 K=1,NDD
      M=MC(K)
      N=NN(K)
      KS=NF(K)
      KS1=KS-1
      KS2=2*KS
      KST=KS+2
      KE(1)=NF(K+1)
      DO 35 L=2,M
 35   KE(L)=KE(L-1)/2
      DO 40 L=M,12
 40   KE(L+1)=KS
      DO 50 J=1,NTOT2,JD
      KSS=J+KS1
      DO 50 I=J,KSS,2
 50   CALL MFORT(A(I),S)
C     END OF LOOP FOR COMPUTING MULTIPLE SUM
C
 70   IF(IFSS) 75,110,100
C
C     DOING FOURIER ANALYSIS, REPLACE A BY CONJUGATE.
 75   DO 80 I=2,NTOT2,2
 80   A(I)=-A(I)
      GO TO 100
 105  IF((IFS.EQ.0).AND.(MS.EQ.14)) GO TO 8
 110  IFERR=1
      GO TO 100
 150  IFERR=-2
 160  NPD=NN(KM)
      M=MC(KM)
      IF (NP.GE.NPD) IFERR=-1
C
C     MAKE TABLE OF S(J)=SIN(2*PI*J/NP),J=1,2,...,NT-1,NT=NP/4
 200  NP=NPD
      NT=NP/4
      MT=M-2
      IF(MT) 260,260,205
 205  THETA=.7853981633974483
C
C     THETA=PI/2**(L+1)    FOR L=1
      JSTEP = NT
C
C     JSTEP = 2**( MT-L+1 ) FOR L=1
      JDIF = NT/2
C
C     JDIF = 2**(MT-L)  FOR L=1
      S(JDIF) = SIN(THETA)
      IF (MT-2)260,220,220
  220 DO 250 L=2,MT
      THETA = THETA/2.
      JSTEP2 = JSTEP
      JSTEP = JDIF
      JDIF = JDIF/2
      S(JDIF)=SIN(THETA)
      JC1=NT-JDIF
      S(JC1)=COS(THETA)
      JLAST=NT-JSTEP2
      IF(JLAST-JSTEP)250,230,230
 230 DO 240 J=JSTEP,JLAST,JSTEP
     JC=NT-J
     JD=J+JDIF
 240 S(JD)=S(J)*S(JC1)+S(JDIF)*S(JC)
 250 CONTINUE
260  IF(IFSS) 15,100,15
100  RETURN
     END
```

```
      SUBROUTINE MFORT(A,B)
C
C         MFORT, MODIFIED VERSION OF FORT FOR USE
C         AS SUBROUTINE BY MDFT
C
      DIMENSION A(1),B(1)
      DIMENSION KE(14)
      COMMON/CFORTC/M,N,NT,KS,KS2,KST,KE
      EQUIVALENCE (KE(13),K1),(KE(12),K2),(KE(11),K3),(KE(10),K4)
      EQUIVALENCE (KE( 9),K5),(KE( 8),K6),(KE( 7),K7),(KE( 6),K8)
      EQUIVALENCE (KE( 5),K9),(KE(4),K10),(KE(3),K11),(KE(2),K12)
      EQUIVALENCE (KE(1),K13),(KE(1),N2)
C
C     SCRAMBLE A, BY SANDE'S METHOD
C     NOTE EQUIVALENCE OF KL AND KE(14-L)
C     BINARY SORT-
      IJ=2
      DO 30 J1=2,K1,KS
      DO 30 J2=J1,K2,K1
      DO 30 J3=J2,K3,K2
      DO 30 J4=J3,K4,K3
      DO 30 J5=J4,K5,K4
      DO 30 J6=J5,K6,K5
      DO 30 J7=J6,K7,K6
      DO 30 J8=J7,K8,K7
      DO 30 J9=J8,K9,K8
      DO 30 J10=J9,K10,K9
      DO 30 J11=J10,K11,K10
      DO 30 J12=J11,K12,K11
      DO 30 JI=J12,K13,K12
      IF(IJ-JI)28,30,30
   28 T=A(IJ-1 )
      A(IJ-1)=A(JI-1)
      A(JI-1)=T
      T=A(IJ)
      A(IJ)=A(JI)
      A(JI)=T
 30   IJ=IJ+KS
C
C     SPECIAL CASE- L=1
 36   DO 40 I=2,N2,KS2
      KSI=I+KS
      T=A(I-1)
      A(I-1)=T+A(KSI-1)
      A(KSI-1)=T-A(KSI-1)
      T=A(I)
      A(I)=T+A(KSI)
 40   A(KSI)=T-A(KSI)
      IF (M.LE.1) GO TO 1
C
C     SET FOR L=2
 50   LEXP1=KS
C
C     LEXP1=KS*2**(L-2)
      LEXP=4*LEXP1
```

```
C
C     LEXP=KS*2**L
      NPL=NT
C
C     NPL=NT*2**(2-L)
C     NT=NP/4,NP IS DEFINED IN COMMENTS IN MDFT.
      DO 130 L=2,M
C
C     SPECIAL CASE- J=0
      DO 80 I=2,N2,LEXP
      I1=I + LEXP1
      I2=I1+ LEXP1
      I3 =I2+LEXP1
      T=A(I-1)
      A(I-1) = T +A(I2-1)
      A(I2-1) = T-A(I2-1)
      T =A(I)
      A(I) = T+A(I2)
      A(I2) = T-A(I2)
      T= -A(I3)
      TI = A(I3-1)
      A(I3-1) = A(I1-1) - T
      A(I3   ) = A(I1 )   - TI
      A(I1-1) = A(I1-1) +T
   80 A(I1)   = A(I1   )  +TI
      IF(L-2) 120,120,90
   90 KLAST=N2-LEXP
      JJ=NPL
C
      DO 110 J=KST,LEXP1,KS
      NPJJ=NT-JJ
      UR=S(NPJJ)
      UI=S(JJ)
      ILAST=J+KLAST
C
      DO 100 I= J,ILAST,LEXP
      I1=I+LEXP1
      I2=I1+LEXP1
      I3=I2+LEXP1
      T=A(I2-1)*UR-A(I2)*UI
      TI=A(I2-1)*UI+A(I2)*UR
      A(I2-1)=A(I-1)-T
      A(I2  )=A(I   ) - TI
      A(I-1) =A(I-1)+T
      A(I)   =A(I)+TI
      T=-A(I3-1)*UI-A(I3)*UR
      TI=A(I3-1)*UR-A(I3)*UI
      A(I3-1)=A(I1-1)-T
      A(I3)   =A(I1  )-TI
      A(I1-1)=A(I1-1)+T
  100 A(I1)   =A(I1)   +TI
C     END OF I LOOP
C
  110 JJ=JJ+NPL
C     END OF J LOOP
```

```
C
  120 LEXP1=2*LEXP1
      LEXP = 2*LEXP
  130 NPL=NPL/2
C     END OF L LOOP
C
    1 RETURN
      END
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

COMPRESSION BY A HYBRID TECHNIQUE

I. <u>NAME</u>

HYBRDC

II. <u>DESCRIPTION</u>

HYBRDC combines a transform system and an adaptive DPCM (differential pulse code modulation) system to compress an image. This hybrid method consists of an orthogonal transformation (HADAMARD or COSINE) in the vertical direction followed by a DPCM in the horizontal direction. The resultant coefficients are adaptively quantized and the image reconstructed by performing inverse transformations. The input image is divided into blocks consisting of 16 records so that only a 16*m row portion of the image will be processed. The bit rate for each row of the block remains constant and is input by the user.

III. <u>CALLING SEQUENCE</u>

CALL HYBRDC (IV, IOUT, D, D, DIF) - Cosine Transformation

CALL HYBRDH (IV, IOUT, D, D, DIF) - Hadamard Transformation

where IV, IOUT, D and DIF are arrays. The array dimensions required are:

```
IV (4, NPL)    }
IOUT (NPL)     } bytes

D (16, NPL)    }
DIF (NPL)      } words
```

where NPL is the number of pixels per scan line.

IV. <u>INPUT/OUTPUT</u>

1. INPUT

The input data should be on logical unit 10 as a data set consisting of NLINE records and NPL pixels per record with four channels per pixel. Each data value has a length of one byte.

Initial parameters needed to process the image are transferred from a driver program through the following common statement:

COMMON NB, NLINE, BIT, NPL

where

o NB is the band currently being processed (integer)

- NLINE is the number of records in the input image which will be processed. It must be an integer multiple of 16. (integer)
- BIT contains the bit rates used to quantize the DPCM differences for each row of a block of 16 x NPL pixels (integer, 16 locations)
- NPL is the number of pixels per record which will be processed (integer)

The bit rates to be input for each MSS spectral band for 5 compression ratios are given in the following table. The rate is specified for each of the 16 transform coefficients.

BIT ASSIGNMENT FOR HYBRID DPCM QUANTIZER

BIT RATE	BAND NO.	COEFFICIENT (ROW) NUMBER															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
4 bits/ sample	1	7	5	5	5	4	4	4	4	3	3	3	3	3	3	3	2
	2	7	5	5	5	5	5	4	4	4	4	4	4	4	4	4	4
	3	7	5	5	5	5	5	5	4	4	4	4	4	3	3	3	4
	4	7	5	5	4	4	4	4	3	3	3	3	2	2	2	2	2
3 bits/ sample	1	6	4	4	4	3	3	3	3	2	2	2	2	2	2	2	1
	2	6	4	4	4	4	4	3	3	3	3	3	3	3	3	3	2
	3	6	4	4	4	4	4	4	3	3	3	3	3	2	2	2	2
	4	6	4	4	3	3	3	3	2	2	2	2	1	1	1	1	1
2 bits/ sample	1	5	3	3	3	2	2	2	2	1	1	1	1	1	1	1	0
	2	5	3	3	3	3	3	2	2	2	2	2	2	2	2	2	1
	3	5	3	3	3	3	3	3	2	2	2	2	2	1	1	1	1
	4	5	3	3	2	2	2	2	1	1	1	1	0	0	0	0	0
1 bit/ sample	1	4	3	2	2	2	1	1	1	0	0	0	0	0	0	0	0
	2	4	3	3	2	2	2	2	1	1	1	0	0	0	0	0	0
	3	4	3	2	2	2	1	1	1	0	0	0	0	0	0	0	0
	4	4	2	2	1	1	1	0	0	0	0	0	0	0	0	0	0

BIT ASSIGNMENT FOR HYBRID DPCM QUANTIZER (CONT.)

BIT RATE	BAND NO.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0.5 bit/ sample	1	3	2	1	1	1	0	0	0	0	0	0	0	0	0	0	0
	2	3	2	1	1	1	1	0	0	0	0	0	0	0	0	0	0
	3	3	2	1	1	1	0	0	0	0	0	0	0	0	0	0	0
	4	3	2	1	1	1	0	0	0	0	0	0	0	0	0	0	0

2. OUTPUT

The input parameters and the means of the original and reconstructed images are printed. The reconstructed image for band NB is written onto unit NB.

V. DESCRIPTION OF SUBROUTINES

The storage requirements and the functions of the non-system subroutines used are given in the following table.

DESCRIPTION OF SUBROUTINES FOR HYBRID
DATA COMPRESSION

SUBROUTINE NAME (ENTRY POINTS)	STORAGE (BYTES)	FUNCTION
HYBRDC (HYBRDH)	3748	Read 16 line blocks from input data set, call Hadamard or Cosine transform on columns, call DPCM on rows of coefficients, reconstruct the image by calling for inverse transformations
DSQ (QUAN)	908	Perform mapping to obtain uniform distribution of co-efficients, quantize to specified number of levels, do inverse to reconstruct coefficient values
HADD	1110	Hadamard transformation of a string of 16 pixels
COST MDFT MFORT	1844 1826 1870	Cosine transformation of a string of 16 pixels

The linkages of the subroutines are given in the following table.

LINKAGES OF SUBROUTINES FOR HYBRID DATA COMPRESSION

HYBRDC (HYBRDH)	HADD
	COST
	DSQ
	QUAN
COST	MDFT
MDFT	MFORT

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

VI. PERFORMANCE SPECIFICATIONS

1. STORAGE REQUIREMENTS

The program requires 60K bytes of storage when operating on a LACIE sample segment.

2. EXECUTION TIME

The processing speed is highly dependent on the type of transform applied and somewhat dependent on the bit rate, being slower at higher bit rates due to the quantization of a greater number of coefficients. Average speeds from several computer runs are given in the following table.

PROCESSING SPEEDS FOR HYBRID COMPRESSION
(4 band pixels/second)

BIT RATE	COSINE SPEED	HADAMARD SPEED
1/2	283	849
1	263	696
2	225	534
3	222	509
4	225	511

VII. METHOD

A one-dimensional transform method and a one-dimensional DPCM are combined in HYBRDX to perform data compression. Initially the input image is divided into blocks consisting of 16 rows of pixels. Correlation factors for the coefficients in each row of the block are assumed to be as follows:

COEFFICIENT NUMBERS	CORRELATION FACTOR
1, 2, 3, 4	15/16
5, 6, 7, 8	3/4
9, 10, ..., 16	1/2

A one-dimensional orthogonal transformation (Hadamard or Cosine) is performed on the columns of the 16 x NPL block of data. Following this transformation, a one-dimensional DPCM is performed on each row of the block. This is accomplished by first computing the mean of a row and the deviation of each point in the row from the mean. Next, an error is predicted for each point by computing the difference between the deviation of that point and the previous deviation multiplied by the correlation factor for that row.

After the DPCM is complete, the variance of the predicted errors is computed for a row. Using this variance, the scaling factors used to map the coefficients in the row into a uniform probability density function are computed.

The difference between the deviation of a point from the mean of the row and the predicted error for the point is quantized. Using these quantized values, the transformed coefficients are reconstructed. The inverse of the orthogonal transformation is performed to complete the reconstruction of the image. The bit rate for each row of the block remains constant and is input by the user. A zero bit rate is accepted as a valid input.

A flow chart of the hybrid compression method is given in the following figure.

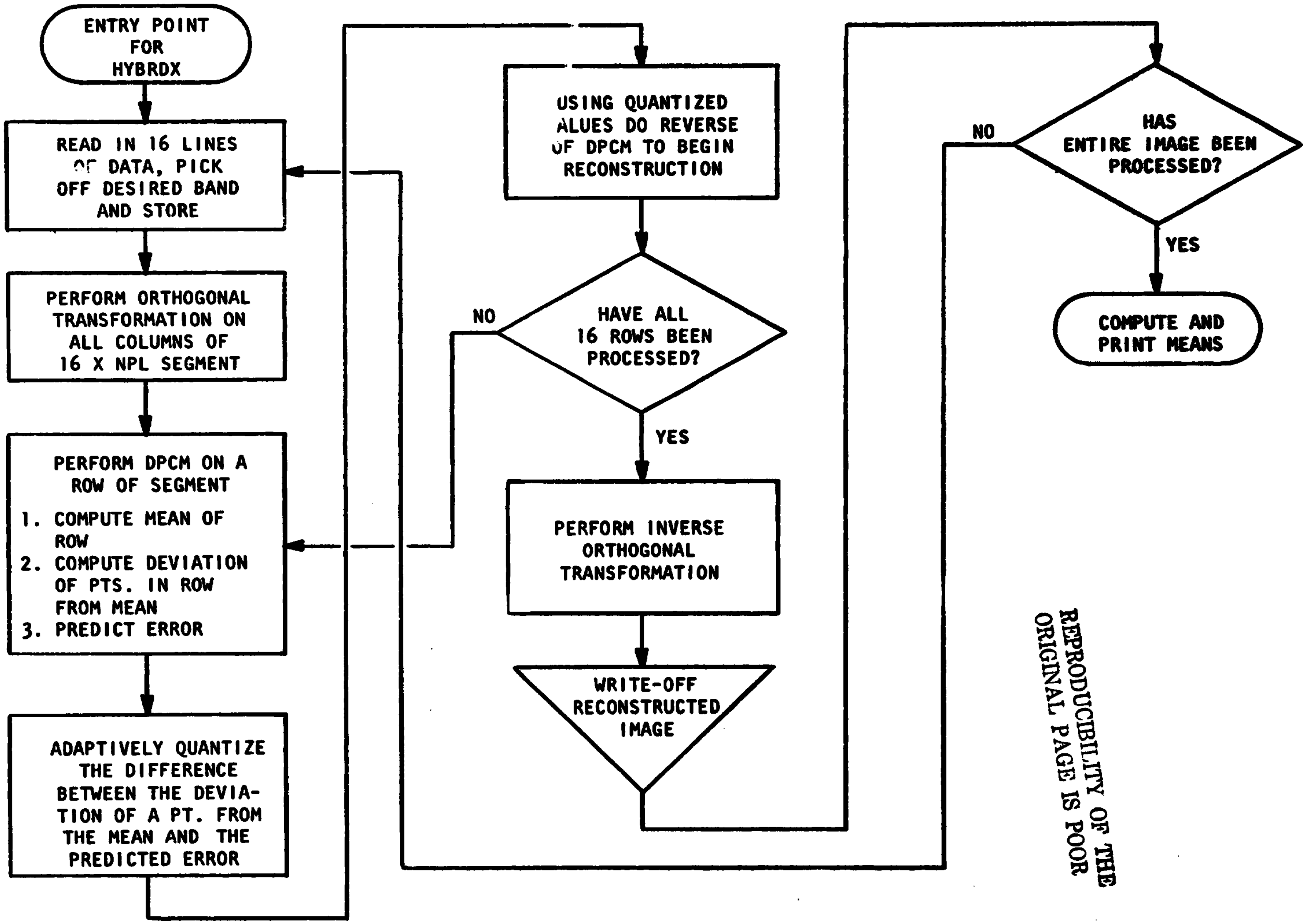

ENTRY POINT FOR HYBRDX
READ IN 16 LINES ^r DATA, PICK OFF DESIRED BAND AND STORE
PERFORM ORTHOGONAL TRANSFORMATION ON ALL COLUMNS OF 16 X NPL SEGMENT
PERFORM DPCM ON A ROW OF SEGMENT
1. COMPUTE MEAN OF ROW
2. COMPUTE DEVIATION OF PTS. IN ROW FROM MEAN
3. PREDICT ERROR
ADAPTIVELY QUANTIZE THE DIFFERENCE BETWEEN THE DEVIATION OF A PT. FROM THE MEAN AND THE PREDICTED ERROR
USING QUANTIZED ALUES DO REVERSE ∪F DPCM TO BEGIN RECONSTRUCTION
HAVE ALL 16 ROWS BEEN PROCESSED?
NO
YES
PERFORM INVERSE ORTHOGONAL TRANSFORMATION
WRITE-OFF RECONSTRUCTED IMAGE
HAS ENTIRE IMAGE BEEN PROCESSED?
NO
YES
COMPUTE AND PRINT MEANS

Figure 13. Flow Chart of the Hybrid Compression Method

VIII. COMMENTS

Only a 16*m row (m an integer) segment of the input image will be processed.

IX. TESTS

Reconstructed images were examined using image means, mean squared error, plots and histograms of the reconstructed images, and plots and histograms of difference images. An example of a difference image follows.

X. LISTINGS

Listings of the subroutines follow.

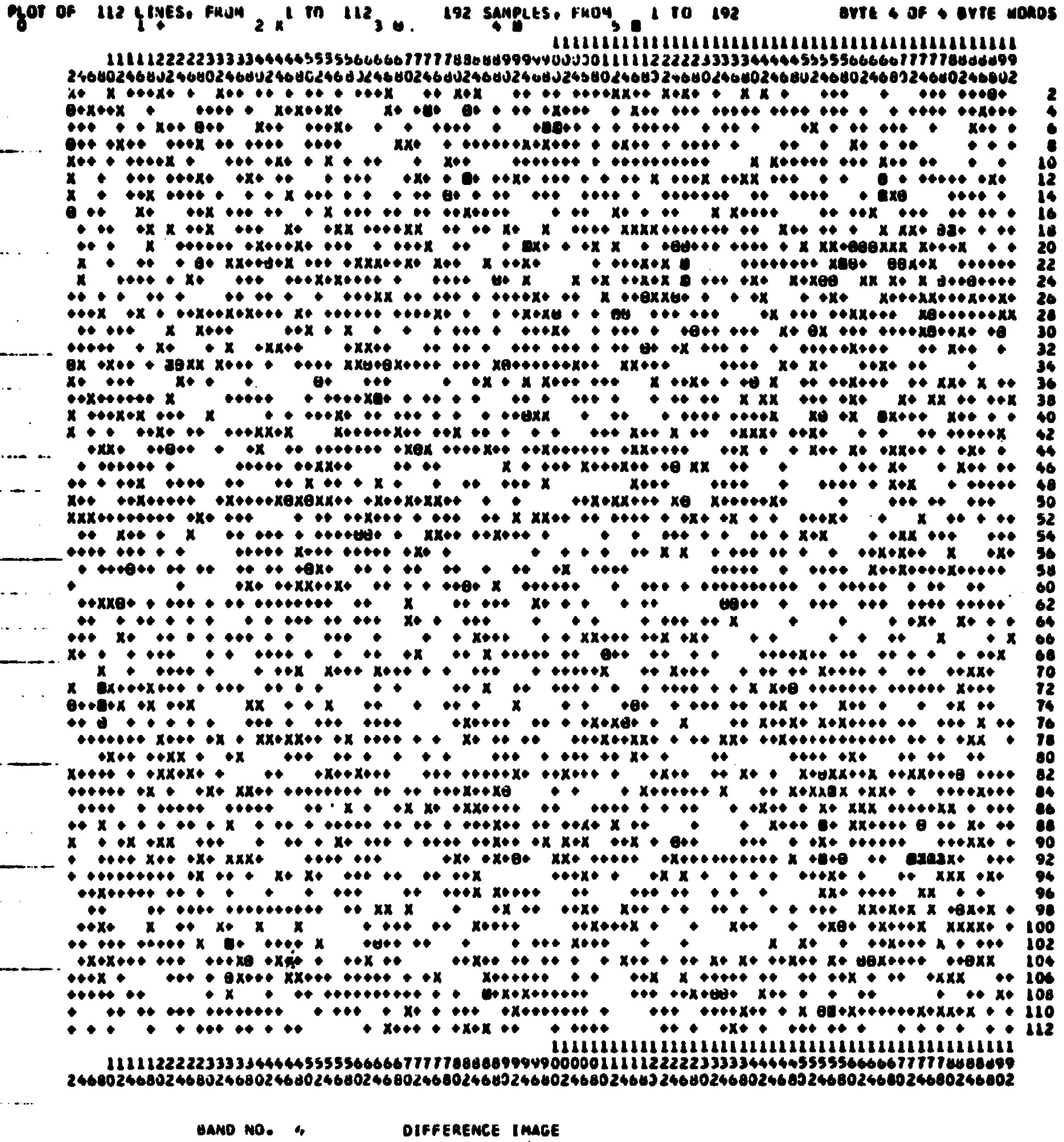


Figure 14. Difference Image |Original - Hybrid Reconstruction|

ORIGINAL PAGE IS
OF POOR QUALITY

```
      SUBROUTINE HYBRDC (IV, IOUT, D, E, DIF)
C
C     SUBROUTINE HYBRID USES A HYBRIC METHOD TO COMPRESS AN IMAGE.
C     THIS METHOD CONSISTS OF A 1-D ORTHOGONAL (HADAMARD OR COSINE)
C     TRANSFORM IN THE VERTICAL DIRECTION FOLLOWED BY A 1-D DPCM IN THE
C     HORIZONTAL DIRECTION.  THE TRANSFORMED COEFFICIENTS ARE
C     ADAPTIVELY QUANTIZED.  USING THESE COEFFICIENTS, THE IMAGE IS
C     RECONSTRUCTED BY DOING INVERSE TRANSFORMATIONS.
C
C        NPL    = NUMBER OF PIXELS PER LINE
C        ITT    = TYPE OF ORTHOGONAL TRANSFORM
C        1 = HADAMARD                      2=COSINE
C        A = INPUT 16 X 1 ARRAY
C        B = TRANSFORMED 16 X 1 OUTPUT ARRAY
C        MINP   = MINIMUM VALUE OF INPUT IMAGE
C        MAXP   = MAXIMUM VALUE OF INPUT IMAGE
C
C     LOGICAL UNITS
C       10 = INPUT PICTURE (BYTES)
C       NB = RECONSTRUCTED PICTURE (BYTES)
C **********************************************************************
C
      DIMENSION IV(4,NPL), IOUT(NPL), D(16,NPL), E(16,NPL), DIF(NPL),
     , A(16), B(16), RHO(16)
      LOGICAL*1 IV, IOUT
      REAL MEAN,MAXP,MINP,MAX(4)/3*127.0,63.0/,MIN(4)/4*0.0/
      INTEGER BIT(16)
      COMMON NB, NLINE, BIT, NPL
      DATA M /16/
C
C     ENTRY FOR COSINE TRANSFORM
      ITT=2
      WRITE(6,45)
      GO TO 10
C
C     ENTRY FOR HADAMARD TRANSFORM
      ENTRY HYBRDH (IV, IOUT, D, E, DIF)
      ITT=1
      WRITE(6,46)
C
C     INITIALIZE COUNTERS, SET IMAGE PARAMETERS
   10 CONTINUE
      ISUM1=0
      ISUM2=0
      NBPV=NLINE/M
      ANPL=NPL
      MAXP = MAX(NB)
      MINP = MIN(NB)
C
C     CHECK FOR INDEX OF FIRST ZERO BIT RATE
C     KZ IS THE POINTER FOR THE LOCATION OF THE FIRST ZERO BIT RATE
          DO 25 I=1,M
          IF ( BIT(I) .NE. 0 ) GO TO 25
          KZ=I
          GO TO 30
   25     CONTINUE
          KZ=M+1
   30     CONTINUE
C
C      STORE CORRELATION FACTORS FOR EACH ROW OF TRANSFORMED DATA
             DO 50 I=1,4
             RHO(I)=15./16.
             RHO(I+4)=3./4.
             RHO(I+8)=0.5
   50        RHO(I+12) =0.5
C
C      READ IN 16 LINES OF DATA, PICK OFF DESIRED BAND AND STORE
C      IN D ARRAY
          DO 500 NV=1,NBPV
             DO 125 I=1,M
             READ(10) IV
               DO 120 J=1,NPL
               IIV = IV(NB,J)
               D(I,J) = IIV
               ISUM1 = ISUM1 + IIV
  120          CONTINUE
  125        CONTINUE
C
C      PERFORM EITHER A HADAMARD OR COSINE TRANSFORM ON COLUMNS OF D
C      DEPENDING ON THE VALUE OF ITT.
             DO 160 J=1,NPL
               DO 130 I=1,M
  130          A(I)=D(I,J)
             GO TO (131,132), ITT
  131        CALL HADD (A, B)
             GO TO 135
  132        CALL COST (A, B, 1)
  135        CONTINUE
C
C      D(I,J).......TRANSFORMED DATA
               DO 140 I=1,M
  140          D(I,J)=B(I)
  160        CONTINUE
C
C      DO DPCM ON ROWS OF ARRAY D
C
C      COMPUTE THE MEAN OF EACH ROW OF ARRAY D AND THE
C      DEVIATION OF EACH POINT IN THE ROW FROM THE MEAN
             DO 350 I=1,M
             SUM=0.0
               DO 220 J=1,NPL
               SUM=SUM+D(I,J)
  220          CONTINUE
             MEAN = SUM/ANPL
             IF ( I .GE. KZ ) GO TO 300
C
C      DIF(J).........ZERO CENTERED TRANSFORMED DATA.
               DO 225 J=1,NPL
               DIF(J) = D(I,J) - MEAN
  225          CONTINUE
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```
C
C     E(I,J).......DIFFERENCE BETWEEN THE DEVIATION AND THE
C     PREVIOUS DEVIATION MULTIPLIED BY THE CORRELATION FACTOR
            E(I,1)=0.0
              DO 240 J=2,NPL
  240         E(I,J)=DIF(J)-DIF(J-1)*RHO(I)
C
C     COMPUTE THE VARIANCE OF THE ERROR AND THE SCALING FACTOR
            PA=0.0
              DO 250 J=1,NPL
  250         PA=PA+E(I,J)
            PA=PA/ANPL
            FA=0.0
              DO 260 J=1,NPL
              FA=FA+(E(I,J)-PA)**2
  260         CONTINUE
            SGM=SQRT(FA/ANPL)
            CALL DSO (3.0, SGM)
            LEVEL = 2**(BIT(I)-1)
C
C     QUANTIZE DEVIATION FROM THE MEAN FOR THE FIRST SAMPLE
            AF=DIF(1)
            CALL QUAN (AF, LEVEL, EQ)
            EE = EQ
            E(I,1) = EE + MEAN
C
C     ADAPTIVELY QUANTIZE THE DIFFERENCE BETWEEN THE DEVIATION FROM
C     THE MEAN AND THE PREDICTED VALUE.  ADD THE PREDICTED VALUE TO THE
C     QUANTIZED OUTPUT.  FINALLY, ADD BACK THE MEAN.
              DO 320 J=2,NPL
              AF = DIF(J) - RHO(I)*EE
              CALL QUAN (AF, LEVEL, EQ)
              EE = EQ + RHO(I)*EE
              E(I,J) = EE + MEAN
  320         CONTINUE
            GO TO 350
C
C      OUTPUT FOR CASE OF ZERO BIT RATE
  300       DO 310 I=KZ,M
            DO 310 J=1,NPL
            E(I,J)=MEAN
  310       CONTINUE
  350       CONTINUE
C
C     DO INVERSE 1-D HADAMARD OR COSINE TRANSFORM
            DO 450 J=1,NPL
              DO 410 I=1,M
  410         A(I)=E(I,J)
            GO TO (411,412), ITT
  411       CALL HADD (A, B)
            GO TO 415
  412       CALL COST (A, B, -1)
  415       CONTINUE
C
C     D(I,J).......RECONSTRUCTED  DATA.
```

```
            DO 420 I=1,M
            IF (B(I).LT.MINP) B(I) = MINP
            IF (B(I).GT.MAXP) B(I) = MAXP
  420       D(I,J) = B(I)
  450     CONTINUE
C
          DO 490 I=1,M
            DO 480 J=1,NPL
            IOUT(J)=D(I,J)+0.5
            ISUM2 = ISUM2 + IOUT(J)
  480       CONTINUE
          WRITE(NB) IOUT
  490     CONTINUE
  500   CONTINUE
C
C     COMPUTE MEANS OF ORIGINAL AND RECONSTRUCTED IMAGES
          PIX=NLINE*NPL
      AMEAN1 = ISUM1 / PIX
      AMEAN2 = ISUM2 / PIX
C
      WRITE(6,43) BIT
      WRITE(6,40) NPL,NLINE,NB,MAXP,MINP
      WRITE(6,41) AMEAN1, AMEAN2
      RETURN
C
   40 FORMAT(' NPL =',I5,15X,'NLINE =',I5,15X,'BAND NO.',I2,15X,'MAXP ='
     .,F6.1,15X,'MINP =',F6.1/)
   41 FORMAT (' MEAN OF ORIGINAL IMAGE =',F8.3,26X,'MEAN OF RECONSTRUCTE
     .D IMAGE =',F8.3//)
   43 FORMAT(' BIT RATES ',16I2,/)
   45 FORMAT(20X,'HYBRID METHOD USING A 1-D COSINE AND A 1-D DPCM TRANSF
     .ORM'//)
   46 FORMAT(25X,'HYBRID METHOD USING A 1-D HADAMARD AND A 1-D DPCM TRAN
     .SFORM'//)
      END
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

CLUSTER CODING ALGORITHM

I. **NAME**

CCA

II. **DESCRIPTION**

The data set is partitioned into arrays of a specified size, and each array is clustered into the required number of clusters. The data is reconstructed by replacing the original data with the closest cluster centroid.

III. **CALLING SEQUENCE**

CALL CCA (A, B, NREC, NPIX, NFEAT, LXL, NXN)
where
A, B are work arrays dimensioned (4, NPIX) and (LXL, NPIX, 4) bytes respectively,
NREC, NPIX are the number of records and the number of pixels per record in the data set,
NFEAT is the number of clusters to be found in each block of data,
LXL, NXN are the length (records) and width (samples) of the data blocks.

IV. **INPUT/OUTPUT**

1. INPUT
   The input to this program is a sequential data set on logical unit 10, having NREC records each NPIX 4-band pixels long.

2. OUTPUT
   The output is the reconstructed image, written in the same format as the input.

V. **DESCRIPTION OF SUBROUTINES**

No additional subroutines are required.

VI. <u>PERFORMANCE SPECIFICATIONS</u>

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

1. STORAGE

   The subroutine occupies 12864 bytes of storage.

2. EXECUTION TIME

   For a LACIE sample segment (117 x 196) the execution time on the IBM 360/75 is approximately 30 seconds.

VII. <u>METHOD</u>

The multispectral data is first divided into blocks of the specified size. All of the unique data vectors in the block are then found. Initial cluster centroids are chosen at equal intervals in the table of vectors. Then all samples are assigned to the cluster containing the closest centroid. The centroids of each cluster are then replaced by the center of mass of samples in that cluster. This procedure is repeated for two iterations. The next step is to replace each data vector with the closest cluster centroid. The cluster numbers are stored in an array at location numbers corresponding to the vector table locations. Each block of imagery is then reconstructed by obtaining the cluster numbers from the array.

VIII. <u>COMMENTS</u>

The program is dimensioned for a maximum of 32 clusters per block.

IX. <u>TESTS</u>

The reconstructed imagery has been examined visually and by computing means, variances, and mean squared errors. Data blocks have been printed to verify the occurrence of the correct number of cluster centers.

X. <u>LISTING</u>

The listing of the routine follows.

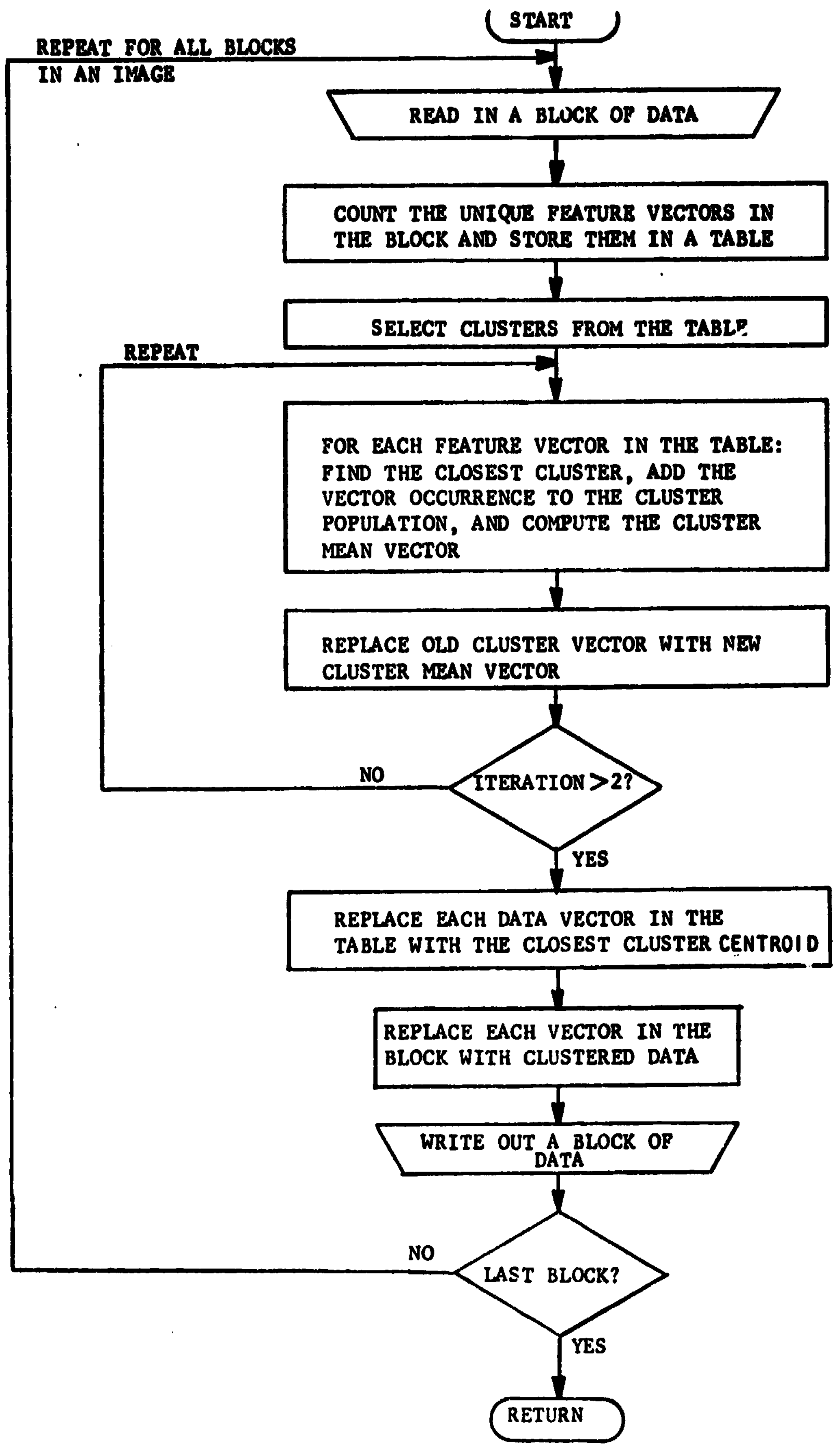

START
REPEAT FOR ALL BLOCKS IN AN IMAGE
READ IN A BLOCK OF DATA
COUNT THE UNIQUE FEATURE VECTORS IN THE BLOCK AND STORE THEM IN A TABLE
SELECT CLUSTERS FROM THE TABLE
REPEAT
FOR EACH FEATURE VECTOR IN THE TABLE: FIND THE CLOSEST CLUSTER, ADD THE VECTOR OCCURRENCE TO THE CLUSTER POPULATION, AND COMPUTE THE CLUSTER MEAN VECTOR
REPLACE OLD CLUSTER VECTOR WITH NEW CLUSTER MEAN VECTOR
NO
ITERATION > 2?
YES
REPLACE EACH DATA VECTOR IN THE TABLE WITH THE CLOSEST CLUSTER CENTROID
REPLACE EACH VECTOR IN THE BLOCK WITH CLUSTERED DATA
WRITE OUT A BLOCK OF DATA
NO
LAST BLOCK?
YES
RETURN

Figure 15. Simplified Flow Diagram for Cluster Coding Algorithm (CCA)

```
      SUBROUTINE CCA (A, B, NREC, NPIX, NFEAT, LXL, NXN)
C
C     CLUSTER CODING ALGORITHM-BREAKS IMAGE INTO LXL BY NXN
C     ARRAYS AND CLUSTERS EACH ARRAY INTO A MAXIMUM OF
C     32 FEATURES, WRITES OUT A NEW TAPE WITH ORIGINAL
C     DATA REPLACED BY CLOSEST MEAN CLUSTER CENTER
C
C     THE INPUT VARIABLES ARE AS FOLLOWS:
C     A=RECORD OF DATA TO BE PUT INTO B(I,J) FORMAT
C     B(I,J)=DATA ARRAY FOR RECORD I, PIXEL J
C     NREC=NUMBER OF RECORDS DESIRED
C     NPIX=NUMBER OF PIXELS DESIRED
C     NFEAT=NUMBER OF FEATURES/ARRAY
C     LXL=ARRAY LENGTH
C     NXN=ARRAY WIDTH
C
      INTEGER A(NPIX), B(LXL,NPIX)
      DIMENSION V(32,4), IV(32), KNT(32), XBAR(32,4), N(500), NPOP(500),
     *MDEX(500), NDEX(500)
      LOGICAL*1 LM(4)
      EQUIVALENCE (IM, LM(1))
      DATA NITR, NVEC, JFAC, IMOD, JMOD /2, 500, 3, 3, 4/
C
      KOUNT=NVEC
      DO 65 KO=1,KOUNT
   65 NDEX(KO)=KO
      WRITE (6,800) NREC, NPIX, NFEAT, LXL, NXN
C
C     NLONG=LENGTH OF SUBIMAGE IN NUMBER OF LXL ARRAYS
C     NWIDE=WIDTH OF SUBIMAGE IN NUMBER OF NXN ARRAYS
C     NRLNG=NUMBER RECORDS LEFT OVER FROM INTEGER MULTIPLE
C     NRWDE=NUMBER PIXELS LEFT OVER FROM INTEGER MULTIPLE
C     IF THERE ARE PIXELS AND/OR RECORDS LEFT OVER,OUTPUT IMAGE
C     WILL BE AN ADDITIONAL ARRAY WIDE AND/OR LONG
      NLONG=NREC/LXL
      NWIDE=NPIX/NXN
      NRLNG=NREC-NLONG*LXL
      NRWDE=NPIX-NWIDE*NXN
      IF (NRLNG.NE.0) NLONG=NLONG+1
      IF (NRWDE.NE.0) NWIDE=NWIDE+1
C
C     START READING IN THE DESIRED DATA
C     NL=THE LENGTH OF THE SUBIMAGE IN LXL ARRAYS
      MXN = LXL
      DO 480 NL=1,NLONG
      IF (NL.EQ.NLONG.AND.NRLNG.NE.0) MXN=NRLNG
C
C     READ IN LXL RECORDS OF DATA (A) AND PUT INTO B
      DO 150 NR=1,MXN
      READ (10) A
      DO 130 NP=1,NPIX
  130 B(NR,NP) = A(NP)
  150 CONTINUE
C
C     THE FIRST STEP IS TO COMPUTE A FOUR-D HISTOGRAM
```

```
C      AND FIND THE MODE
C
C      EACH SUBIMAGE RECORD IS NWIDE NUMBER OF NXN ARRAYS
C      WIDE-MSTART AND MSTOP ARE THE START AND STOP PIXELS
C      IN EACH RECORD FOR EACH ARRAY
       DO 440 NW=1,NWIDE
       MSTOP=NW*NXN
       MSTART= MSTOP-NXN+1
       IF(NW.EQ.NWIDE) MSTOP=NPIX
       TOT=MXN*(MSTOP-MSTART+1)
       DO 160 KC=1,KOUNT
       L=INDEX(KC)
  160  NPOP(L)=0
       KOUNT=0
C
C      LOOP OVER MXN RECORDS
       DO 190 NR=1,MXN
C
       DO 185 NP=MSTART,MSTOP
       IM = M(NR,NP)
C
C      HASH ROUTINE
C      ACCUMULATE FOUR DIMENSIONAL HISTOGRAM
C
C      COMPUTE TABLE LOCATION FROM VECTOR COMPONENTS
       L=0
       DO 172 NC=1,4
       NXX = IM(NC)
  172  L = L*JMOD + MOD(NXX,IMOD)
       L=L*JFAC+1
C
C      CHECK FOR EMPTY TABLE LOCATION
  174  IF(NPOP(L).NE.0)GO TO 177
C
C      HAVE FOUND A NEW VECTOR, INCREMENT VECTOR COUNTER
       KOUNT=KOUNT+1
C
C      SET POPULATION OF NEW VECTOR TO ONE
       NPOP(L)=1
       INDEX(KOUNT)=L
C
C      PUT NEW VECTOR INTO TABLE
       H(L) = IM
       GO TO 180
C
C      CHECK TO SEE IF VECTOR IS IN TABLE
  177  IF (H(L).NE.IM) GO TO 179
C
C      VECTOR IS IN TABLE, INCREMENT POPULATION COUNTER
       NPOP(L)=NPOP(L)+1
       GO TO 180
C
C      VECTOR IS NOT THE SAME AS THE ONE WITH INDEX L
C      TRY THE NEXT INDEX
  179  L=L+1
```

ORIGINAL PAGE IS OF POOR QUALITY

```
C
C     CHECK TO SEE IF INDEX IS LARGER THAN END OF TABLE
C     IF SO, SET INDEX TO ONE AND START AT BEGINNING OF TABLE
      IF(L.GT.NVEC)L=1
      GO TO 174
C
C     REPLACE THE ORIGINAL DATA WITH ITS TABLE LOCATION L
  180 CONTINUE
      B(NR,NP) = L
C
C     RETURN TO NEXT VECTOR
  185 CONTINUE
C
C     RETURN TO NEXT RECORD
  190 CONTINUE
C
C     FIND NFEAT CLUSTERS
      LFEAT = MINO (NFEAT, KOUNT)
      DELTA = FLOAT(KOUNT-1)/FLOAT(LFEAT-1)
      DO 220 MF=1,LFEAT
      ICHEK = 1.0+(MF-1)*DELTA+0.5
      L=NDEX(ICHEK)
      IM = N(L)
      DO 210 NC=1,4
  210 V(MF,NC) = LM(NC)
  220 CONTINUE
C
C     THE NEXT STEP IS TO ITERATE TO IMPROVE INITIAL CLUSTER ESTIMATE
C     KNT(NF) = POPULATION OF CLUSTER NF
C     XBAR(NF,NC)=MEAN VECTOR OF CLUSTER NF
      DO 380 IT=1,NITR
      DO 300 NF=1,LFEAT
      KNT(NF) = 0
      DO 290 NC=1,4
  290 XBAR(NF,NC)=0.0
  300 CONTINUE
C
C     FOR EACH PIXEL VECTOR, FIND THE CLOSEST CLUSTER
      DO 340 KO=1,KOUNT
      L=NDEX(KC)
      IM = N(L)
      XMIN=100000.0
      DO 320 NF=1,LFEAT
      SUM = 0.0
      DO 310 NC=1,4
      XX = LM(NC)
      SUM = SUM + (V(NF,NC)-XX)**2
      IF (SUM.GE.XMIN) GO TO 320
  310 CONTINUE
      XMIN = SUM
      MFEAT=NF
      IF (SUM.LE.1.5) GO TO 325
  320 CONTINUE
C
C     HAVE NOW FOUND CLUSTER THAT IS CLOSEST TO A PIXEL
```

```
C     VECTOR = NEXT UPDATE CLUSTER POPULATION AND COMPUTE
C     CLUSTER MEAN VECTOR
  325 C = KNT(MFEAT)
      KNT(MFEAT) = KNT(MFEAT)+NPOP(L)
      DO 330 NC=1,4
      XX = LM(NC)*NPOP(L)
  330 XBAR(MFEAT,NC)=(C*XBAR(MFEAT,NC)+XX)/KNT(MFEAT)
C
C     RETURN TO NEXT PIXEL
  340 CONTINUE
C
C     REPLACE OLD CLUSTER VECTOR WITH NEW CLUSTER VECTOR
      DO 360 NF=1,LFEAT
      DO 360 NC=1,4
  360 V(NF,NC)=XBAR(NF,NC)
C
C     RETURN FOR NEXT ITERATION
  380 CONTINUE
C
C     AFTER COMPLETING ITERATIONS, FIND THE CLUSTER CLOSEST TO
C     EACH DATA VECTOR IN THE TABLE
      DO 420 KC=1,KOUNT
      L=NDEX(KC)
      IM = N(L)
      XMIN=100000.0
      DO 400 NF=1,LFEAT
      SUM = 0.0
      DO 390 NC=1,4
      XX = LM(NC)
      SUM = SUM + (V(NF,NC)-XX)**2
      IF (SUM.GE.XMIN) GO TO 400
  390 CONTINUE
      XMIN = SUM
      MFEAT=NF
      IF (SUM.LE.1.5) GO TO 410
  400 CONTINUE
  410 CONTINUE
C
C     STORE THE CLUSTER NUMBER IN ARRAY NDEX
      NDEX(L)=MFEAT
C
C     RETURN TO NEXT PIXEL
  420 CONTINUE
C
C     REPLACE DATA VECTORS WITH CLUSTER CENTROIDS
      DO 422 NF=1,LFEAT
      DO 421 NC=1,4
  421 LM(NC) = V(NF,NC) + 0.5
  422 IV(NF) = IM
      DO 435 NR=1,NXN
      DO 430 NP=NSTART,NSTOP
      L = M(NR,NP)
      MFEAT=NDEX(L)
      M(NR,NP) = IV(MFEAT)
  430 CONTINUE
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
  435 CONTINUE
C
C     RETURN TO NEXT DATA BLOCK (NXN ARRAY) ACROSS THE IMAGE
  440 CONTINUE
C
C     AFTER CLUSTERING HAS BEEN COMPLETED ACROSS THE IMAGE,
C     THE CLUSTERED DATA IS WRITTEN OUT LXL RECORDS AT A TIME BY
C     PUTTING B BACK INTO THE A DATA FORMAT
      DO 470 NR=1,MXN
      DO 460 NP=1,NPIX
  460 A(NP) = B(NR,NP)
  470 WRITE (11) A
C
C     LXL RECORDS HAVE BEEN WRITTEN THAT GO ACROSS THE
C     IMAGE, THE NEXT STEP IS TO READ IN LXL MORE DATA
C     RECORDS TO GO DOWN THE IMAGE, THIS IS ACCOMPLISHED BY
C     THE NEXT RETURN,
  480 CONTINUE
      RETURN
C
  800 FORMAT ('1',20X,'CLUSTER CODING ALGORITHM'/21X,24('*')//21X,'RECOR
     .DS USED',I12//21X,'PIXELS USED',I13//21X,'FEATURES/ARRAY',I10//21X
     .,'ARRAY LENGTH (SCANS)',I4//21X,'ARRAY WIDTH (PIXELS)',I4//)
      END
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

VECTOR REDUCTION

I. NAME

VREDUC

II. DESCRIPTION

The number of measurement vectors required to represent an image is reduced by merging the component values to the nearest of a set of equally spaced values (e.g., multiples of 3).

III. CALLING SEQUENCE

Call VREDUC (KNT, KOUNT, INT, NUM, N, NOP, IFEAT, JFAC, IMOD, JMOD)

where

KNT is the number of pixels in the image,

KOUNT is the number of vectors,

INT is the separation of modified data values (e.g., 3),

NUM is the maximum population for which vectors are modified,

N, NPOP are the tables of vectors and their populations (from HASH),

IFEAT is the lengths of the tables,

JFAC, IMOD, JMOD are the multiplier, divisor and base used in HASH.

IV. INPUT/OUTPUT

Input and output are by the tables N and NPOP.

V. DESCRIPTION OF SUBROUTINES

No other subroutines are called.

VI. PERFORMANCE SPECIFICATIONS

1. STORAGE

The subroutine requires 3880 bytes. The tables N and NPOP will require several thousand bytes for a large image.

2. EXECUTION TIME

The vectors are modified at a rate of approximately 1000 per second.

VII. METHOD

For each vector in the table having population of up to "NUM," the components are changed to the nearest multiple of "INT". If a different vector has been created, a table location is computed as in HASH. If the modified vector is the same as one previously existing before modification, the number of vectors is reduced by one count, and the populations are added. If a new vector has been generated, the vector and population are transferred to the new table location. In either case, the popula-[illegible]n at the original table location is set to -1 as a flag, and a pointer to the new table location is put in the vector component table entry. The mean squared error of the image defined by the reduced vector set is computed.

VIII. COMMENTS

None.

IX. TESTS

The modified vectors are multiples of INT as expected. An image reconstructed from the reduced vector set has the expected mean squared error.

X. LISTING

The subroutine listing follows.

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
      SUBROUTINE VREDUC (KNT,KOUNT,INT,NUM,N,NPOP,IFEAT,JFAC,IMOD,JMOD)
C
C     KNT=A VARIABLE USED TO COUNT THE NUMBER OF PICTURE ELEMENTS(PELS)
C     KOUNT=A VARIABLE USED TO COUNT THE NUMBER OF DIFFERENT VECTORS
C     INT = INTEGER MULTIPLIER TO WHICH COMPONENTS ARE MODIFIED
C     NUM = MAXIMUM POPULATION FOR WHICH VECTORS ARE MODIFIED
C     N(I)=COMPONENTS OF THE I'TH VECTOR IN THE TABLE
C     NPOP(I)=THE NUMBER OF OCCURRENCES OF THE ITH VECTOR IN THE TABLE
C     IFEAT=MAXIMUM NUMBER OF DIFFERENT VECTORS ALLOWED
C     JFAC=MULTIPLIER
C     IMOD=DIVISOR
C     JMOD=BASE
C
      DIMENSION N(IFEAT), NPOP(IFEAT), NN(300), MSE(4)
      LOGICAL*1 LM(4), LO(4)
      EQUIVALENCE (IM,LM(1)), (NOLD,LO(1))
      DATA NBAND /4/
C
      KONST = INT/2
      DO 400 NB=1,NBAND
  400 MSE(NB) = 0
C
      DO 445 I=1,IFEAT
      IF (NPOP(I).EQ.0) GO TO 445
      IF (NPOP(I).GT.NUM) GO TO 445
C
C     COMPUTE THE COMPONENTS OF THE INTEGER MODE VECTOR
      IM = N(I)
      NOLD = IM
      DO 410 NC=1,4
  410 LM(NC) = (LM(NC)+KONST)/INT*INT
C
C     IF THE VECTOR IS UNCHANGED, JUMP OUT
      IF (IM.EQ.NOLD) GO TO 445
C
C     HASH ROUTINE
C     COMPUTE MEAN SQUARED ERROR
      L=0
      DO 415 NC=1,4
      NX1 = LO(NC)
      NXX = LM(NC)
      MSE(NC) = MSE(NC) + NPOP(I)*IABS(NX1-NXX)
      L = JMOD*L + MOD(NXX,IMOD)
  415 CONTINUE
      L=L*JFAC+1
C
C     CHECK FOR EMPTY TABLE LOCATION TO PUT NEW VECTOR
  417 IF (NPOP(L).GT. 0) GO TO 425
      IF (NPOP(L).EQ.-1) GO TO 435
C
C     HAVE FOUND A NEW VECTOR
C     TRANSFER POPULATION AND NEW VECTOR TO TABLE LOCATION L
C     PUT POINTER FLAG AND POINTER AT OLD TABLE LOCATION
      NPOP(L) = NPOP(I)
      N(L) = IM
      NPOP(I) = -1
      N(I) = L
      GO TO 445
C
C     TABLE LOCATION IS FILLED,
C     CHECK TO SEE IF VECTOR IS IN TABLE
  425 IF (N(L).NE.IM) GO TO 435
C
C     INTEGER MODE CREATED A PREVIOUSLY EXISTING VECTOR
C     ADD PREVIOUS POPULATION AND REDUCE VECTOR COUNT BY 1
C     PUT POINTER FLAG AND POINTER AT OLD TABLE LOCATION
      NPOP(L) = NPOP(L) + NPOP(I)
      KCOUNT=KCOUNT-1
      NPOP(I) = -1
      N(I) = L
      GO TO 445
C
C     VECTOR NOT IN TABLE, INCREMENT TABLE INDEX, GO BACK TO CHECK
C     ON TABLE LOCATION
  435 L=L+1
      IF(L.GT.IFEAT) L=1
      GO TO 417
  445 CONTINUE
C
      WRITE (6,750)
      WRITE(6,860) INT,NUM
      WRITE (6,800) KCOUNT
C
C     WRITE OUT FEATURE VECTORS THAT OCCUR AT LEAST 1000 TIMES
      NVEC = 0
      DO 450 I=1,IFEAT
      IF (NPOP(I).LT.1000)GO TO 450
      IM = N(I)
      NVEC = NVEC + 1
      IF (NVEC.EQ.1) WRITE (6,810)
      IF (MOD(NVEC,2).EQ.1) WRITE (6,811) LM, NPOP(I)
      IF (MOD(NVEC,2).EQ.0) WRITE (6,812) LM, NPOP(I)
  450 CONTINUE
C
C     POPULATION DISTRIBUTION IN LOGARITHMIC INCREMENTS
      DO 500 I=1,300
  500 NN(I)=0
      DO 550 I=1,IFEAT
      IF (NPOP(I).LT.1) GO TO 550
C
C     COUNT THE NUMBER OF VECTORS THAT OCCUR 1000'S OF TIMES
      II=NPOP(I)/1000
      IF(II.LT.1)GO TO 510
      II=II+108
      GO TO 540
C
C     COUNT THE NUMBER OF VECTORS THAT OCCUR 100'S OF TIMES
  510 II=NPOP(I)/100
      IF (II.LT.1)GO TO 530
      II=II+99
      GO TO 540
C
C     COUNT THE NUMBER OF VECTORS THAT OCCUR FROM 1 TO 99 TIMES
  530 II=NPOP(I)
  540 NN(II)=NN(II)+1
  550 CONTINUE
C
C     PRINT THE NUMBER OF VECTORS THAT OCCUR 1-99 TIMES, 100'S AND
C     1000'S OF TIMES
      WRITE (6,815)
      J = 0
      INC = 1
      DO 560 I=1,300
      IF (I.EQ.101) INC = 100
      IF (I.EQ.110) INC = 1000
      J = J + INC
      IF (NN(I).EQ.0) GO TO 560
      X = NN(I)*100.0/KOUNT
      WRITE (6,820) J, NN(I), X
  560 CONTINUE
C
C     COMPUTE THE MEAN, VARIANCE, AVERAGE MSE
      CNT = KNT
      ALN2 = ALOG10(2.0)
      AMSE = 0.0
      DO 610 NB=1,NBAND
      NXM = 0
      NXV = 0
      DO 600 I=1,IFEAT
      IF (NPOP(I).LT.1) GO TO 600
      IM = N(I)
      NXX = LM(NB)
      NXM = NXM + NPOP(I)*NXX
      NXV = NXV + NPOP(I)*NXX**2
  600 CONTINUE
      XMEAN = NXM / CNT
      SIGMA = (NXV-CNT*XMEAN**2) / (CNT-1.0)
      SGERR = MSE(NB) / CNT
      AMSE = AMSE + SGERR
  610 WRITE (6,830) NB, XMEAN, SIGMA, SGERR
      AMSE = AMSE/NBAND
      WRITE (6,840) AMSE
      RETURN
C
  750 FORMAT ('1')
  800 FORMAT (//'   VECTOR NO. =',I8)
  810 FORMAT ('1',20X,'VECTORS WITH POPULATIONS OF AT LEAST 1000'//)
  811 FORMAT (10X,4I4,I10)
  812 FORMAT ('+',50X,4I4,I10)
  815 FORMAT ('1',10X,'NO. OF TIMES'/8X,'A VECTOR OCCURRED',10X,'NO. OF
     .VECTORS',10X,'PERCENT OF TOTAL'//)
  820 FORMAT (I20,I25,F25.4)
  830 FORMAT (/10X,'BAND',I3,5X,'MEAN =',F10.3,5X,'VARIANCE =',F10.3,5X,
     .'MEAN SQUARED ERROR ='F8.4)
  840 FORMAT (/60X,'AVERAGE MEAN SQUARED ERROR =',F8.4)
  860 FORMAT ('  INT =',I5,'    NUM =',I6)
      END
```

BLOB-1

SEGMENTATION OF PICTURE

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

I. NAME

REFORM

II. DESCRIPTION

The REFORM program partitions the image into smaller segments in order to conserve core in the execution of the BLOB subroutine. REFORM reads the portion of the image which is to be processed, creates rectangular segments, and writes one segment per channel to the segmented image file to be used by subroutine BLOB.

III. CALLING SEQUENCE

Call REFORM (SEG, RDATA, BDATA)

where

SEG is an array which holds four virtual segments;

RDATA is an array which holds one line of image data expanded to one pixel component value per word.

BDATA is an array which holds one line of image data in bytes.

IV. INPUT/OUTPUT

1. INPUT

Unit 12 - a sequential data set containing the image to be processed. The data set consists of NROW records of NCOL * TCHA bytes per record, each record corresponding to a line of the image. Each record is in vector format and contains one pixel component per byte.

COMMON/DIM/NROW, NCOL, HAFM, HAFN, TCHA, VSIZH, VSIZV, RSIZ, NREC, BLKS, DSIZ, VDIMH, VDIMV, VFIL, MIPS

where

NROW is the number of lines to be processed.

NCOL is the number of pixels per line to be processed.

HAFM is NROW/2.

HAFN is NCOL/2.

TCHA is the total number of components (channels) in the input image.

VSIZH is the horizontal dimension of a virtual segment in pixels.

VSIZV is the vertical dimension of a virtual segment in pixels.

RSIZ is the size of one segment in words (RSIZ = VSIZH * VSIZV).

NREC is the number of segments in the virtual file.
(NREC = VDIMH * VDIMV * TCHA)

BLKS is the number of segments per component (channel).
(BLKS = VDIMH * VDIMV + 1)

DSIZ is the maximum size of the directional lists.

VDIMH is the horizontal dimension of the image in segments.

VDIMV is the vertical dimension of the image in segments.

VFIL is the unit number of the segmented image file.

MIPS is the maximum number of initial points. This parameter is used by the program RECON.

2. OUTPUT

The output of REFORM will be on unit VFIL as a direct access data set with NREC records. There will be one record per segment per channel in the virtual file.

3. FILE STORAGE

No additional files are required by REFORM.

V. DESCRIPTION OF SUBROUTINES

The storage requirements and the functions of subroutines used are given in the following table.

DESCRIPTION OF SUBROUTINES FOR REFORM

SUBROUTINE NAME (Entry Points)	STORAGE (Bytes)	FUNCTION
REFORM	1708	Initialize starting point, call routine to read line of data, build one row of virtual segments, write to file.
GADLIN	608	Read one line of image data, expand data to one pixel value per word.

The linkages of the subroutines are given in the following table.

LINKAGES OF SUBROUTINES FOR REFORM

REFORM	GADLIN

VI. PERFORMANCE SPECIFICATIONS

1. STORAGE

The subroutine REFORM is 1708 bytes long. The storage needed to run this subroutine depends on the size of the image to be processed. The storage to process an image of 112 x 112 pixels, including a driver and the required subroutines, is 94K.

2. EXECUTION TIME

The execution time is dependent on the size of the image to be processed. To reformat an image 112 x 112 pixels required approximately 11 seconds of CPU time.

VII. METHOD

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

The input unit number is set to 12.

The area to be processed is determined by setting the starting line and sample to 1, the number of lines to NROW, and the number of samples to NCOL.

Virtual segments are constructed one row at a time as follows:

REFORM calls subroutine GADLIN which reads one line of data, expands it to one pixel component per word, and stores the expanded data in the real array RDATA.

The virtual segments are then written to the output file on unit VFIL. One row of segments is written per channel.

This process is repeated until NROW lines have been read and segmented.

SUBROUTINE REFORM

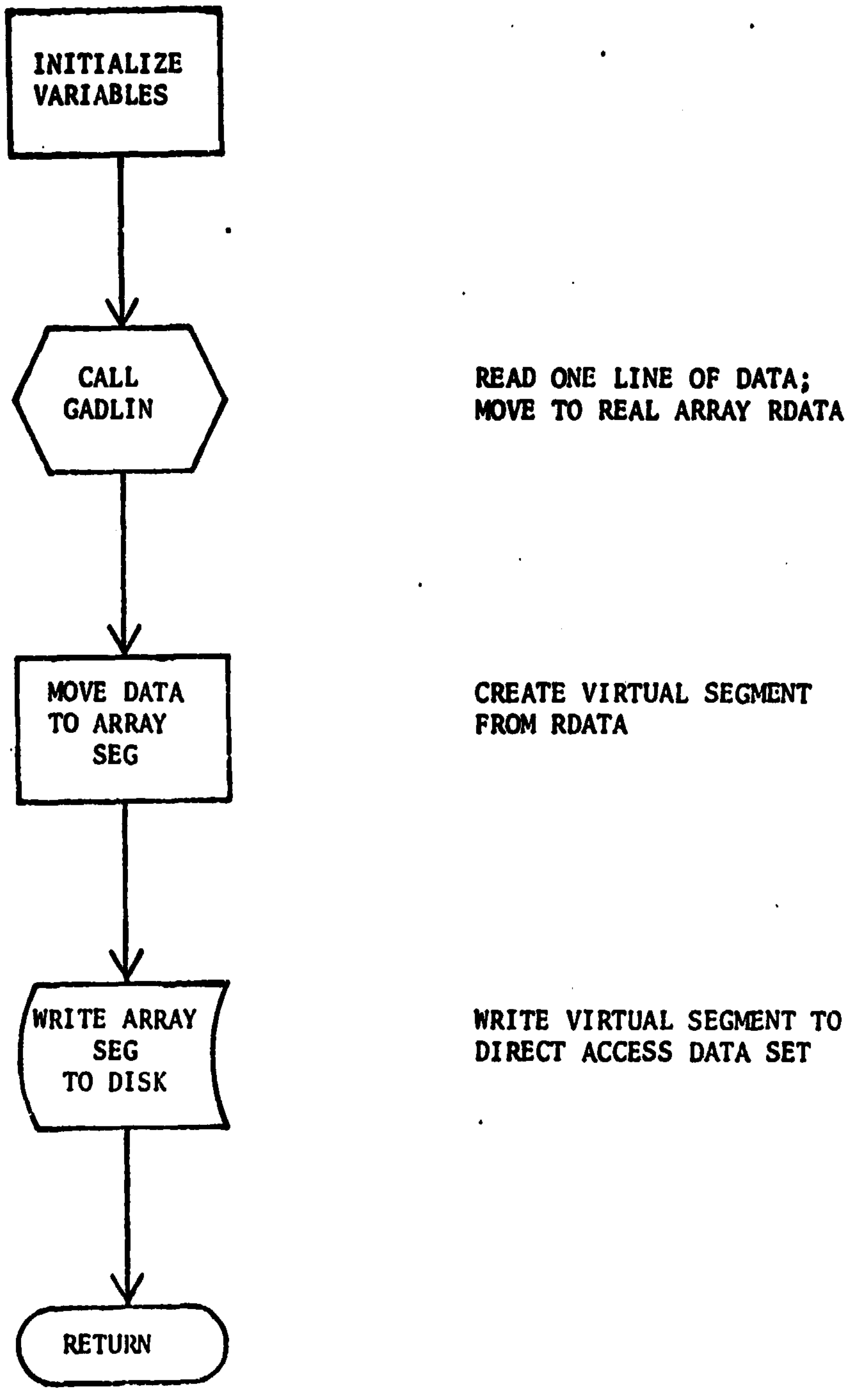

INITIALIZE VARIABLES
CALL GADLIN
READ ONE LINE OF DATA; MOVE TO REAL ARRAY RDATA
MOVE DATA TO ARRAY SEG
CREATE VIRTUAL SEGMENT FROM RDATA
WRITE ARRAY SEG TO DISK
WRITE VIRTUAL SEGMENT TO DIRECT ACCESS DATA SET
RETURN

Figure 16. Flow Chart for Subroutine REFORM

VIII. COMMENTS

The vertical and horizontal dimensions of the image must be integral multiples of the respective vertical and horizontal dimensions of the virtual segments.

IX. TESTS

The reconstructed images from the BLOB package have been examined by use of mean square error calculations and plots and histograms of difference images.

X. LISTINGS

Listings of the subroutines follow.

```
      SUBROUTINE REFORM (SEG, RDATA, BDATA)
C
C     SUBROUTINE REFORM EXTRACTS THE AREA OF INTEREST FROM THE
C     INPUT PICTURE IN VECTOR FORMAT AND PARTITIONS IT INTO VIRTUAL
C     SEGMENTS FOR ALCH PROGRAM.
C
C     MODIFIED BY HANS G. MOIK NASA/GSFC CODE 935, SEPTEMBER 1976
C     MODIFIED BY JULIA M. HODGES, DECEMBER 1977
C
C ****************************************************************
C ****************************************************************
C
      IMPLICIT INTEGER*4 (A-Z)
      REAL*4 SEG(VSIZV,VSIZH,VDIMH,TCHA),RDATA(TCHA,NCOL)
      LOGICAL*1 BDATA(TCHA,NCOL)
      COMMON/DIM/NCOL,NROW,MAFM,MAFN,TCHA,VSIZH,VSIZV,RSIZ,NREC,BLKS,
     * DSIZ,VDIMH,VDIMV,VFIL,MIPS
C
C     INITIALIZE INPUT UNIT, STARTING LINE, STARTING SAMPLE,
C               NUMBER OF LINES, NUMBER OF SAMPLES.
      UNIT=12
      SL=1
      SS=1
      NL=NROW
      NS=NCOL
      WRITE( 6,1603)
C
      DO 5 YSEG = 1, VDIMV
         LINE = 0
C
C        BUILD ONE ROW OF VIRTUAL SEGMENTS
         DO 4 SLNUM = 1, VSIZV
            LINE = LINE + 1
            CALL GADLIN (RDATA, BDATA, UNIT, SS, NS, LINE, &80, &90)
C
C           NO ERR, SO MOVE DATA TO VIRTUAL PICTURE FILE
    7       DO 3 CHAN = 1, TCHA
               COL = 0
               DO 2 XSEG = 1, VDIMH
                  DO 1 SEGCOL = 1, VSIZH
                     COL = COL + 1
                     SEG(LINE, SEGCOL, XSEG, CHAN) = RDATA(CHAN, COL)
    1             CONTINUE
    2          CONTINUE
    3       CONTINUE
    4    CONTINUE
C
C        WRITE ONE ROW OF SEGMENTS PER CHANNEL
         DO 5 CHAN = 1, TCHA
            DO 5 XSEG = 1, VDIMH
               BLK = (CHAN-1)*(BLKS-1) + (YSEG-1)*VDIMH + XSEG
               WRITE(6,1602) BLK
               WRITE(VFIL'BLK)((SEG(X,Y,XSEG,CHAN),X=1,VSIZV),Y=1,VSIZH)
    5 CONTINUE
      RETURN
C
```

```
C      ERROR RETURNS FOR GADLIN
   90 WRITE( 6,1600)
      RETURN
   80 WRITE( 6,1601)
      RETURN
C
 1600 FORMAT (' PERMANENT I/O ERROR ON INPUT TAPE')
 1601 FORMAT (' END OF FILE OR VOLUME ON INPUT TAPE')
 1602 FORMAT (' WRITING BLOCK',I6)
 1603 FORMAT (' STARTING THE REFORMATING')
      END
      SUBROUTINE GADLIN (RDATA, BDATA, UNIT, SS, NS, LINE, *, *)
C
C     SUBROUTINE GADLIN MOVES ONE LINE OF PICTURE DATA
C        TO REAL NUMBER ARRAY
C     INPUT TAPE SHOULD BE IN VECTOR FORMAT.
C
C ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C ************************************************************************
C
      INTEGER MAFM,MAFN,TCHA,VSIZH,VSIZV,RSIZ,BLKS,DSIZ,
     C  VDIMH,VDIMV,VFIL,MIPS
      INTEGER    UNIT
      REAL*4 RDATA(TCHA,NCOL)
      LOGICAL*1 BDATA(TCHA,NCOL)
      COMMON/DIM/NCOL,NROW,MAFM,MAFN,TCHA,VSIZH,VSIZV,RSIZ,NREC,BLKS,
     # DSIZ,VDIMH,VDIMV,VFIL,MIPS
C
C     READ A LINE OF PICTURE DATA
      READ (UNIT,END=400,ERR=500) BDATA
C
C     STORE EACH RECORD BYTE (PIXEL COMPONENT VALUE)
C        AS A FULL WORD REAL NUMBER.
      DO 100 I=1,NS
         DO 200 J = 1, TCHA
         RDATA(J,I) = BDATA(J,I)
 200     CONTINUE
 100  CONTINUE
C
      RETURN
C
C     ERROR RETURN FOR READ
 400  RETURN 1
 500  RETURN 2
      END
```

BLOB-2

DETECT HOMOGENEOUS REGIONS

I. **NAME**

BLOB

II. **DESCRIPTION**

The BLOB program locates homogeneous regions (blobs) in an image by detecting the boundaries between regions. Statistical F- and t- tests are performed to determine whether a pixel is an element of the region currently being outlined. Output files are created for reconstruction of the image using the program RECON.

III. **CALLING SEQUENCE**

Call BLOB (DIR, OUT, ADIR, VMEM, XCPL, YCPL, BLOBM, IPSSQ, MEANS, BKSTAT, IPMEAN, STATUS)

where

DIR is the stack used to allow the program to back up in its tracing of contours when its forward path is blocked.

OUT is the direction upon leaving a pixel group in a contour.

ADIR is the list of directionals for the current initial point.

VMEM is the virtual memory array containing four segments.

XCPL is the X-coordinate of a pixel group in the comparison pointer list.

YCPL is the Y-coordinate of a pixel group in the comparison pointer list.

BLOBM is the mean of the current pixel group.

IPSSQ is the sum of the squares of the pixel group in the initial point.

MEANS is the mean values of the pixel group in the current initial point.

BKSTAT is the index for the virtual segment to be retrieved.

IPMEAN is the mean of the current initial point.

STATUS is the array holding the status flags for each pixel group.

IV. INPUT/OUTPUT

1. INPUT

a. Card 1 in FORMAT (2F10.3):

FVAL the F-test value

TVAL the t-test value

Possible values for FVAL and TVAL are listed below, they do not have to be at the same level. (See References 3 and 4.)

LEVEL	FVAL	TVAL
1	29.45	1.44
2	47.467	1.949
3	141.1	2.447
4	261.0	3.143
5	884.6	3.707
6	1514.0	5.959

b. Card 2 in FORMAT (I2, I1):

NCHA	- the number of components (channels) to be used for contouring. The first NCHA components will be used, and NCHA must not be larger than the value specified by the symbolic parameter TCHA.
IPRINT	- a non-zero value for IPRINT will cause the region description and directional list to be printed.

c. Direct access segmented image file on unit VFIL as generated by the reformatting program REFORM. Four segments at a time are kept in main memory. BLOB traces contours and if one of the segments in main memory does not contain a referenced pixel, the required segment is read in and replaces the oldest previous segment.

d. COMMON/DIM/NCOL, NROW, HAFN, HAFM, TCHA, VSIZV, VSIZH, RSIZ, NREC, BLKS, DSIZ, VDIMV, VDIMH, VFIL, MIPS as described for the subroutine REFORM.

2. OUTPUT

a. Description of each detected region in form IPX, IPY, BLOBN, NDIR, MEANS on unit 8.

IPX - the column coordinate of the initial point of a region.

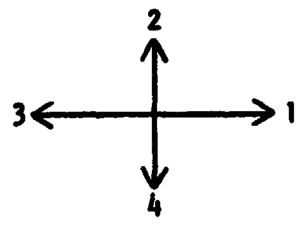
IPY - the line coordinate of the initial point of a region.

BLOBN - the number of points in the region times four. BLOBN = 4 denotes a singular point.

NDIR - the number of directional elements in the contour of the region.

MEANS - the mean values for the NCHA components of the region.

b. Directional list for each detected region. NDIR directional elements are written on unit 9. A directional element can only assume the values 1, 2, 3, or 4 corresponding to the directions shown below.



c. Printed Output - the number of initial points (regions) and the total number of directional elements are printed. Optionally, the region description and the directional list are printed.

3. FILE STORAGE

No additional files are used by this program.

V. <u>DESCRIPTION OF SUBROUTINES</u>

The subroutine BLOB calls several subroutines.

DESCRIPTION OF SUBROUTINES FOR BLOB

SUBROUTINE	STORAGE (DECIMAL BYTES)	FUNCTION
BLOB	2776	Initializes variables and arrays. Locates new initial point, traces contour, and sets contour bits when completed. Writes initial point information and directionals to files.
COMPAR	1444	Compare pixel groups to determine points in current contour.
ERR	1820	Print error messages.
ERRMSG	Entry under ERR	Traceback for errors.
GETPIX	2958	Looks for pixel groups to add to the current contour.
GSTAT	1036	Gets the current status of a pixel group.
INITV	Entry under PIXEL	Initializes virtual memory parameters.
IPCPAR	900	Compares pixel groups to locate initial point of new contour.
MEAN	650	Computes the mean of a pixel group.
NEWIP	1734	Locates initial point for a new contour.
PIXEL	1542	Locates virtual segment containing required pixel.
PSTAT	Entry under GSTAT	Stores current status of pixel group.
SET	Entry under GSTAT	Sets status bit to designate a point in some contour.
SICB	478	Sets status contour bit for all points in a contour.
SSQ	510	Computes the sum ot the squares for a pixel group.

LINKAGES OF SUBROUTINES FOR BLOB

CALLING PROGRAM	PROGRAM CALLED
BLOB	INITV*
	PSTAT+
	SET+
	SICB
	SSQ
	MEAN
	NEWIP
	GETPIX
SCIB	SET+
SSQ	PIXEL
MEAN	PIXEL
NEWIP	GSTAT
	ERR
	IPCPAR
GETPIX	ERR
	PSTAT+
	COMPAR
PIXEL	ERRMSG#
IPCPAR	MEAN
	SSQ
COMPAR	MEAN
	PSTAT+
	SSQ
	GSTAT

* Entry under PIXEL

+ Entry under GSTAT

# Entry under ERR

VI. PERFORMANCE SPECIFICATIONS

1. STORAGE

The storage required to process an image 112 X 112 pixels with 4 channels, including a driver and the required subroutines, is 121K.

2. EXECUTION TIME

The execution time is dependent on the size and complexity of the image, and the F- and t- values chosen. To process an image of 112 x 112 pixels required approximately 1 minute and 16 seconds of CPU time.

3. RESTRICTIONS

The dimensions of the image to be processed must be multiples of 2. The dimensions of the virtual segments must be multiples of 2.

VII. METHOD

The BLOB program package is an implementation of the BLOB algorithm (see References 3 and 4) for IBM 360/370 computers with OS/MVT. The BLOB algorithm detects homogeneous regions (blobs) in monochrome images or multi-images (multispectral, multitemporal). The algorithm guarantees closed boundaries of the regions. Its output is a list description of the detected regions consisting of the coordinates of an initial point for each region, the number of points within each region, the number of boundary points for each region, the mean values of each component of the multi-image for each region and a list of directionals describing the contour of each region. A multi-image may be reconstructed from this compressed description using the program RECON.

The values for FVAL, TVAL, NCHA, and IPRINT are read from cards. The arrays VMEM and STATUS are set to zero, and other variables are initialized.

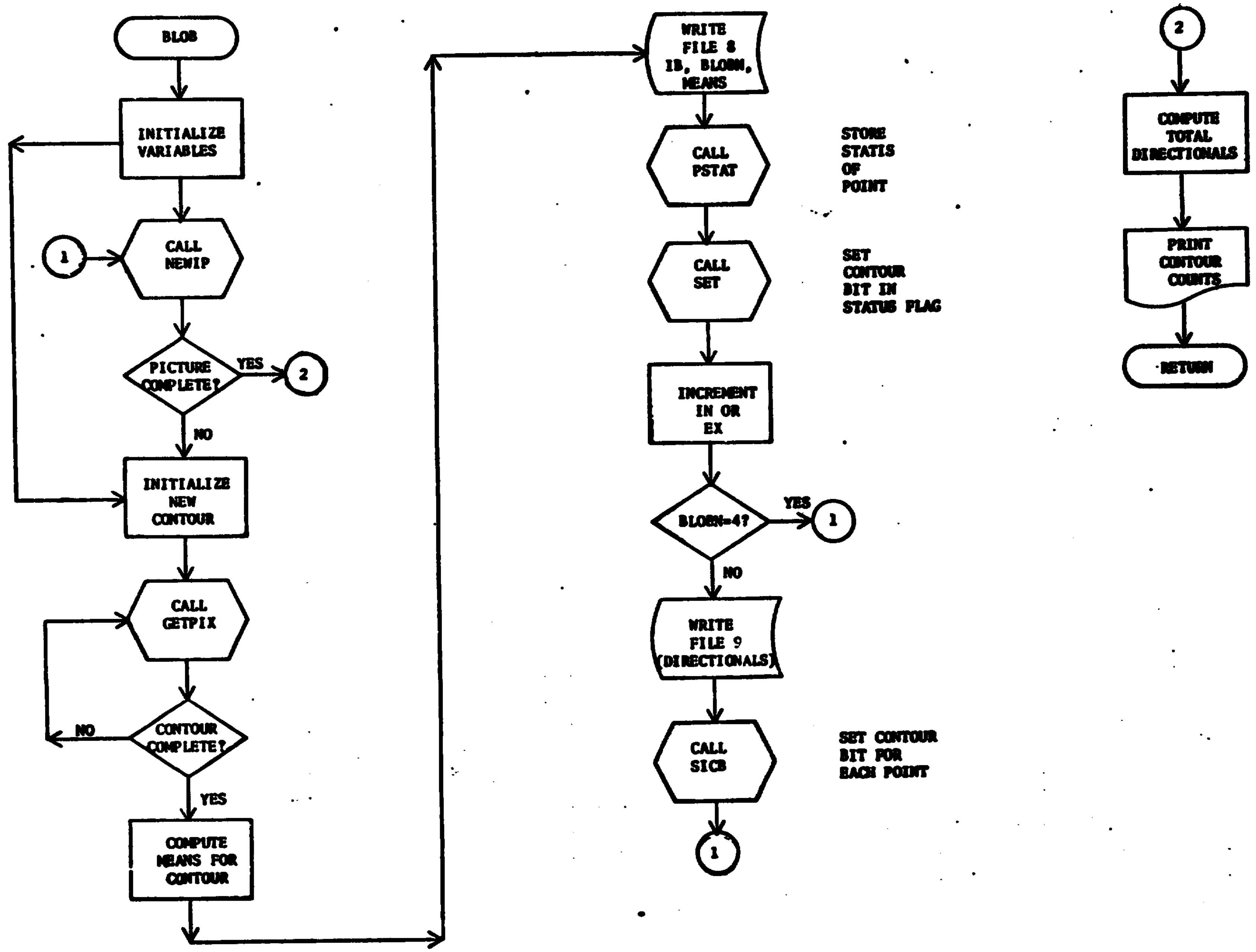


Figure 17. Flow Chart for the BLOB Program.

The first initial point is set to (1,1) and the contour parameters are initialized. The image is processed in groups of four pixels, one point being a 2 x 2 pixel group.

The subroutine GETPIX is called repeatedly to trace out the contour by searching for adjacent pixel groups having the same statistical values as the initial point. The search continues until it is determined that the blob contains a single pixel group or until the contour is completed. Entries into the directional list ADIR are created, and the status of each point is computed.

The mean of the blob is computed for each channel. The initial point coordinates, number of points in the blob, number of entries in the directional list, and the means are written out to unit 8.

If the blob contains more than one pixel group, the directional list is written to unit 9 and subroutine SICB is called to set the status contour bit for each point of the contour.

The subroutine NEWIP is called to locate the initial point of a new contour and the contouring process is repeated.

VIII. COMMENTS

None.

IX. TESTS

The reconstructed images from the BLOB package have been examined by use of mean square error calculations and plots and histograms of difference images.

X. LISTINGS

Listings of the subroutines follow.

```
C      MAIN PROGRAM  *  DRIVER FOR BLOB ROUTINE
C
C      THE FOLLOWING LIST DEFINES THE PARAMETERS USED BY THE BLOB
C      SUBROUTINES, ALONG WITH SAMPLE VALUES.
C
C      NOTE THAT NROW, NCCL, VSIZV, AND VSIZH MUST BE MULTIPLES OF 2 AND
C      NROW = VSIZV * VDIMV
C      NCOL = VSIZH * VDIMH
C
C      NROW = 256      THE VERTICAL SIZE OF THE PICTURE BEING CONTOUREC
C      HAFM = 128      HALF OF NROW
C      NCOL = 256      THE HORIZONTAL SIZE OF THE PICTURE BEING CONTOURED
C      HAFN = 128      HALF OF NCOL
C      TCHA = 4        TOTAL NUMBER OF CHANNELS IN THIS PICTURE
C      VSIZV = 32      VERTICAL DIMENSION OF VIRTUAL SEGMENT IN PIXELS
C      VSIZH = 32      HORIZONTAL DIMENSION OF VIRTUAL SEGMENT IN PIXELS
C      RSIZ = 1024     SIZE OF VIRTUAL SEGMENT IN WORDS (VSIZV*VSIZH)
C      NREC = 256      NUMBER OF RECS IN VIRT FILE = VDIMV*VDIMH*TCHA
C      BLKS = 65       NUMBER OF VIRTUAL SEGMENTS PER CHANNEL PLUS 1
C      DSIZ = 4096     MAXIMUM SIZE OF THE DIRECTIONAL LISTS
C      VDIMV = 8       VERTICAL DIMENSION OF PICTURE IN VIRTUAL SEGMENTS
C      VDIMH = 8       HORIZONTAL DIMENSION OF PICTURE IN VIRTUAL SEGMENTS
C      VFIL = 11       UNIT NUMBER OF VIRTUAL FILE
C      MIPS = 9999     MAXIMUM INITIAL POINTS ALLOWED FOR PICTURE
C
C      DEFINE FILE VFIL (NREC,RSIZ,U,NXTREC)
C
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C***************************************************************************
C
       DIMENSION SEG(28,32,6,4), RDATA(4,192), BLOBM(4)
       REAL MEANS(4), IPMEAN(4), IPSSQ(4)
       INTEGER BKSTAT(25), OUT(96)
       INTEGER   PMEANS(7000,4)
       INTEGER ADIR(2000), DIR(2000), XCPL(96), YCPL(96)
       INTEGER BOUND(96,56), IPBS(96)
       INTEGER HAFM,HAFN,TCHA,VSIZH,VSIZV,RSIZ,BLKS,DSIZ,
      C  VDIMH,VDIMV,VFIL,MIPS
       LOGICAL*1 STATUS(96,56),SLINE(4,192),TSL(768)
       EQUIVALENCE (SEG(1,1,1,1),PMEANS(1,1))
C
       COMMON//ATOP,DTOP
       COMMON/DIM/NCOL,NROW,HAFM,HAFN,TCHA,VSIZH,VSIZV,RSIZ,NREC,BLKS,
      # DSIZ,VDIMH,VDIMV,VFIL,MIPS
       COMMON/BLOB1/BLCBN
       COMMON/PIX/N,M,MAXMN,NEXTB
       COMMON/IPL/IPX,IPY,IPCNT,EXT
       COMMON /LEVELS/ FVAL, TVAL, NCHA
       COMMON /LAST/ XLAST,YLAST
       COMMON/VMEMRY/PGCNT
       COMMON /CPL/ TOPCPL
C
       DEFINE FILE 11 ( 96,896,U,NXTREC)
C
       NROW = 112
       NCOL = 192
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
      VSIZV = 28
      VSIZH = 32
      HAFM = NROW/2
      HAFN = NCOL/2
      TCHA = 4
      VDIMV = NROW/VSIZV
      VDIMH = NCOL/VSIZH
      HSIZ = VSIZV*VSIZH
      NREC = VDIMV*VDIMH*TCHA
      BLKS = VDIMV*VDIMH + 1
      VFIL = 11
      DSIZ = 2000
      MIPS = 7000
      N = NCOL
      M = NROW
      MAXMN = (MAX0(M,N))/2
      N4 = 4 * NCOL
C
      CALL REFORM (SEG, HDATA, SLINE)
C
      CALL BLOB (DIR,CUT,ADIR,SEG ,XCPL,YCPL,BLOBM,IFSSQ,MEANS,
     #   HKSTAT,IPMEAN,STATUS)
C
      REWIND 8
      REWIND 9
      REWIND 12
      CALL RECON(ADIR,SLINE,TSL,STATUS,PMEANS,BOUND,IPBS,N4)
      STOP
      END
      SUBROUTINE BLOB (DIR,OUT,ADIR,VMEM,XCPL,YCPL,BLOBM,IPSSQ,MEANS,
     #    BKSTAT,IPMEAN,STATUS)
C
C     WRITTEN BY PETER C. MILLER IN THE FALL AND WINTER OF 1974
C     MODIFIED BY HANS G. MOIK, NASA/GSFC CODE 933 IN SEPTEMBER 1976
C     FOR IBM 360 WITH OS/MVT
C     MODIFIED BY JULIA M. HODGES, DECEMBER 1977
C
C     THIS PROGRAM TRACES OUT THE CONTOUR BASED ON THE FIRST
C     N CHANNELS OF THE PRINCIPAL COMPONENT TRANSFORM WHERE N
C     CAN BE UP TO ALL THE TRANSFORMED CHANNELS.
C
C  FILES USAGE:
C     IP = THIS FILE HOLDS ALL THE INFORMATION ABOUT ALL THE
C          INTIAL POINTS (IP) FOR A PICTURE AND THE MEAN OF EACH CONTOUR
C           THE DATA IS FIXED LENGTH RECORDS.
C          ( FORTRAN UNIT NUMBER 8 )
C     DIR = THIS FILE HOLDS THE DIRECTIONAL INFORMATION FOR ALL THE
C           CONTOURS. THE DATA IS VARIABLE LENGTH RECORDS.
C           ( FORTRAN UNIT NUMBER 9 )
C     VFILE = THIS IS THE VIRTUAL PICTURE MEMORY FILE WHICH IS
C             USED TO STORE THE PICTURE SEGMENTS THAT ARE NOT
C             CURRENTLY NEEDED IN CORE.  (READ ONLY FILE )
C             ( FORTRAN UNIT NUMBER 11 )
C     OUTPUT = FILE USED TO PRINT STATISTICS AND ANY ERROR MESSAGES.
C              ( FORTRAN UNIT NUMBER 6 )
C     INPUT = FILE USED TO READ IN FVAL, TVAL, TCHA, AND IPRINT
C             (  FORTRAN UNIT NUMBER 5  )
C
C       FVAL = THE CURRENT F=TEST VALUE
C       TVAL = THE CURRENT T=TEST VALUE
C     POSSIBLE VALUES FOR FVAL AND TVAL ARE LISTED BELOW.
C     THEY BOTH DON'T HAVE TO BE AT THE SAME LEVEL.
C     FURTHERMORE THE LEVELS USED IN IPCPAR AND COMPAR NEED
C     NOT BE THE SAME. (GROUPS OF FOUR PIXELS HAVE BEEN ASSUMED
C     FOR THESE VALUES.)
C LEVEL       F=VALS          T=VALS
C   1          29.45           1.44
C   2          47.467          1.943
C   3         141.1            2.447
C   4         261.0            3.1433
C   5         884.6            3.707
C   6        1514.0            5.959
C
C       ADIR = A LIST OF ALL THE DIRECTIONALS FOR THIS IP
C       ATOP = POINTS TO THE TOP ENTRY IN THE ACTUAL DIRECTIONAL LIST
C       DIR = STACK USED TO ALLOW THE PROGRAM TO BACKUP IN ITS TRACING
C              OF CONTOURS WHEN ITS FORWARD PATH IS BLOCKED
C       DTOP = POINTS TO THE TOP OF THE DIRECTION STACK
C     XLAST = THE X COORDINATE OF THE PIXEL GROUP AT THE FRONT OF
C             OUR CURRENT CONTOUR.  (USUALLY THE LAST GROUP FOUND BY
C             GETPIX , EXCEPT IF WE HAD TO BACK UP )
C     YLAST = THE Y COORDINATE OF THE PIXEL GROUP AT THE FRONT OF
C             OUR CURRENT CONTOUR.  (USUALLY THE LAST GROUP FOUND BY
C             GETPIX , EXCEPT IF WE HAD TO BACK UP )
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```
C       IPX = THE X COORDINATE OF THE CURRENT INITAL POINT (IP)
C       IPY = THE Y COORDINATE OF THE CURRENT IP
C       IPMEAN = THE MEAN OF THE CURRENT IP
C       IPSSQ = THE SUM OF THE SQUARES OF THE PIXEL GROUP IN THE IP
C       IPCNT = RUNNING SUM OF THE NUMBER OF IPS
C       EXT = BOOLEAN FLAG TO INDICATE INTERNAL OR EXTERNAL CONTOUR
C                TRUE = EXTERNAL
C                FALSE = INTERNAL
C       STATUS = ARRAY HOLDING THE STATUS FLAGS FOR EACH
C                 PIXEL GROUP.
C       BLOBM = HAS THE MEAN OF THE CURRENT BLOB GROUP
C       BLOBN = HAS THE NUMBER OF PIXELS IN THE CURRENT BLOB
C         VMEM = VIRTUAL MEMORY OF OUR PICTURE;
C                IT HOLDS FOUR SUBPICTURES.
C          TOPCPL = TOP OF THE COMPARISON POINTER LIST
C         XCPL = X COORDINATE OF THE PIXEL GROUPS IN THE CPL
C         YCPL = Y COORDINATE OF THE PIXEL GROUPS IN THE CPL
C       N = THE HORIZONTAL DIMENSION OF THE PICTURE MATRIX IN PIXELS
C       M = THE VERTICAL DIMENSION OF THE PICTURE MATRIX IN PIXELS
C               (M AND N MUST BE MULTIPLES OF 2.)
C     MAXMN = (MAX(M,N))/2
C       PIXEL GROUP = IS A GROUP OF FOUR PIXEL ELEMENTS,
C                   WHOSE COORDINATES ARE GIVEN BELOW,
C                   (J,K) , (J,K+1) , (J+1,K) , (J+1,K+1)
C                   THIS DEFINES ONE PIXEL GROUP WHERE J IS DEFINED
C                   BY    J=1,3,5,....,N-1 WHERE N IS THE
C                   HORIZONTAL SIZE OF THE PICTURE
C                   AND   K=1,3,5,....,M-1 WHERE M IS THE VERTICAL
C                   SIZE OF THE PICTURE
C       DONE = BOOLEAN FLAG USED TO TELL CONTOUR THERE ARE NO MORE IPS
C       IPDONE = BOOLEAN FLAG RETURNED BY GETPIX TO SAY IT'S DONE WITH
C                 THE PRESENT CONTOUR
C       X = THE X COORDINATE OF THE CURRENT PIXEL GROUP
C       Y = THE Y COORDINATE OF THE CURRENT PIXEL GROUP
C       MEAN = REAL FUNCTION TO COMPUTE THE MEAN OF 1 PIXEL GROUP
C       MEANS = ARRAY HOLDING THE MEANS OF PRINCIPLE COMPONENTS
C               FOR THE PRESENT IP
C       TDCNT = RUNNING COUNT OF TOTAL NUMBER OF DIRECTIONALS
C               IN THIS PICTURE
C       IN = ARRAY HOLDING COUNT OF NUMBER OF INTERNAL CONTOURS OF SIZE
C              1 , 2 , 3  PIXEL GROUPS
C       EX  = SAME AS IN BUT FOR EXTERNAL CONTOURS
C
C ...............................................................................
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      REAL IPMEAN(TCHA),IPSSQ(TCHA)
      REAL VMEM(VSIZV,VSIZH,4,TCHA)
      REAL MEAN
      INTEGER HAFM,HAFN,TCHA,VSIZH,VSIZV,HSIZ,BLKS,DSIZ,
     C  VDIMH,VDIMV,VFIL,MIPS
      INTEGER BKSTAT(BLKS), OUT(HAFN)
      INTEGER ATOP,DTOP,ADIR( DSIZ),DIR( DSIZ)
      INTEGER XLAST,YLAST,BLOBN,PGCNT
      INTEGER XCPL(MAXMN),YCPL(MAXMN),TOPCPL
      INTEGER X,Y,TDCNT,CHAN,IN(3),EX(3)
      LOGICAL EXT,DONE,IPDONE
      LOGICAL*1 STATUS(MAFN,MAFM)
      DIMENSION BLOBM (TCHA),MEANS(TCHA)
      COMMON/DIM/NCOL,NROW,MAFM,MAFN,TCHA,VSIZH,VSIZV,RSIZ,NREC,BLKS,
     # DSIZ,VDIMH,VDIMV,VFIL,MIPS
      COMMON /LEVELS/ FVAL, TVAL, NCHA
      COMMON // ATOP,DTOP
      COMMON /LAST/ XLAST,YLAST
      COMMON /IPL/ IPX,IPY,IPCNT,EXT
      COMMON/BLOB1/BLOBN
      COMMON/VMEMRY/PGCNT
      COMMON /CPL/ TOPCPL
      COMMON/PIX/N,M,MAXMN,NEXTB
      DATA TOCNT/0/
      DATA IN,EX/6*0/
C
C        INITALIZATION OF VARIABLES
      IPRINT=0
      READ (5,109) FVAL,TVAL
      READ(5,111) NCHA,IPRINT
      IF (NCHA.GT. TCHA) NCHA= TCHA
      WRITE(6,113) NCHA
      WRITE(6,106) FVAL, TVAL
      CALL INITV (0, VMEM, BKSTAT)
      IPCNT = 0
      PGCNT = 0
      ITCNT = 0
C
C     THESE NEXT DO LOOPS ARE REALLY ZEROING OUT STATUS
C     THIS ASSUMES THAT .FALSE. IS REPRESENTED BY A ZERO BYTE
      DO 1 I=1, MAFM
         DO 1 J=1,MAFN
            STATUS(J,I) = .FALSE.
    1 CONTINUE
C
      IPX = 1
      IPY = 1
      GO TO 7
    2 CONTINUE
      CALL NEWIP (DONE,XCPL,YCPL,STATUS,OUT,ADIR,DIR,VMEM,BKSTAT)
      IF( DONE ) GO TO 6
C
C          INITIALIZATION FOR EACH CONTOUR
    7 BLOBN=4
         DO 11 CHAN=1,NCHA
         IPSSQ(CHAN) = SSQ(IPX,IPY,CHAN,VMEM,BKSTAT)
         IPMEAN(CHAN) = MEAN(IPX,IPY,CHAN,VMEM,BKSTAT)
   11    BLOBM(CHAN) = IPMEAN(CHAN)
C
      ATOP = 1
      DTOP=1
      DIR(DTOP)=1
      ADIR(ATOP)=1
      X=IPX
      Y=IPY
      IPCNT=IPCNT+1
      IF( IPY .LT. ITCNT )GO TO 4
      ITCNT = ITCNT + ( VSIZV/ 2 )
      WRITE(6,107) IPY,IPCNT,TDCNT,PGCNT
C
C        GO LOOK FOR ANOTHER PIXEL GROUP
    4 CALL GETPIX (X, Y, IPDONE, ADIR, DIR, BLOBM, IPMEAN, IPSSQ,
     .STATUS, VMEM, BKSTAT, OUT)
      IF( .NOT. IPDONE ) GO TO 4
C
      TDCNT=TDCNT+ATOP
         DO 12 I=1,NCHA
         RMEANS   = BLOBM(I)/FLOAT(BLOBN/4)
   12    MEANS(I) = RMEANS + 0.5
      WRITE(8) IPX,IPY,BLOBN,ATOP,MEANS
      IF(IPRINT.NE.0) WRITE(6,701) IPX,IPY,BLOBN,ATOP,MEANS
      CALL PSTAT (IPX, IPY, ADIR(ATOP), 1, STATUS)
      CALL SET (IPX, IPY, STATUS)
C
C        INCREMENT "IN" OR "EX" IF NECESSARY
      IF( ATOP .GT. 5 )GO TO 10
      IF( EXT )GO TO 9
      IN(ATOP/2+1)=IN(ATOP/2+1)+1
      GO TO 10
    9 EX(ATOP/2+1)=EX(ATOP/2+1)+1
   10 IF( BLOBN .EQ. 4 )GO TO 2
      WRITE(9) ( ADIR(I) , I=1,ATOP )
      IF(IPRINT.NE.0) WRITE(6,702) (ADIR(I),I=1,ATOP)
C
C        SET THE IN CONTOUR BIT FOR CONTOUR JUST COMPLETED
      CALL SICB (ADIR, STATUS)
      GO TO 2
C
    6 TDCNT=TDCNT-IPCNT
      WRITE(6,103) IPCNT,TDCNT
      WRITE(6,104) IN,EX
      RETURN
C
  103 FORMAT(' CONTOURS: NO OF INITIAL POINTS=',I5,//
     #   1X,'TOTAL NO OF DIRECTIONALS=',I9)
  104 FORMAT('0INTERNAL CONTOURS:  IN(1-3)=',3(I5,2X),/
     #    ' EXTERNAL CONTOURS  EX(1-3)=',3(I5,2X))
  105 FORMAT(2I1)
  106 FORMAT('1FLEVEL=',F10.3,'     TLEVEL=',F10.3)
  107 FORMAT('0IPY=',I4,'  IP COUNT=',I5,'  DIR COUNT=',I6,
     #     '  PAGE FAULTS=',I5)
  109 FORMAT(2F10.3)
  111 FORMAT(I2,I1)
  113 FORMAT(5X,'CHANNELS USED =',I3)
  701 FORMAT(8(1X,I4))
  702 FORMAT(20(1X,I4))
      END
      LOGICAL FUNCTION COMPAR (X, Y, ADIR, DIR, BLOBM, IPMEAN, IPSSQ,
     C STATUS, VMEM, BKSTAT)
C
C     THIS FUNCTION COMPARES TWO PIXEL GROUPS TO DETERMINE
C     IF THEY BELONG TO THE SAME CONTOUR, UPDATING
C     STATUS FLAGS AS NECESSARY.
C
C     WRITTEN BY PETER C. MILLER IN THE FALL AND WINTER OF 1974
C
C     XLAST = THE X COORDINATE OF THE PIXEL GROUP AT THE FRONT OF
C             OUR CURRENT CONTOUR.  (USUALLY THE LAST GROUP FOUND BY
C             GETPIX , EXCEPT IF WE HAD TO BACK UP )
C     YLAST = THE Y COORDINATE OF THE PIXEL GROUP AT THE FRONT OF
C             OUR CURRENT CONTOUR.  (USUALLY THE LAST GROUP FOUND BY
C             GETPIX , EXCEPT IF WE HAD TO BACK UP )
C
C       ADIR = A LIST OF ALL THE DIRECTIONALS FOR THIS IP
C       ATOP = POINTS TO THE TOP ENTRY IN THE ACTUAL DIRECTIONAL LIST
C       DIR = STACK USED TO ALLOW THE PROGRAM TO BACKUP IN ITS TRACING
C                OF CONTOURS WHEN ITS FORWARD PATH IS BLOCKED
C       DTOP = POINTS TO THE TOP OF THE DIRECTION STACK
C
C
C       BLOBM = HAS THE MEAN OF THE CURRENT BLOB GROUP
C       BLOBN = HAS THE NUMBER OF PIXELS IN THE CURRENT BLOB
C
C
C       IPX = THE X COORDINATE OF THE CURRENT INITAL POINT (IP)
C       IPY = THE Y COORDINATE OF THE CURRENT IP
C       IPMEAN = THE MEAN OF THE CURRENT IP
C       IPSSQ = THE SUM OF THE SQUARES OF THE PIXEL GROUP IN THE IP
C       IPCNT = RUNNING SUM OF THE NUMBER OF IPS
C       EXT = BOOLEAN FLAG TO INDICATE INTERNAL OR EXTERNAL CONTOUR
C                 TRUE = EXTERNAL
C                 FALSE = INTERNAL
C
C       FVAL = THE CURRENT F-TEST VALUE
C       TVAL = THE CURRENT T-TEST VALUE
C
C ...............................................................
C **************************************************************
C
      REAL MEAN
      REAL IPMEAN(TCHA), IPSSQ(TCHA)
      INTEGER HAFM,HAFN,TCHA,VSIZH,VSIZV,RSIZ,BLKS,DSIZ,
     C  VDIMH,VDIMV,VFIL,MIPS
      INTEGER XLAST,YLAST
      INTEGER ATOP,DTOP,ADIR( DSIZ),DIR( DSIZ)
      INTEGER BLOBN
      INTEGER CHAN
      INTEGER X,Y,GSTAT
      INTEGER BKSTAT(BLKS)
      LOGICAL EXT
      LOGICAL*1 STATUS(HAFN,HAFM)
      DIMENSION VMEM (VSIZV,VSIZH,4,TCHA)
      DIMENSION BLOBM(TCHA)
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```
      COMMON/DIM/NCOL,NROW,HAFM,HAFN,TCHA,VSIZH,VSIZV,RSIZ,NREC,BLKS,
     # DSIZ,VDIMH,VDIMV,VFIL,MIPS
      COMMON /LAST/ XLAST,YLAST
      COMMON // ATOP,DTOP
      COMMON/BLOB1/BLOBN
      COMMON /IPL/ IPX,IPY,IPCNT,EXT
      COMMON /LEVELS/ FVAL, TVAL ,NCHA
      COMMON/PIX/N,M,MAXMN,NEXTB
C
      COMPAR=.FALSE.
      IF ( GSTAT(X,Y,STATUS) .GT. 3) RETURN
      DO 1 CHAN=2,NCHA
         XYM = MEAN(X,Y,CHAN,VMEM,BKSTAT)
         F1 = IPSSQ(CHAN) - 4.0 * IPMEAN(CHAN) * IPMEAN(CHAN)
         F2 = SSQ(X,Y,CHAN,VMEM,BKSTAT) - 4.0*XYM*XYM
         IF (F1 .LE. 0.0 .OR. F2 .LE. 0.0) RETURN
         T = (IPMEAN(CHAN) - XYM) * SQRT(12.0/(F1+F2))
C
C          PERFORM THE T-TEST ON THE TWO GROUPS
         IF(ABS(T) .GE. TVAL) RETURN
C
C          PERFORM THE F-TEST ON THE TWO GROUPS
         F = F1/F2
         IF (F .GE. FVAL .OR. 1.0/F .GE. FVAL) RETURN
    1 CONTINUE
    5 COMPAR=.TRUE.
      IF(GSTAT(X,Y,STATUS) .NE.0) GO TO 6
      BLOBN=BLOBN+4
      DO 2 CHAN=2, TCHA
         BLOBM(CHAN) = BLOBM(CHAN) + MEAN(X,Y,CHAN,VMEM,BKSTAT)
    2 CONTINUE
C
C       UPDATE THE STATUS OF THE PRESENT GROUP BEFORE WE GO TO THE NEXT
    6 IF( XLAST .NE. X )GO TO 20
      IF( YLAST .GT. Y )GO TO 10
      CALL PSTAT(XLAST,YLAST,ADIR(ATOP),2,STATUS)
      RETURN
C
   10 CALL PSTAT (XLAST, YLAST, ADIR(ATOP), 4, STATUS)
      RETURN
C
   20 IF( XLAST .GT. X )GO TO 30
      CALL PSTAT (XLAST, YLAST, ADIR(ATOP), 1, STATUS)
      RETURN
C
   30 CALL PSTAT (XLAST, YLAST, ADIR(ATOP), 3, STATUS)
      RETURN
      END
      SUBROUTINE ERR (ERRNUM, ADIR, DIR, STATUS, OUT)
C
C     WRITTEN BY PETER C. MILLER IN THE FALL AND WINTER OF 1978
C       ADIR = A LIST OF ALL THE DIRECTIONALS FOR THIS IP
C       ATOP = POINTS TO THE TOP ENTRY IN THE ACTUAL DIRECTIONAL LIST
C       DIR = STACK USED TO ALLOW THE PROGRAM TO BACKUP IN ITS TRACING
C               OF CONTOURS WHEN ITS FORWARD PATH IS BLOCKED
C       DTOP = POINTS TO THE TOP OF THE DIRECTION STACK
C       STATUS = ARRAY HOLDING THE STATUS FLAGS FOR EACH
C                 PIXEL GROUP. EACH STATUS FLAG IS 3 BITS LONG
C                 AND THERE ARE (NROW X NCOL)/4 OF THEM FOR A
C                 NROW X NCOL PICTURE.  THEY ARE MANIPULATED
C                 WITH A COMPASS SUBROUTINE CALLED GSTAT.
C       N = THE HORIZONTAL DIMENSION OF THE PICTURE MATRIX IN PIXELS
C       M = THE VERTICAL DIMENSION OF THE PICTURE MATRIX IN PIXELS
C                 (M AND N MUST BE MULTIPLES OF 2.)
C
C ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      INTEGER HAFM,HAFN,TCHA,VSIZH,VSIZV,RSIZ,BLKS,DSIZ,
     C  VDIMH,VDIMV,VFIL,MIPS
      INTEGER ATOP,DTOP,ADIR( DSIZ),DIR( DSIZ)
      INTEGER ERRNUM, X, Y, OUT(HAFN), STA(9)
      LOGICAL*1 STATUS(HAFN,HAFM)
      COMMON/DIM/NCOL,NROW,HAFM,HAFN,TCHA,VSIZH,VSIZV,RSIZ,NREC,BLKS,
     # DSIZ,VDIMH,VDIMV,VFIL,MIPS
      COMMON // ATOP,DTOP
      COMMON/PIX/N,M,MAXMN,NEXTB
      DATA STA/1HN,1HA,1HD,1HT,1HE,1HA,1HD,1HT,1H*/
C
      IF( ERRNUM .LT. 1 .OR. ERRNUM .GT. 10 )GO TO 50
C
      GO TO(1,2,3,4,5,6,7,8,9,10),ERRNUM
1     WRITE (6,100) MAXMN
      STOP
2     WRITE(6,101)
      INDX = 0
      DO 41 Y=1, HAFM
         DO 40 X=1,HAFN
            INDX = STATUS(X,Y)
            IF (INDX .GE. 0 .AND. INDX .LE. 7) GO TO 40
            INDX = 8
            OUT(X) = STA(INDX+1)
   40    CONTINUE
         WRITE(6,600) Y,OUT
   41 CONTINUE
      STOP
3     WRITE(6,102)
      STOP
4     WRITE(6,103)
      STOP
5     WRITE(6,104)
      STOP
6     WRITE(6,105)
      STOP
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
7       WRITE(6,106)
        STOP
8       WRITE(6,107)
        STOP
9       WRITE(6,108)
        STOP
10      WRITE(6,109)
        STOP
C
50      WRITE(6,51)
        STOP 0
C
C ******************************************************************************
C ******************************************************************************
C
        ENTRY ERRMSG ( X,Y )
C
        WRITE(6,110) X, Y
C
C       ERRTRA IS IBM'S SUBROUTINE TRACEBACK ROUTINE
        CALL ERRTRA
        STOP
C
51      FORMAT('0ERROR NUMBER PASSED TO ERR IS LESS THAN 0 OR GREATER THAN
     * 10')
100     FORMAT('0TOPCPL GREATER THAN ',I3)
101     FORMAT('0TOPCPL LESS THAN ZERO',//)
102     FORMAT('0EOF OR NO INPUT DATA')
103     FORMAT('0DIR LIST OVERFLOW')
104     FORMAT('0POINTER TO TOP OF DIRECTIONAL LIST HAS GONE NEGATIVE')
105     FORMAT('0CONFLICT OF DIRECTIONS ON DIRECTION LIST')
106     FORMAT('0DIRECTIONAL LIST OVERFLOW')
107     FORMAT('0ATOP GREATER THAN DCNT')
108     FORMAT('0EOF DETECTED ON THE DIR FILE')
109     FORMAT('0    X AND/OR Y > &PDIM AND/OR X AND/OR Y < 0')
110     FORMAT(' ERROR IN PIXEL X=',I4,'  Y=',I4)
600     FORMAT(' ROW',I4,2X,100I1)
        END
```

```
      SUBROUTINE GETPIX (X, Y, IPDONE, ADIR, DIR, BLOBM, IPMEAN, IPSSQ,
     C STATUS, VMEM, BKSTAT, OUT)
C
C       WRITTEN BY PETER C. MILLER IN THE FALL AND WINTER OF 1974
C
C     THIS SUBROUTINE LOOKS FOR PIXEL GROUPS TO ADD TO THE
C     CONTOUR. THE STATEMENT NUMBER GROUPS (X00,X25,X50,X75 WHERE X IS 1,
C       2,3, OR 4) REPRESENT THE DIFFERENT DIRECTION OF ENTERING
C     PIXEL GROUP. THE ARRAY 'ADIR' SERVES AS A LIST OF THESE DIRECTIONS
C     THAT WERE TAKEN TO TRACE OUT THE CONTOUR. THE ARRAY 'DIR' IS A
C     STRING THAT IS LET OUT AS THE CONTOUR IS TRACED OUT SO THAT IF WE
C     HAVE TO BACKUP WE CAN FIND OUR WAY BACK. THE FUNCTION COMPAR IS USED
C     TO COMPARE THE BLOB PIXEL GROUP (PRESENTLY THIS INCLUDES ONLY THE IP
C       PIXEL GROUP BUT COULD BE MADE TO INCLUDE ALL GROUPS THAT
C     CAN BE ADDED TO THE BLOB GROUP.
C
C               (X,Y) ARE COORDINATE PAIRS.
C         X = ON ENTRY CONTAINS PIXEL GROUP TO START LOOKING FROM
C                 ON EXIT IT CONTAINS NEXT GROUP TO ADD TO THE BLOB
C                 IF ONE EXISTS.
C         Y = SAME AS X BUT Y COORDINATE
C         IPDONE = FLAG TO TELL MAIN PROGRAM WE ARE DONE WITH THIS CONTOUR
C
C         IPX = THE X COORDINATE OF THE CURRENT INITAL POINT (IP)
C         IPY = THE Y COORDINATE OF THE CURRENT IP
C         IPMEAN = THE MEAN OF THE CURRENT IP
C         IPSSQ = THE SUM OF THE SQUARES OF THE PIXEL GROUP IN THE IP
C         IPCNT = RUNNING SUM OF THE NUMBER OF IPS
C         EXT = BOOLEAN FLAG TO INDICATE INTERNAL OR EXTERNAL CONTOUR
C                 TRUE = EXTERNAL
C                 FALSE = INTERNAL
C
C       XLAST = THE X COORDINATE OF THE PIXEL GROUP AT THE FRONT OF
C               OUR CURRENT CONTOUR.  (USUALLY THE LAST GROUP FOUND BY
C               GETPIX , EXCEPT IF WE HAD TO BACK UP )
C       YLAST = THE Y COORDINATE OF THE PIXEL GROUP AT THE FRONT OF
C               OUR CURRENT CONTOUR.  (USUALLY THE LAST GROUP FOUND BY
C               GETPIX , EXCEPT IF WE HAD TO BACK UP )
C
C         BLOBM = HAS THE MEAN OF THE CURRENT BLOB GROUP
C         BLOBN = HAS THE NUMBER OF PIXELS IN THE CURRENT BLOB
C
C         ADIR = A LIST OF ALL THE DIRECTIONALS FOR THIS IP
C         ATOP = POINTS TO THE TOP ENTRY IN THE ACTUAL DIRECTIONAL LIST
C         DIR = STACK USED TO ALLOW THE PROGRAM TO BACKUP IN ITS TRACING
C                OF CONTOURS WHEN ITS FORWARD PATH IS BLOCKED
C         DTOP = POINTS TO THE TOP OF THE DIRECTION STACK
C
C         N = THE HORIZONTAL DIMENSION OF THE PICTURE MATRIX IN PIXELS
C         M = THE VERTICAL DIMENSION OF THE PICTURE MATRIX IN PIXELS
C                 (M AND N MUST BE MULTIPLES OF 2.)
C
C ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
      REAL IPMEAN(TCHA), IPSSQ(TCHA)
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
      INTEGER HAFM,HAFN,TCHA,VSIZH,VSIZV,RSIZ,BLKS,DSIZ,
     C  VDIMH,VDIMV,VFIL
      INTEGER XLAST,YLAST
      INTEGER BLOBN
      INTEGER BKSTAT(BLKS), OUT(HAFN)
      INTEGER ATOP,DTOP,ADIR( DSIZ),DIR( DSIZ)
      INTEGER X,Y
      LOGICAL EXT
      LOGICAL COMPAR,IPDONE
      LOGICAL*1 STATUS(HAFN,HAFM)
      DIMENSION VMEM (VSIZV,VSIZH,4,TCHA)
      DIMENSION BLOBM(TCHA)
      COMMON/DIM/NCOL,NROW,HAFN,HAFM,TCHA,VSIZH,VSIZV,RSIZ,NREC,BLKS,
     # DSIZ,VDIMH,VDIMV,VFIL,MIPS
      COMMON /IPL/ IPX,IPY,IPCNT,EXT
      COMMON /LAST/ XLAST,YLAST
      COMMON/BLOB1/BLOBN
      COMMON // ATOP,DTOP
      COMMON/PIX/N,M,MAXMN,NEXTB
C
1     XLAST=X
      YLAST=Y
      LDIR=ADIR(ATOP)
      GO TO(100,200,300,400),LDIR
C
C        LOOK ON THE LEFT SIDE FOR NEXT PIXEL GROUP
  100 IF( YLAST .LE. IPY )GO TO 125
      Y=YLAST-2
      IF ( COMPAR(X,Y,ADIR,DIR,BLOBM,IPMEAN,IPSSQ,STATUS,VMEM,BKSTAT)  )
     C  GO TO 500
      Y=YLAST
C
C        LOOK AHEAD FOR THE NEXT PIXEL GROUP
  125 IF( XLAST .GE. N-1 )GO TO 150
      X=XLAST+2
      IF ( COMPAR(X,Y,ADIR,DIR,BLOBM,IPMEAN,IPSSQ,STATUS,VMEM,BKSTAT)  )
     C  GO TO 500
      X=XLAST
C
C        LOOK TO THE RIGHT SIDE FOR THE NEXT PIXEL GROUP
  150 IF( YLAST .GE. M-1 )GO TO 175
      Y=YLAST+2
      IF ( COMPAR(X,Y,ADIR,DIR,BLOBM,IPMEAN,IPSSQ,STATUS,VMEM,BKSTAT)  )
     C  GO TO 500
      Y=YLAST
C
C       THE IF TEST CATCHES CONTOURS MADE UP OF ONLY AN IP
C        HAVE TO BACKUP CAN'T FIND A PIXEL GROUP THAT WILL
C        PASS THE F-TEST AND T-TEST
  175 IF( ATOP .EQ. 1 )GO TO 510
      CALL PSTAT(XLAST,YLAST,ADIR(ATOP),3,STATUS)
      DTOP=DTOP-1
      IF(DTOP.LT.0) CALL ERR(5,ADIR,DIR,STATUS,OUT)
      XLAST=XLAST-2
      ATOP=ATOP+1
      IF(ATOP .GE. DSIZ) CALL ERR(7,ADIR,DIR,STATUS,OUT)
      ADIR(ATOP)=3
      X=XLAST
      IF( DTOP .EQ. 1 )GO TO 510
      LDIR=DIR(DTOP)
      GO TO(150,225,600,475),LDIR
C
  200 IF( XLAST .GE. N-1 )GO TO 225
      X=XLAST+2
      IF ( COMPAR(X,Y,ADIR,DIR,BLOBM,IPMEAN,IPSSQ,STATUS,VMEM,BKSTAT)  )
     C  GO TO 500
      X=XLAST
C
  225 IF( YLAST .GE. M-1 )GO TO 250
      Y=YLAST+2
      IF ( COMPAR(X,Y,ADIR,DIR,BLOBM,IPMEAN,IPSSQ,STATUS,VMEM,BKSTAT)  )
     C  GO TO 500
      Y=YLAST
C
  250 IF( XLAST .LE. 2 ) GO TO 275
      X=XLAST-2
      IF ( COMPAR(X,Y,ADIR,DIR,BLOBM,IPMEAN,IPSSQ,STATUS,VMEM,BKSTAT)  )
     C  GO TO 500
      X=XLAST
C
  275 CALL PSTAT (XLAST, YLAST, ADIR(ATOP), 4, STATUS)
      DTOP=DTOP-1
      IF(DTOP.LT.0) CALL ERR(5,ADIR,DIR,STATUS,OUT)
      YLAST=YLAST-2
      ATOP=ATOP+1
      IF(ATOP .GE. DSIZ) CALL ERR(7,ADIR,DIR,STATUS,OUT)
      ADIR(ATOP)=4
      Y=YLAST
      IF( DTOP .EQ. 1 ) GO TO 510
      LDIR=DIR(DTOP)
      GO TO(175,250,325,600),LDIR
C
  300 IF( YLAST .GE. M-1 ) GO TO 325
      Y=YLAST+2
      IF ( COMPAR(X,Y,ADIR,DIR,BLOBM,IPMEAN,IPSSQ,STATUS,VMEM,BKSTAT)  )
     C  GO TO 500
      Y=YLAST
C
  325 IF( XLAST .LE. 2 )GO TO 350
      X=XLAST-2
      IF ( COMPAR(X,Y,ADIR,DIR,BLOBM,IPMEAN,IPSSQ,STATUS,VMEM,BKSTAT)  )
     C  GO TO 500
      X=XLAST
C
  350 IF( YLAST .LE. IPY )GO TO 375
      Y=YLAST-2
      IF ( COMPAR(X,Y,ADIR,DIR,BLOBM,IPMEAN,IPSSQ,STATUS,VMEM,BKSTAT)  )
     C  GO TO 500
      Y=YLAST
C
```

```
  375 CALL PSTAT (XLAST, YLAST, ADIR(ATOP), 1, STATUS)
      DTOP=DTOP-1
      IF(DTOP.LT.0) CALL ERR(5,ADIR,DIR,STATUS,OUT)
      XLAST=XLAST+2
      ATOP=ATOP+1
      IF(ATOP .GE. DSIZ) CALL ERR(7,ADIR,DIR,STATUS,OUT)
      ADIR(ATOP)=1
      X=XLAST
      IF( DTOP .EQ. 1 )GO TO 510
      LDIR=DIR(DTOP)
      GO TO(600,275,350,425),LDIR
C
  400 IF( XLAST .LE. 2 )GO TO 425
      X=XLAST-2
      IF ( COMPAR(X,Y,ADIR,DIR,BLOBM,IPMEAN,IPSSQ,STATUS,VMEM,BKSTAT)   )
     C  GO TO 500
      X=XLAST
C
  425 IF( YLAST .LE. IPY )GO TO 450
      Y=YLAST-2
      IF ( COMPAR(X,Y,ADIR,DIR,BLOBM,IPMEAN,IPSSQ,STATUS,VMEM,BKSTAT)   )
     C  GO TO 500
      Y=YLAST
C
  450 IF( XLAST .GE. N-1 )GO TO 475
      X=XLAST+2
      IF ( COMPAR(X,Y,ADIR,DIR,BLOBM,IPMEAN,IPSSQ,STATUS,VMEM,BKSTAT)   )
     C   GO TO 500
      X=XLAST
C
  475 CALL PSTAT (XLAST, YLAST, ADIR(ATOP), 2, STATUS)
      DTOP=DTOP-1
      IF(DTOP,LT,0) CALL ERR(5,ADIR,DIR,STATUS,OUT)
      YLAST=YLAST+2
      ATOP=ATOP+1
      IF(ATOP .GE. DSIZ) CALL ERR(7,ADIR,DIR,STATUS,OUT)
      ADIR(ATOP)=2
      Y=YLAST
      IF( DTOP .EQ. 1) GO TO 510
      LDIR=DIR(DTOP)
      GO TO(125,600,375,450),LDIR
C
  500 IF( X .NE. IPX )GO TO 520
      IF( Y .NE. IPY )GO TO 520
      IPDONE=.TRUE.
      GO TO 523
  520 IPDONE=.FALSE.
  523 ATOP=ATOP+1
      IF(ATOP .GE. DSIZ) CALL ERR(7,ADIR,DIR,STATUS,OUT)
      DTOP=DTOP+1
C
      IF(DTOP .GT. DSIZ) CALL ERR(4, ADIR, DIR, STATUS, OUT)
C
C  WE NOW PUT THE DIRECTION WE LEAVE LAST GROUP IN "ADIR" AND "DIR"
      IF( X .NE. XLAST )GO TO 530
      IF( Y .GT. YLAST )GO TO525
      ADIR(ATOP)=4
      DIR(DTOP)=4
      RETURN
C
  525 DIR(DTOP)=2
      ADIR(ATOP)=2
      RETURN
C
  530 IF( X .GT. XLAST )GO TO 540
      DIR(DTOP)=3
      ADIR(ATOP)=3
      RETURN
C
  540 DIR(DTOP)=1
      ADIR(ATOP)=1
      RETURN
C
C        THIS ERR IS AN IMPOSSIBLE CONDITION AND SHOULD NEVER HAPPEN
  600 CALL ERR(6)
C
C        WE HAVE BACKED UP TO THE IP AND MUST
C        TELL MAIN PROGRAM WE ARE DONE WITH THIS CONTOUR
  510 IPDONE=.TRUE.
      RETURN
      END
      INTEGER FUNCTION GSTAT (X, Y, STATUS)
C     ENTRY POINTS GSTAT, PSTAT, SET
C
C              LISTED BELOW ARE ALL POSSIBLE STATUS VALUES
C        N = MEANS NOT ON ANY CONTOUR BOUNDARY (NOT TO BE CONFUSED WITH
C               THE "N" IN COMMON BLOCK /PIX/ )
C        A = MEANS START OF A CONTOUR WHEN SCANNING LEFT TO RIGHT
C        D = MEANS END OF CONTOUR WHEN SCANNING FROM LEFT TO RIGHT
C        T = MEANS IT'S IN THE MIDDLE SOMEWHERE
C
C             NOT IN CONTOUR      IN CONTOUR
C                 N = 0
C                 A = 1             A = 5
C                 D = 2             D = 6
C                 T = 3             T = 7
C                ( NOTE N=4 IS AN IMPOSSIBLE CONDITION )
C              THE STATUS IS CALCULATED BY FINDING THE PRESENT STATUS
C              IN THE PRESENT PASS TABLE, THEN USING THIS VALUE IN
C              THE NEW STATUS TABLE ALONG WITH THE PAST STATUS VALUE,
C
C                PRESENT PASS STATUS TABLE
C
C                               DIRECTION OUT OF
C                               THE PIXEL GROUP
C                                *  UP   * DOWN  *
C                                *  OR   *  OR   *
C                                * RIGHT * LEFT  *
C                        ************************
C                         UP    *       *       *
C              DIRECTION  OR    *   A   *   T   *
C                INTO    LEFT   *       *       *
C                THE     ************************
C               PIXEL    DOWN   *       *       *
C               GROUP     OR    *   T   *   D   *
C                        RIGHT  *       *       *
C                        ************************
C
C                          NEW STATUS TABLE
C
C                          PRESENT PASS STATUS
C                        *       *       *       *
C                        *   A   *   D   *   T   *
C                ********************************* 
C                        *       *       *       *
C                    N   *   A   *   D   *   T   *
C                *********************************
C     PREVIOUS           *       *       *       *
C                    A   *   A   *   T   *   A   *
C       PASS      *********************************
C                        *       *       *       *
C      STATUS        D   *   T   *   D   *   D   *
C                *********************************
C                        *       *       *       *
C                    T   *   A   *   D   *   T   *
C                *********************************
```

```
C :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      IMPLICIT INTEGER*4 (A-Z)
      INTEGER*4 PRESTA(4,4), NEWSTA(4,3)
      LOGICAL*1 STATUS(HAFN,HAFM)
      COMMON/DIM/NCOL,NROW,HAFM,HAFN,TCHA,VSIZH,VSIZV,RSIZ,NREC,BLKS,
     # DSIZ,VDIMH,VDIMV,VFIL,MIPS
C
      DATA PRESTA/ 3, 3, 1, 1, 2, 2, 3, 3, 2, 2, 3, 3, 3, 3, 1, 1/
      DATA NEWSTA/ 1, 1, 3, 1, 2, 3, 2, 2, 3, 1, 2, 3/
      DATA CBIT /4/
C
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C-------------------------------------------------------------------
C
C     ENTRY GSTAT GETS THE CURRENT  STATUS OF THE PIXEL GROUP
C     LOCATION (X,Y) WHERE X IS THE COLUMN NUMBER AND
C     Y IS THE ROW NUMBER OF THE DESIRED LOCATION.
C
      GSTAT = STATUS(X/2+1,Y/2+1)
      RETURN
C
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C-------------------------------------------------------------------
C
      ENTRY PSTAT (X, Y, IN, OUT, STATUS)
C
C     ENTRY PSTAT STORES THE NEW STATUS OF PIXEL GROUP
C     LOCATED AT (X,Y)
C
      IGSTAT = STATUS(X/2+1,Y/2+1)
      PSTAT = PRESTA( IN,OUT )
      IGSTAT = NEWSTA( IGSTAT+1,PSTAT )
      STATUS (X/2+1,Y/2+1) = IGSTAT
      RETURN
C
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C-------------------------------------------------------------------
C
      ENTRY SET (X, Y, STATUS)
C
C     ENTRY SET SETS THE INCONTOUR BIT ON FOR PIXEL GROUP (X,Y)
C
      BITCHK = STATUS(X/2+1,Y/2+1)
      LGRSLT=LOR(CBIT,BITCHK)
      STATUS(X/2+1,Y/2+1) = LGRSLT
      SET = 0
      RETURN
      END
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
      LOGICAL FUNCTION IPCPAR (X, Y, XCPL, YCPL, VMEM, BKSTAT)
C
C        THIS FUNCTION COMPARES TWO PIXEL GROUPS TOGETHER TO SEE IF
C        THEY ARE STATISTICALLY DIFFERENT ENOUGH FOR (X, Y) TO BE
C        A NEW IP.
C
C     WRITTEN BY PETER C. MILLER IN THE FALL AND WINTER OF 1974
C     MODIFIED BY JULIA M. HODGES, DECEMBER 1977
C
C           TOPCPL = TOP OF THE COMPARSON POINTER LIST
C           XCPL = X COORDINATE OF THE PIXEL GROUPS IN THE CPL
C           XCPL = Y COORDINATE OF THE PIXEL GROUPS IN THE CPL
C        FVAL = THE CURRENT F-TEST VALUE
C        TVAL = THE CURRENT T-TEST VALUE
C
C  *************************************************************************
C
      REAL MEAN
      INTEGER XCPL(MAXMN),YCPL(MAXMN),TOPCPL
      INTEGER HAFM,HAFN,TCHA,VSIZH,VSIZV,RSIZ,BLKS,DSIZ,
     C  VDIMH,VDIMV,VFIL
      INTEGER X,Y,GSTAT, CHAN
      INTEGER BKSTAT(BLKS)
      DIMENSION VMEM (VSIZV,VSIZH,4,TCHA)
      COMMON/DIM/NCOL,NROW,HAFM,HAFN,TCHA,VSIZH,VSIZV,RSIZ,NREC,BLKS,
     # DSIZ,VDIMH,VDIMV,VFIL,MIPS
      COMMON/PIX/N,M,MAXMN,NEXTB
      COMMON/CPL/TOPCPL
      COMMON /LEVELS/ FVAL,TVAL,NCHA
C
      IPCPAR=.FALSE.
C
      DO 1 CHAN=1,NCHA
         XYM = MEAN(X,Y,CHAN,VMEM,BKSTAT)
         XLVLM = MEAN(XCPL(TOPCPL),YCPL(TOPCPL),CHAN,VMEM,BKSTAT)
         F1 = SSQ(XCPL(TOPCPL),YCPL(TOPCPL),CHAN,VMEM,BKSTAT)
     #         - 4.0*XLVLM*XLVLM
         F2 = SSQ(X,Y,CHAN,VMEM,BKSTAT) - 4.0*XYM*XYM
         IF (F1 .LE.0.0 .OR. F2 .LE. 0.0) RETURN
         T = (XLVLM - XYM) * SQRT(12.0/(F1+F2))
         IF (ABS(T) .GE. TVAL) RETURN
         F = F1/F2
         IF (F .GE. FVAL .OR. 1.0/F .GE.FVAL) RETURN
    1 CONTINUE
C
      IPCPAR=.TRUE.
      RETURN
      END
      REAL FUNCTION MEAN (X, Y, CHAN, VMEM, BKSTAT)
C
C        THIS SUBROUTINE COMPUTES THE MEAN OF A PIXEL GROUP
C     WRITTEN BY PETER C. MILLER IN THE FALL AND WINTER OF 1974
C :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      INTEGER BKSTAT(BLKS)
      INTEGER X,Y,CHAN
      INTEGER MAFM,MAFN,TCHA,VSIZH,VSIZV,RSIZ,BLKS,DSIZ,
     C  VDIMH,VDIMV,VFIL,MIPS
      DIMENSION VMEM (VSIZV,VSIZH,4,TCHA)
      COMMON/DIM/NCOL,NROW,MAFM,MAFN,TCHA,VSIZH,VSIZV,RSIZ,NREC,BLKS,
     # DSIZ,VDIMH,VDIMV,VFIL,MIPS
      COMMON/PIX/N,M,MAXMN,NEXTS
      COMMON /IPL/ IPX,IPY,IPCNT,EXT
C
C
      MEAN = ( PIXEL(X,Y,CHAN,VMEM,BKSTAT) +
     C         PIXEL(X+1,Y,CHAN,VMEM,BKSTAT)  +
     C         PIXEL(X,Y+1,CHAN,VMEM,BKSTAT)  +
     C         PIXEL (X+1,Y+1,CHAN,VMEM,BKSTAT)  ) / 4.0
C
      RETURN
      END
      SUBROUTINE NEWIP(DONE,XCPL,YCPL,STATUS,OUT,ADIR,DIR,VMEM,BKSTAT)
C
C     THIS SUBROUTINE LOCATES THE INITIAL POINT FOR
C     A NEW CONTOUR.
C
C     WRITTEN BY PETER C. MILLER IN THE FALL AND WINTER OF 1974
C     MODIFIED BY JULIA M. HODGES, DECEMBER 1977
C
C            TOPCPL = TOP OF THE COMPARISON POINTER LIST
C        XCPL = X COORDINATE OF THE PIXEL GROUPS IN THE CPL
C        XCPL = Y COORDINATE OF THE PIXEL GROUPS IN THE CPL
C      N = THE HORIZONTAL DIMENSION OF THE PICTURE MATRIX IN PIXELS
C      M = THE VERTICAL DIMENSION OF THE PICTURE MATRIX IN PIXELS
C              (M AND N MUST BE MULTIPLES OF 2)
C      MAXMN = MAXIMUM OF M AND N
C      IPX = THE X COORDINATE OF THE CURRENT INITAL POINT (IP)
C      IPY = THE Y COORDINATE OF THE CURRENT IP
C      IPMEAN = THE MEAN OF THE CURRENT IP
C      IPSSQ = THE SUM OF THE SQUARES OF THE PIXEL GROUP IN THE IP
C      IPCNT = RUNNING SUM OF THE NUMBER OF IPS
C      EXT = BOOLEAN FLAG TO INDICATE INTERNAL OR EXTERNAL CONTOUR
C               TRUE = EXTERNAL
C               FALSE = INTERNAL
C      DONE = SEE DESCRIPTION IN MAIN ROUTINE
C      TOPCPL = POINTS TO TOP OF THE CPL LISTS
C
C :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
 ******************************************************************
C
      INTEGER ADIR(DSIZ), DIR(DSIZ)
      INTEGER BKSTAT(BLKS), OUT(HAFN)
      INTEGER XCPL(MAXMN),YCPL(MAXMN),TOPCPL
      INTEGER Y,X
      INTEGER START,STAT,GSTAT
      INTEGER HAFM,HAFN,TCHA,VSIZH,VSIZV,RSIZ,BLKS,DSIZ,
     C  VDIMH,VDIMV,VFIL
      LOGICAL EXT,DONE,IPCPAR
      LOGICAL*1 STATUS(HAFN,HAFM)
      DIMENSION VMEM (VSIZV,VSIZH,4,TCHA)
C
      COMMON /IPL/ IPX,IPY,IPCNT,EXT
      COMMON/PIX/N,M,MAXMN,NEXTB
      COMMON/CPL/TOPCPL
      COMMON/DIM/NCOL,NROW,HAFN,HAFM,TCHA,VSIZH,VSIZV,RSIZ,NREC,BLKS,
     # DSIZ,VDIMH,VDIMV,VFIL,NIPS
C
C
      EXT=.FALSE.
      IF( IPX .EQ. N-1 )GO TO 8
      TOPCPL=0
      Y=IPY
C
C       BUILD UP THE CPL TO THE LAST IP
      DO 3 X=1,IPX,2
         STAT = LAND(GSTAT(X,Y,STATUS),3) + 1
         GO TO (3,1,2,3), STAT
```

```
C
C          ADD COORDINATES TO CPL LIST
    1      TOPCPL = TOPCPL +1
           IF (TOPCPL .GT. MAXMN) CALL ERR(1,ADIR,DIR,STATUS,OUT)
           XCPL(TOPCPL) = X
           YCPL(TOPCPL) = Y
           GO TO 3
C
C          DELETE LAST COORDINATES FROM CPL
    2      TOPCPL = TOPCPL - 1
           IF (TOPCPL .LT. 0) CALL ERR(2,ADIR,DIR,STATUS,OUT)
    3 CONTINUE
C
C        NOW REAL SEARCH FOR A NEW IP STARTS
         START=IPX+2
         DO 7 X=START,N,2
           STAT = LAND(GSTAT(X,Y,STATUS),3) + 1
           GO TO (6,4,5,7), STAT
C
C          ADD COORDINATES TO CPL
    4      TOPCPL = TOPCPL + 1
           IF (TOPCPL .GT. MAXMN) CALL ERR(1,ADIR,DIR,STATUS,OUT)
           YCPL(TOPCPL) = Y
           XCPL(TOPCPL) = X
           GO TO 7
C
C          DELETE LAST COORDINATES FROM CPL
    5      TOPCPL = TOPCPL - 1
           IF (TOPCPL .LT. 0) CALL ERR(2,ADIR,DIR,STATUS,OUT)
           GO TO 7
C
C          TEST FOR NEW IP
    6      IF (TOPCPL .EQ. 0) GO TO 15
           IF (.NOT. IPCPAR(X,Y,XCPL,YCPL,VMEM,BKSTAT)) GO TO 14
    7 CONTINUE
C
    8 IF( IPY .GE. M-1 )GO TO 13
         START=IPY + 2
         DO 12 Y=START,N,2
           TOPCPL = 0
           DO 12 X=1,N,2
             STAT = LAND(GSTAT(X,Y,STATUS),3) + 1
             GO TO (9,10,11,12), STAT
C
C            CHECK FOR NEW IP
    9        IF (TOPCPL .EQ. 0) GO TO 15
             IF (.NOT. IPCPAR(X,Y,XCPL,YCPL,VMEM,BKSTAT)) GO TO 14
             GO TO 12
C
C            ADD COORDINATES TO TOP OF CPL
   10        TOPCPL = TOPCPL + 1
             IF (TOPCPL .GT. MAXMN) CALL ERR(1,ADIR,DIR,STATUS,OUT)
             YCPL(TOPCPL) = Y
             XCPL(TOPCPL) = X
             GO TO 12
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```
C
C               DELETE LAST COORDINATES FROM CPL
   11           TOPCPL = TOPCPL - 1
                IF (TOPCPL .LT. 0) CALL ERR(2,ADIR,DIR,STATUS,OUT)
   12 CONTINUE
C
   13 DONE=.TRUE.
      RETURN
   15 EXT=.TRUE.
   14 DONE=.FALSE.
C
C         SETS IPX AND IPY TO COORDINATES OF NEW IP
      IPX=X
      IPY=Y
      RETURN
      END
      REAL FUNCTION PIXEL (X, Y, CHANUM, VMEM, BKSTAT)
C
C     THIS FUNCTION MAKES THE PICTURE LOOK LIKE THE WHOLE
C     THING IN MEMORY AT THE SAME TIME TO THE REST OF THE
C     PROGRAM.  THIS IS ACCOMPLISHED BY KEEPING FOUR SEGMENTS FROM
C     EACH CHANNEL IN ARRAY VMEM.
C
C     JAMES J. BESEMER          18 JULY 1974
C     MODIFIED BY PETER C. MILLER     OCT. 23, 1074
C     MODIFIED BY PETER C. MILLER     MAR. 20, 1976
C
C     BKSTAT - ARRAY OF FLAGS USED TO TELL WHICH SEGMENTS OF
C              PICTURE FILE IS CURRENTLY IN CURE. IS ZERO IF NOT
C              IN CORE, ELSE IT POINTS TO ITS LOCATION IN VMEM.
C     MEM - ARRAY POINTING TO BLOCK IN WHICH HAS SEGMENT WE NEED
C     X - X COORDINATE OF THE DESIRED PIXEL
C     Y - Y COORDINATE OF THE DESIRED PIXEL
C     CHANUM - CHANNEL NUMBER OF THE DESIRED PIXEL
C     BLK   HOLDS THE RECORD NUMBER OF THE RECORD BEING LOADED FROM
C           RANDOM FILE.
C     VMEM - VIRTUAL MEMORY FOR OUR PICTURE
C       N - THE HORIZONTAL DIMENSION OF THE PICTURE MATRIX IN PIXELS
C       M - THE VERTICAL DIMENSION OF THE PICTURE MATRIX IN PIXELS
C                (M AND N MUST BE MULTIPLES OF 2.)
C     MAXMN = (MAX(M,N))/2
C       PIXEL GROUP - IS A GROUP OF FOUR PIXEL ELEMENTS,
C                     WHOSE COORDINATES ARE GIVEN BELOW.
C                     (J,K) , (J,K+1) , (J+1,K) , (J+1,K+1)
C                     THIS DEFINES ONE PIXEL GROUP WHERE J IS DEFINED
C                     BY    J=1,3,5,,,,,N-1 WHERE N IS THE HORIZONTAL
C                     SIZE OF THE PICTURE
C                     AND   K=1,3,5,,,,,M-1 WHERE M IS THE VERTICAL
C                     SIZE OF THE PICTURE
C
C .................................................................
C +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      REAL INITV
      INTEGER HAFM,HAFN,TCHA,VSIZH,VSIZV,RSIZ,BLKS,DSIZ,
     C  VDIMH,VDIMV,VFIL,MIPS
      INTEGER X,Y,CHANUM,BKSTAT(BLKS),CHAN,MEM(4),BLK
      INTEGER PGCNT
      DIMENSION VMEM (VSIZV,VSIZH,4,TCHA)
      COMMON/DIM/NCOL,NROW,HAFM,HAFN,TCHA,VSIZH,VSIZV,RSIZ,NREC,BLKS,
     # DSIZ,VDIMH,VDIMV,VFIL,MIPS
      COMMON/VMEMRY/PGCNT
      COMMON/PIX/N,M,MAXMN,NEXTB
      COMMON /IPL/ IPX,IPY,IPCNT,EXT
C
C     SEE IF X AND Y ARE LEGAL VALUES
      IF( X .LT. 0 .OR. X .GT. N )CALL ERRMSG( X,Y )
      IF( Y .LT. 0 .OR. Y .GT. M )CALL ERRMSG( X,Y )
C
C     CHECK TO SEE CHANUM IS A LEGAL VALUE
      IF( CHANUM .LT. 0 .OR. CHANUM .GT.  TCHA )STOP 11
C
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
C      CALCULATE REQUIRED BLOCK
       IB = ((Y-1)/VSIZV)*VDIMH + ((X-1)/VSIZH) + 1
       KX = MOD( X-1, VSIZH) + 1
       KY = MOD( Y-1, VSIZV) + 1
C
C      SEE IF BLOCK IS IN CORE
       IF( BKSTAT( IB ) .NE. 0 )GO TO 2
C
C      IT WASN'T SO GET IT
       NEXTB = NEXTB + 1
       IF( NEXTB .GT. 3 )NEXTB = 0
       NEXT = NEXTB + 1
       BKSTAT( MEM(NEXT) ) = 0
       BKSTAT( IB ) = NEXT
       MEM( NEXT ) = IB
C
C      READ IN NEEDED SEGMENT OF PICTURE FOR ALL CHANNELS
       DO 4 CHAN=1, TCHA
          BLK = (CHAN - 1) * (BLKS - 1) + IB
          READ(VFIL'BLK) ((VMEM(I,J,NEXT,CHAN),I=1,VSIZV),J=1,VSIZH)
     4 CONTINUE
C
C      INCREMENT THE PAGE FAULT COUNT
       PGCNT = PGCNT + 1
     2 PIXEL = VMEM(KY,KX,BKSTAT(IB),CHANUM)
       RETURN
C
C.......................................................................
C-----------------------------------------------------------------------
C
       ENTRY INITV (X, VMEM, BKSTAT)
C
C      INITALIZE VIRTUAL SYSTEM
C
       DO 20 I=1, BLKS
          BKSTAT(I) = 0
    20 CONTINUE
C
       DO 21 I=1,4
          MEM(I) = BLKS
    21 CONTINUE
       NEXTB = 0
C
C      THIS IS JUST TO AVOID A DIAGNOSTIC
       INITV = 0
       RETURN
       END
```

```
      SUBROUTINE SICB (ADIR, STATUS)
C
C     SICB - SET IN CONTOUR BIT
C     WHICH IS THE HIGH ORDER BIT IN EACH THREE-BIT STATUS GROUP.
C     THIS MEANS THAT THIS PIXEL GROUP IS IN THE BOUNDARY OF
C     SOME CONTOUR.
C     WRITTEN BY PETER C. MILLER IN THE FALL AND WINTER OF 1974
C
C       ADIR - A LIST OF ALL THE DIRECTIONALS FOR THIS IP
C       ATOP - POINTS TO THE TOP ENTRY IN THE ACTUAL DIRECTIONAL LIST
C       DIR - STACK USED TO ALLOW THE PROGRAM TO BACKUP IN ITS TRACING
C               OF CONTOURS WHEN ITS FORWARD PATH IS BLOCKED
C       DTOP - POINTS TO THE TOP OF THE DIRECTION STACK
C       IPX - THE X COORDINATE OF THE CURRENT INITAL POINT (IP)
C       IPY - THE Y COORDINATE OF THE CURRENT IP
C
C ********************************************************************
C
C
      INTEGER X,Y,DIRECT,GSTAT
      INTEGER ATOP,DTOP,ADIR( DSIZ)
      INTEGER HAFM,HAFN,TCHA,VSIZH,VSIZV,RSIZ,BLKS,DSIZ,
     C  VDIMH,VDIMV,VFIL,MIPS
      LOGICAL*1 STATUS(HAFN,HAFM)
      COMMON/DIM/NCOL,NROW,HAFM,HAFN,TCHA,VSIZH,VSIZV,RSIZ,NREC,BLKS,
     # DSIZ,VDIMH,VDIMV,VFIL,MIPS
      COMMON // ATOP,DTOP
      COMMON /IPL/ IPX,IPY,IPCNT,EXT
C
      X=IPX
      Y=IPY
C
      DO 8 DIRECT=2,ATOP
         CALL SET (X,Y,STATUS)
         ITMP = ADIR(DIRECT)
         GO TO (4,5,6,7), ITMP
    4    X = X + 2
         GO TO 8
C
    5    Y = Y + 2
         GO TO 8
C
    6    X = X - 2
         GO TO 8
C
    7    Y = Y - 2
    8 CONTINUE
C
      RETURN
      END
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
      REAL FUNCTION SSQ (X, Y, CHAN, VMEM, BKSTAT)
C
C     THIS SUBROUTINE COMPUTES THE SUM OF THE SQUARES FOR A PIXEL GROUP
C     WRITTEN BY PETER C. MILLER IN THE FALL AND WINTER OF 1974
C
C    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
C ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      INTEGER HAFM,HAFN,TCHA,VSIZH,VSIZV,RSIZ,BLKS,DSIZ,
     C  VDIMH,VDIMV,VFIL,MIPS
      INTEGER X,Y,CHAN
      INTEGER BKSTAT(BLKS)
      DIMENSION VMEM (VSIZV,VSIZH,4,TCHA)
      COMMON/DIM/NCOL,NROW,HAFM,HAFN,TCHA,VSIZH,VSIZV,RSIZ,NREC,BLKS,
     # DSIZ,VDIMH,VDIMV,VFIL,MIPS
      COMMON/PIX/N,M,MAXMN,NEXTB
      COMMON /IPL/ IPX,IPY,IPCNT,EXT
C
      SSQ=0.0
      DO 1 I=1,2
         DO 1 J=1,2
            TEMP = PIXEL(X+J-1, Y+I-1, CHAN, VMEM, BKSTAT)
            SSQ = SSQ + TEMP*TEMP
    1 CONTINUE
C
      RETURN
      END
```

BLOB-3
RECONSTRUCTION OF PICTURE

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

I. **NAME**

RECON

II. **DESCRIPTION**

The RECON program reconstructs the picture from the list description generated by BLOB.

III. **CALLING SEQUENCE**

Call RECON (ADIR, SLINE, STATUS, PMEANS, BOUND, IPBS)

where

ADIR is the list of directionals for the current initial point.

SLINE is a line buffer for the reconstructed picture.

STATUS is the array holding the status flags for each pixel group.

PMEANS is the array holding the gray level information for each of the pictures.

BOUND is the array in which the boundaries of each contour are traced out.

IPBS is the initial point boundary stack used to keep track of which boundary contains a given point.

IV. **INPUT/OUTPUT**

1. INPUT
   a. Sequential data set on unit number 8 which contains a list of initial point coordinates for each contour, the number of points in the contour, the number of entries in the associated directional list, and the means for the initial point.

   b. Sequential data set on unit 9 which contains directional lists for the contours.

   c. COMMON/DIM/NCOL, NROW, HAFN, HAFM, TCHA, VSIZV, VSIZH, RSIZ, NREC, BLKS, DSIZ, VDIMV, VDIMH, VFIL, MIPS as described for the subroutine REFORM.

d. COMMON/PIX/N, M, MAXMN

where

N is the number of pixels per line in the picture.

M is the number of lines in the picture.

MAXMN is the maximum of N and M.

2. OUTPUT

RECON generates a picture showing the boundaries of the detected contours in byte format. The contours are represented by the value 255, the inside of the regions by the value zero. This picture is written one line at a time to unit 10.

RECON also reconstructs the picture by filling the inside of each region with its mean value in each component (channel), and each pixel component is represented by a byte value. This reconstructed image is written to unit 13.

3. FILE STORAGE

No additional files are used by this program.

V. <u>DESCRIPTION OF SUBROUTINES</u>

The following table gives the storage requirements and the functions of subroutines used by RECON.

DESCRIPTION OF SUBROUTINES FOR RECON

SUBROUTINE NAME	STORAGE (BYTES)	FUNCTION
RECON	2978	Read initial point and directional files, generate contour pictures.
GSTAT	1036	Gets the current status of a pixel group.
PSTAT	Entry Under GSTAT	Stores current status of pixel group.

The following table shows the linkage of the subroutines.

LINKAGE OF SUBROUTINES FOR RECON

CALLING PROGRAM	PROGRAM CALLED
RECON	GSTAT PSTAT

VI. PERFORMANCE SPECIFICATIONS

1. STORAGE

   The subroutine RECON is 2970 bytes. The storage needed to run this subroutine, including the required subroutines and a driver, is 97K for an image of 112 x 112 pixels.

2. EXECUTION TIME

   The time required to run this subroutine to reconstruct a 112 x 112 image is approximately 10 seconds of CPU time.

3. RESTRICTIONS

   None.

VII. METHOD

The output unit number is set to 10. Arrays SLINE, STATUS, and BOUND are set to zero.

Each contour is processed as follows:
The initial point information (coordinates of point, number of pixels in current blob, number of entries in corresponding directional list, means) is read. If the blob consists of more than a single point, the directional list is also read for the region.

The contour is then traced out by following the directional list. The contour picture is produced by setting the associated line buffer position to 255, leaving all points not on a contour equal to zero.

After all contours have been processed, the picture is reconstructed by filling the inside of each region with its mean value in each component (channel). One line is written to the output array for each channel. This is repeated for each row of the picture.

VIII. COMMENTS

The program PLTPIX has been used to print the resulting contour image and the reconstructed image from files 10 and 13, respectively. However, the user may process these files as he desires.

IX. TESTS

The reconstructed images have been examined by use of mean square error calculations and plots and histograms of difference images.

X. LISTINGS

Listings of the subroutines follow.

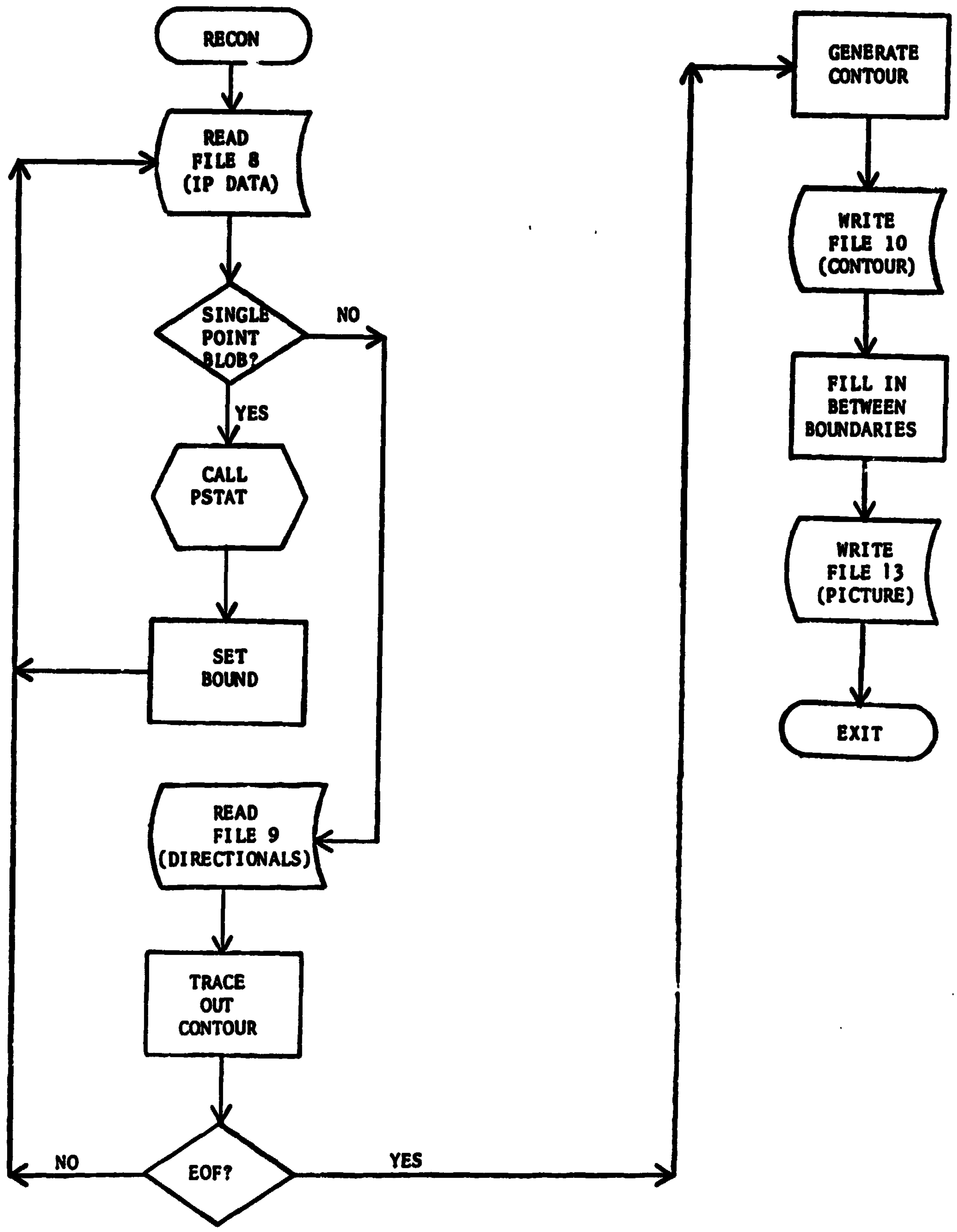


Figure 18. Flow Chart for Subroutine RECON

```
      SUBROUTINE RECON (ADIR,SLINE,TSL,STATUS,PMEANS,BOUND,IPBS,N4)
C
C     THIS PROGRAM IS ONE POSSIBLE WAY OF REBUILDING THE CONTOURED
C     PICTURE, AND THE OUTPUT FROM IT IS EASILY CHANGED.
C
C     WRITTEN BY PETER C. MILLER IN FALL AND WINTER OF 1974
C     MODIFIED FOR OPERATION UNDER OS/MVT ON IBM 360 AND CONTOUR
C     OUTPUT BY HANS G. MOIK, NASA/GSFC CODE 933 IN SEPTEMBER 1976
C
C     BLOBN - NUMBER OF PIXELS IN THE CURRENT BLOB
C     SLINE - LINE BUFFER FOR RECONSTRUCTED PICTURE
C     M, N - VERTICAL, HORIZONTAL DIMENSIONS OF THE PICTURE MATRIX
C                 (M AND N MUST BE MULTIPLES OF 2.)
C     MAXMN = (MAX(M,N))/2
C     IPX - THE X COORDINATE OF THE CURRENT INITIAL POINT (IP)
C     IPY - THE Y COORDINATE OF THE CURRENT IP
C     IPMEAN - THE MEAN OF THE CURRENT IP
C     IPSSQ - THE SUM OF THE SQUARES OF THE PIXEL GROUP IN THE IP
C     IPCNT - RUNNING SUM OF THE NUMBER OF IPS
C     EXT - BOOLEAN FLAG TO INDICATE INTERNAL OR EXTERNAL CONTOUR
C                TRUE = EXTERNAL
C                FALSE = INTERNAL
C     STATUS - ARRAY HOLDING THE STATUS FLAGS FOR EACH
C                PIXEL GROUP. EACH STATUS FLAG IS 1 BYTE LONG
C                AND IS MANIPULATED BY SUBROUTINE GSTAT.
C     PMEANS - ARRAY WHICH HOLDS THE GRAY LEVEL INFO. FOR EACH THE
C             PICTURE.
C     ATOP - TOP OF DIRECTIONAL LIST
C     ADIR - HOLDS THE DIRECTIONALS FOR ONE IP
C     BOUND - THE BOUNDARIES OF EACH CONTOUR ARE TRACED OUT IN THIS
C             ARRAY, POINTERS ARE LEFT BEHIND POINTING INTO THE PMEAN
C             SO WE CAN INDIRECTLY WRITE THE PICTURE.
C     TIPBS - TOP OF IP BOUNDARY STACK
C     IPCNT - RUNNING COUNT OF THE NUMBER OF IP WE HAVE PROCESSED
C     IPBS - IP BOUNDARY STACK USED TO KEEP TRACK OF WHICH BOUNDARY
C           WE ARE INSIDE OF.
C
C  ...............................................................
C  ...............................................................
C
      REAL PMEANS( MIPS, TCHA)
      INTEGER HAFM,HAFN,TCHA,VSIZH,VSIZV,RSIZ,BLKS,DSIZ,
     C  VDIMH,VDIMV,VFIL,MIPS
      INTEGER ATOP,ADIR(DSIZ)
      INTEGER X, Y, TIPBS, START, IPCNT, STAT, IPBS(MAXMN), GSTAT
      INTEGER CHAN, TAPE, BOUND(HAFN,HAFM), BLOBN, TMP
      LOGICAL*1 TSL(N4), SLINE(TCHA,NCOL), STATUS(HAFN,HAFM)
      LOGICAL EXT
      COMMON/DIM/NCOL,NROW,HAFM,HAFN,TCHA,VSIZH,VSIZV,RSIZ,NREC,BLKS,
     * DSIZ,VDIMH,VDIMV,VFIL,MIPS
      COMMON // ATOP,CTOP
      COMMON/BLOB1/BLOBN
      COMMON/IPL/IPX,IPY,IPCNT,EXT
      COMMON/PIX/N,M,MAXMN,NEXTB
C
      TAPE=10
      DO 300 I=1, MAFN
      DO 300 J=1, TCMA
  300 SLINE(J,I)=0
C
      IPCNT=0
      DO 2 Y=1,MAFM
         DO 2 X=1,MAFN
            STATUS(X,Y) = 0
            BOUND(X,Y) = 0
    2 CONTINUE
C
C     READ THE IP INFORMATION FOR ONE REGION
    3 IPCNT = IPCNT + 1
      IF( IPCNT .GT.  MIPS )GO TO 25
      READ(8,END=16)IPX,IPY,BLOBN,ATOP,(PMEANS(IPCNT,CHAN),CHAN=1,TCMA)
C
C     DO WE HAVE A SINGLE POINT IP?
      IF( BLOBN .NE. 4 )GO TO 4
C
C     YES
      CALL PSTAT(IPX,IPY,1,1,STATUS)
      BOUND(IPY/2+1,IPX/2+1)=IPCNT
      GO TO 3
    4 Y = IPY
      X=IPX
C
C     READ THE DIRECTIONAL INFORMATION FOR ONE REGION
      READ(9,END=21)( ADIR(I) , I=1,ATOP )
      LASTD=1
C
C     TRACE OUT THE CONTOUR
      DO 15 NXTDIR=2,ATOP
      CALL PSTAT(X,Y,LASTD,ADIR(NXTDIR),STATUS)
      BOUND(Y/2+1,X/2+1)=IPCNT
      LASTD=ADIR(NXTDIR)
      GO TO(11,12,13,14),LASTD
   11    X = X + 2
         GO TO 15
   12    Y = Y + 2
         GO TO 15
   13    X = X - 2
         GO TO 15
   14    Y = Y - 2
   15 CONTINUE
C
      CALL PSTAT(IPX,IPY,ADIR(ATOP),1,STATUS)
      IF( X .NE. IPX .OR. Y .NE. IPY)WRITE(6,105) IPCNT
C
C     PROCESS NEXT REGION
      GO TO 3
C
C     GENERATE CONTOUR PICTURE
   16 CONTINUE
      DO 333 Y=1,MAFM
         DO 331 X=1,MAFN
```

```
            TMP = BOUND(X,Y)
            IF (TMP .NE. 0) TMP=255
            IX = 2*X - 1
            TSL(IX  )=TMP
            TSL(IX+1)=TMP
  331     CONTINUE
      DO 333 I=1,2
  333 WRITE (TAPE) TSL
C
C     FILL IN BETWEEN THE BOUNDARIES
      DO 20 Y=1,M,2
      TIPBS=0
         DO 20 X=1,N,2
            STAT = GSTAT(X,Y,STATUS) + 1
            GO TO (17,18,19,20),STAT
   17       BOUND(X/2+1,Y/2+1) = IPBS(TIPBS)
            GO TO 20
   18       TIPBS = TIPBS + 1
            IF (TIPBS .GT. MAXMN) GO TO 26
            IPBS(TIPBS) = BOUND(X/2+1,Y/2+1)
            GO TO 20
   19       TIPBS = TIPBS - 1
            IF (TIPBS .GE. 0) GO TO 20
            WRITE(6,101)
   20 CONTINUE
C
C     WRITE OUT THE PICTURE
      DO 33 Y = 1,HAFM
         DO 32 CHAN = 1,TCHA
         DO 31 X= 1,HAFN
            IXB=BOUND(X,Y)
            IF(IXB .EQ. 0) GO TO 30
            TMP = MMEANS(IXB,CHAN) + 0.5
   30       CONTINUE
            IX=2*X-1
            SLINE(CHAN,IX  )=TMP
            SLINE(CHAN,IX+1)=TMP
   31     CONTINUE
   32     CONTINUE
      DO 33 I=1,2
   33 WRITE (13) SLINE
      WRITE(6,109)
      RETURN
C
21    WRITE(6,102)
25    WRITE(6,103) MAXIPS
      RETURN
C
26    WRITE(6,106)
      RETURN
C
100   FORMAT(16I3)
101   FORMAT('0IPBS UNDERFLOW')
102   FORMAT( '0EOF DETECTED ON THE DIRECTIONAL FILE')
103   FORMAT('0IPCNT GT MAX IP COUNT  ',I6)
```

```
105   FORMAT('0 CONTOUR ',I6,' DID NOT RETURN TO ITS INITIAL POINT')
106   FORMAT('0IPBS OVERFLOW')
109   FORMAT('0RECON COMPLETED')
      END
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

CHAPTER V

CLASSIFICATION

SEQUENTIAL LINEAR CLASSIFIER

I. **NAME**

LINEAR, NCLASS

II. **DESCRIPTION**

In this technique, the discriminant functions which are evaluated to determine classes are linear. The decision is based on a positive or negative value of the function of the input vector of reflectances. The discriminants are calculated sequentially, with one class separated from all other classes by each function. The steps required are examining the data to determine which class is separated from the others, and determining the coefficients of the discriminant function.

III. **CALLING SEQUENCE**

Call LINEAR (X, CLASS, W, MOC, S, Y, B, A2)
Call NCLASS (BUFF, MCLASS, CLASS, W, MOC)
where the arrays are dimensioned as follows:
X (NN, MM, NSC) - bytes of training data,
CLASS (MM) - double precision array of class names,
W (NN+1, MM-1) - array of discriminant function coefficients,
MOC (MM) - class numbers in order of testing,
S (NN+1, NN+1) - double precision work array,
Y (MM X NSC) - work array,
B(MM X NSC) - work array,
A2 (NN+1, MM X NSC) - work array,
BUFF (NN, NSAMP) - input buffer (bytes),
MCLASS (NSAMP) - output buffer (bytes),
and
NN - number of channels of data,
MM - number of classes present,
NSC - number of training samples per class,
NSAMP - number of pixels per scan line.

IV. INPUT/OUTPUT

1. INPUT

The input data set is sequential, with data samples arranged by measurement vectors and word length of one byte. The following parameters are specified in the common block:

/CLSSFR/

NN, MM, NRECS, NSAMP, NSC.

2. OUTPUT

The output data set is a file containing the class number for each input pixel.

V. DESCRIPTION OF SUBROUTINES

The subroutine linkages are given in the following table.

CLASSIFIER SUBROUTINES

CALLING PROGRAM	PROGRAMS CALLED
LINEAR	SNOPAL NTEST
SNOPAL	GASINV
NTEST	NOPACA
NCLASS	NOPACA

The subroutines are described in the following table.

SUBROUTINE OR ENTRY	STORAGE (BYTES)	FUNCTION
LINEAR	1154	Calls routines to compute and test discriminant functions.
SNOPAL	4550	Compute coefficients of the discriminant function, and distances from data classes to discriminant hyperplanes.
NTEST	1428	Test discriminants on known data samples.

SUBROUTINE OR ENTRY	STORAGE (BYTES)	FUNCTION
NOPACA	942	Classification of data samples.
NCLASS	1252	Classify the data set, compute and print class occupancies.
GASINV	1838	Invert a symmetric matrix.

VI. PERFORMANCE SPECIFICATIONS

1. STORAGE

The program requires 125K bytes when classifying the Mobile Bay data set which has scan lines of 1200 pixels.

2. EXECUTION TIME

Approximately 1 minute is required to compute the discriminant coefficients. The Mobile Bay data was classified into six classes at the rate of 4590 pixels/second, averaged over several runs.

VII. METHOD

The class which is to be separated from the others should be widely separated from the discriminant hyperplane and from the other classes. The criterion used is the sum of the signed distances of the training data samples from the plane. (Samples which are incorrectly discriminated are given negative distances.) The discriminant planes for each class of training data vs. the other classes are determined by the following method.

The coefficients of the discriminant function are then determined by setting up a system of discriminant equations (one for each training sample). The value of the function for each data sample is the distance of the sample from the plane represented by the function. The method consists of maximizing the total distance of the training samples from the discriminant hyperplane. This process is repeated until a single class remains. Samples are classified by evaluating the discriminant functions sequentially until a positive value is obtained. Flow charts are given in Figures 19 and 20.

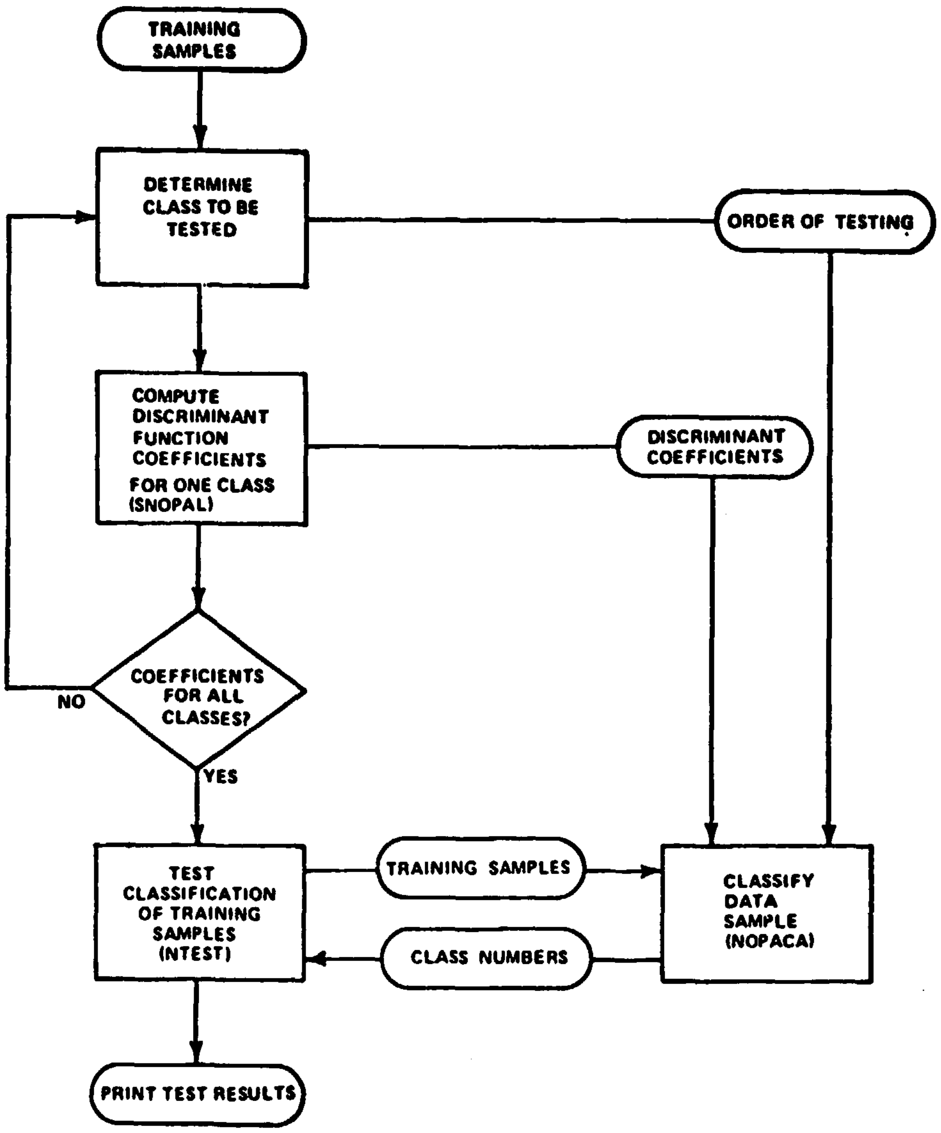

TRAINING SAMPLES
DETERMINE CLASS TO BE TESTED
ORDER OF TESTING
COMPUTE DISCRIMINANT FUNCTION COEFFICIENTS FOR ONE CLASS (SNOPAL)
DISCRIMINANT COEFFICIENTS
COEFFICIENTS FOR ALL CLASSES?
NO
YES
TEST CLASSIFICATION OF TRAINING SAMPLES (INTEST)
TRAINING SAMPLES
CLASSIFY DATA SAMPLE (NOPACA)
CLASS NUMBERS
PRINT TEST RESULTS

Figure 19. Discriminant Training Phase of Sequential Linear Classifier

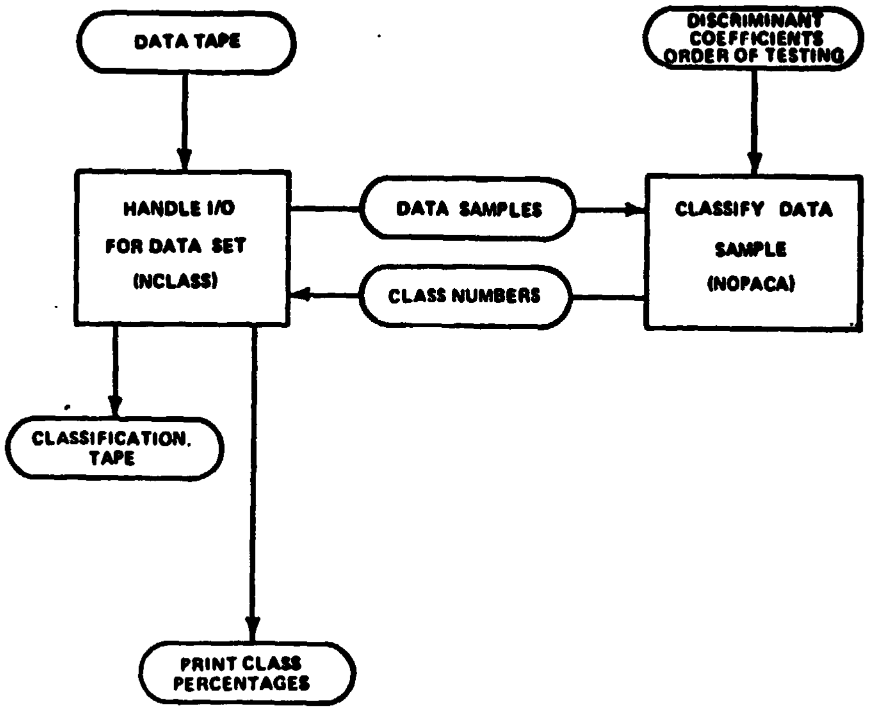

DATA TAPE
DISCRIMINANT COEFFICIENTS ORDER OF TESTING
HANDLE I/O FOR DATA SET (NCLASS)
DATA SAMPLES
CLASSIFY DATA SAMPLE (NOPACA)
CLASS NUMBERS
CLASSIFICATION TAPE
PRINT CLASS PERCENTAGES

Figure 20. Classification Phase of Linear Classifier

VIII. COMMENTS

None.

IX. TESTS

The algorithms have been tested extensively and have been found to be comparable in accuracy to the maximum likelihood method.

X. LISTINGS

Listings of the routines follow.

```
      SUBROUTINE LINEAR (X, CLASS, W, MOC, S, Y, B, A2)
C
C  COMPUTE LINEAR DISCRIMINANTS AND TEST USING TRAINING SAMPLES
C
      DIMENSION X(1), W(1), MOC(1), S(1), Y(1), B(1), A2(1)
      DOUBLE PRECISION CLASS(MM)
      LOGICAL ORDER
      COMMON /DISTNC/ DNW, D, ORDER
      COMMON /CLSSFR/ NN, MM, NRECS, NSAMP, NSC
C
C  COMPUTE THE ORDER OF DISCRIMINANT TESTING AND LINEAR DISCRIMINANTS
C
      NN1 = NN + 1
      MM1 = MM - 1
      DO 100 NC=1,MM
  100 MOC(NC) = NC
C
C     FIND CLASS WITH MAXIMUM DISTANCE FROM HYPERPLANE
      DO 1000 NW1=1,MM1
      WRITE (6,5010)
      IF (NW1.EQ.MM1) GO TO 1000
      WRITE (6,5000)
      ORDER = .TRUE.
      DMAX = -1.0 E 50
      DO 500 NC1=NW1,MM
      CALL SNOPAL (X, CLASS, W, MOC, S, Y, B, A2, NW1, NN1)
      DD = DNW + D
      WRITE (6,5020) CLASS(MOC(NW1)), DNW, D, DD
      IF (DD.LT.DMAX) GO TO 150
      DMAX = DD
      NCLASS = NC1
C
C     ROTATE CLASS NUMBERS IN ARRAY 'MOC'
  150 MSAVE = MOC(NW1)
      DO 200 NC=NW1,MM1
  200 MOC(NC) = MOC(NC+1)
      MOC(MM) = MSAVE
  500 CONTINUE
C
C     PLACE DISCRIMINATED CLASS NUMBER AT TOP OF ARRAY 'MOC'
      ITEMP = MOC(NW1)
      MOC(NW1) = MOC(NCLASS)
      MOC(NCLASS) = ITEMP
      ORDER = .FALSE.
 1000 CALL SNOPAL (X, CLASS, W, MOC, S, Y, B, A2, NW1, NN1)
C
C  TEST THE CLASSIFICATION OF THE TRAINING SAMPLES
C
      CALL NTEST (X, CLASS, W, MOC)
      RETURN
C
 5000 FORMAT (20X,31('*')/20X,'* DATA - HYPERPLANE DISTANCES *'/20X,
     .31('*'))
 5010 FORMAT ('1')
 5020 FORMAT (/A30,5X,'OTHER',5X,'TOTAL'/20X,3F10.4)
      END
      SUBROUTINE SNOPAL (X, CLASS, W, MOC, S, Y, B, A2, NW1, NN1)
C
C  SUPERVISED NON-PARAMETRIC LEARNING
C
      LOGICAL*1 X(NN,MM,NSC)
      DIMENSION W(NN1,1), MOC(MM), Y(1), B(1), A2(NN1,1)
      DOUBLE PRECISION CLASS(MM), S(NN1,NN1), DET
      INTEGER SIJ
      LOGICAL TEST, ORDER
      COMMON /DISTNC/ DNW, D, ORDER
      COMMON /CLSSFR/ NN, MM, NRECS, NSAMP, NSC
      DATA DELTA /0.01/
C
C  COMPUTE INVERSE OF A(TRANSPOSE) A WHERE 'A' IS ALGMENTEC MATRIX
C  OF SAMPLES
C
      DO 130 I=1,NN1
      DO 130 J=1,I
      SIJ = 0
C
      DO 70 NC1=NW1,MM
      NC = MOC(NC1)
      IF (I.EQ.NN1) GO TO 133
C
C     INDEX I NOT EQUAL TO NO. OF BANDS + 1
      DO 132 NS1=1,NSC
  132 SIJ = SIJ + X(I,NC,NS1)*X(J,NC,NS1)
      GO TO 70
C
C     INDEX I EQUAL TO NO. OF BANDS + 1
  133 IF (J.EQ.NN1) GO TO 70
      DO 134 NS1=1,NSC
  134 SIJ = SIJ + X(J,NC,NS1)
   70 CONTINUE
C
      S(I,J) = SIJ
  130 S(J,I) = SIJ
      S(NN1,NN1) = NSC * (MM-NW1+1)
C
      CALL GASINV (S, NN1, DET)
C
C     INITIALIZE W, Y, B, AND A2 ARRAYS
      DO 1112 NFA=1,NN1
 1112 W(NFA,NW1) = 0.0
      NST2 = 0
      DO 1110 NC1=NW1,MM
      NC = MOC(NC1)
      DO 1110 NS1=1,NSC
      NST2 = NST2 + 1
      Y(NST2) =-1.0
      B(NST2) = 1.0
      DO 1110 I=1,NN1
      A2(I,NST2) = S(I,NN1)
      DO 1110 K=1,NN
 1110 A2(I,NST2) = A2(I,NST2) + S(I,K)*X(K,NC,NS1)
```

```
C
C  DO NI ITERATIONS OF THE HO-KASHYAP ALGORITHM, UNLESS ALL COEFFICIENTS
C  CHANGE BY LESS THAN 1 PERCENT
C
      NI = 50
      IF (ORDER) NI = 10
      NW2 = NW1 + 1
      NW = MOC(NW1)
      IF (.NOT.ORDER) WRITE (6,102) NW, CLASS(NW), CLASS(NW)
      DO 1040 INDEX=1,NI
      TEST = .TRUE.
C
C     COMPUTE NN1 COEFFICIENTS FOR CLASS NW AND STORE IN W
      DO 1101 I=1,NN1
      W0 = W(I,NW1)
C
C     USE SAMPLES FROM CLASS NW
      DO 2101 J=1,NSC
 2101 W(I,NW1) = W(I,NW1) + A2(I,J)*ABS(Y(J))
C
C     LOOP OVER REMAINING CLASSES
      J = NSC
      DO 2000 NC1=NW2,MM
      DO 2000 NS1=1,NSC
      J = J + 1
 2000 W(I,NW1) = W(I,NW1) - A2(I,J)*ABS(Y(J))
C
C     TEST COEFFICIENTS FOR CHANGE
      IF (ABS(W(I,NW1)-W0).GT.ABS(DELTA*W0)) TEST = .FALSE.
 1101 CONTINUE
C
C  COMPUTE NEW DISCRIMINANT VALUES AND CLASSIFICATION ERRORS
C
C     DO FOR CLASS NW
      NERR1 = 0
      DNW = 0.0
      DO 1004 I=1,NSC
      IF (Y(I).GT.0.0) B(I) = B(I) + 2.0*Y(I)
      Y(I) = W(NN1,NW1)
      DO 1141 NF2=1,NN
 1141 Y(I) = Y(I) + W(NF2,NW1)*X(NF2,NW,I)
      IF (Y(I).LE.0.0) NERR1 = NERR1 + 1
      DNW = DNW + Y(I)
 1004 Y(I) = Y(I) - B(I)
C
C     LOOP OVER REMAINING CLASSES
      I = NSC
      NERR2 = 0
      D = 0.0
      DO 1005 NC1=NW2,MM
      NC = MOC(NC1)
      DO 1000 NS1=1,NSC
      I = I + 1
      IF (Y(I).GT.0.0) B(I) = B(I) + 2.0*Y(I)
      Y(I) = W(NN1,NW1)
      DO 105 NF2=1,NN
  105 Y(I) = Y(I) + W(NF2,NN1)*X(NF2,NC,NS1)
      IF (Y(I).GT.0.0) NERR2 = NERR2 + 1
      D = D + Y(I)
 1000 Y(I) = -Y(I) - B(I)
 1005 CONTINUE
      DNW = DNW / NSC
      D = -D / (NSC*(MM-NN1))
C
      IF (.NOT.ORDER) WRITE (6,100) INDEX, NERR1, NERR2, (W(NFA,NN1),
     ,NFA=1,NN1)
      IF (TEST) GO TO 1010
 1040 CONTINUE
 1010 NERR = NERR1 + NERR2
      IF (.NOT.ORDER) WRITE (6,220) NERR
C
      IF (NN1.NE.MM-1) RETURN
      WRITE (6,216)
      MM1 = MM - 1
      DO 1220 NW=1,MM1
 1220 WRITE (6,101) MOC(NW), CLASS(MOC(NW)), (W(NFA,NW), NFA=1,NN1)
      WRITE (6,101) MOC(MM), CLASS(MOC(MM))
      RETURN
C
  100 FORMAT (I8,I11,I8,4X,1P7E14.3/(31X,1P7E14.3))
  101 FORMAT (/I13,A10,10X,1P7E14.3/(33X,1P7E14.3))
  102 FORMAT (//22X,22('*')/22X,'* CLASS',I3,A10,' *'/22X,22('*')//' ITE
     ,RATION NO.',5X,'ERRORS',10X,'LINEAR DISCRIMINANT COEFFICIENTS'/
     ,A22,'  OTHER'/)
  216 FORMAT ('1'/10X,'ORDERED CLASSES',20X,'ELEMENTS OF THE DISCRIMINAN
     ,T VECTOR'/10X,15('*'),20X,35('*')/)
  220 FORMAT (/7X,'TOTAL ERRORS',I5)
      END
      SUBROUTINE NTEST (X, CLASS, N, MOC)
C
C  CLASSIFIES KNOWN DATA SAMPLES - NONPARAMETRIC CLASSIFICATION
C
      LOGICAL*1 X(NN,MM,NSC), ICLASS
      DIMENSION N(1), MOC(1), KS(20)
      DOUBLE PRECISION CLASS(MM)
      COMMON /CLSSFR/ NN, MM, NRECS, NSAMP, NSC
C
      NN1 = NN + 1
      MM1 = MM * 1
      WRITE(6,2008) CLASS
      TE = 0.0
C
      DO 1500 NC=1,MM
      DO 1109 I=1,MM
 1109 KS(I) = 0
C
      DO 1400 NS1 = 1,NSC
 1400 CALL NOPACA (X(1,NC,NS1), ICLASS, KS, N, MOC, 1, NN1, MM1)
C
      EFF = 100.0 * KS(NC) / NSC
      WRITE (6,2009) NC, CLASS(NC), NSC, KS(NC), EFF, (KS(N), N=1,MM)
 1500 TE = TE + EFF
      AVE = TE / FLOAT(MM)
      WRITE (6,2104) AVE
      RETURN
C
 2008 FORMAT ('1'/30X,29('*')/30X,'* RESULTS OF CLASSIFICATION *'/30X,'*
     *      TRAINING SAMPLES      *'/30X,29('*')///10X,'NUMBER OF  NUMBER
     *      PERCENT     NUMBER OF SAMPLES CLASSIFIED AS'/6X,'CLASS    SAM
     *PLES   CORRECT    CORRECT',10A9/(43X,10A9))
 2009 FORMAT (/I4,A9,I7,I11,F10.1,10I9/(41X,10I9))
 2104 FORMAT (//30X,18HAVERAGE ACCURACY =,F6.1,8H PERCENT/30X,32(1H*))
      END
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
      SUBROUTINE NCLASS (S, MCLASS, CLASS, W, MOC)
C
C NONPARAMETRIC CLASSIFICATION
C CLASSIFIES DATA SAMPLES AND STORES CLASSIFICATION RESULTS ON TAPE
C
      LOGICAL*1 S(NN,NSAMP), MCLASS(NSAMP)
      DIMENSION W(1), MOC(1), KS(20)
      DOUBLE PRECISION CLASS(MM)
      COMMON /CLSSFR/ NN, MM, NRECS, NSAMP, NSC
C
      NN1 = NN + 1
      MM1 = MM + 1
      DO 1009 NC=1,MM
 1009 KS(NC) = 0
      NTOT = NSAMP * NRECS
C
      DO 12 NREC=1,NRECS
      READ (10) S
      CALL NOPACA (S, MCLASS, KS, W, MOC, NSAMP, NN1, MM1)
   12 WRITE (11) MCLASS
C
      WRITE (6,2010)
      DO 1221 NC=1,MM
      PCT = 100.0 * KS(NC) / NTOT
 1221 WRITE (6,2011) NC, CLASS(NC), KS(NC), PCT
      WRITE (6,2012) NTOT
      RETURN
C
 2010 FORMAT ('1'/30X,29('*')/30X,'* RESULTS OF CLASSIFICATION *'/30X,
     .'*',7X,'DATA SAMPLES',8X,'*'/30X,29('*')///20X,'CLASS',15X,'SAMPLE
     .S',15X,'PERCENT'/20X,5('*'),2(15X,7('*')))
 2011 FORMAT (/I17,A9,I20,F22.2)
 2012 FORMAT (/13X,13HTOTAL SAMPLES,I20/13X,13(1H*))
      END
      SUBROUTINE NOPACA (X, NW, KS, W, MOC, NSS, NN1, MM1)
C
C NON-PARAMETRIC CLASSIFICATION OF A STRING OF NSS FEATURE VECTORS
C USING PRE-LEARNED LINEAR DISCRIMINANT FUNCTIONS
C
      LOGICAL*1 X(NN,NSS), NW(NSS)
      DIMENSION W(NN1,MM1), MOC(1), KS(1), X1(20)
      COMMON /CLSSFR/ NN, MM, NRECS, NSAMP, NSC
C
C     COMPUTE VALUES OF DISCRIMINANT FUNCTION AND TRANSFER IF POSITIVE
      DO 20 NS1=1,NSS
      DO 5 NF1=1,NN
    5 X1(NF1) = X(NF1,NS1)
C
      DO 1 NW1=1,MM1
      G = W(NN1,NW1)
        DO 2 NF1=1,NN
    2   G = G + W(NF1,NW1)*X1(NF1)
      IF (G.GT.0.0) GO TO 3
    1 CONTINUE
      NW1 = MM
C
    3 NC=MOC(NW1)
      KS(NC) = KS(NC)+1
      NW(NS1) = NC
   20 CONTINUE
      RETURN
      END
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```
      SUBROUTINE GASINV (A, N, DET)
C
C  COMPUTE INVERSE AND DETERMINANT OF SYMMETRIC MATRIX A
C
      DOUBLE PRECISION A(30,30), DET, TEST, TEMP, FAC, W, D
      DIMENSION IORD(20)
      DET = 1.0
      DO 1 I=1,N
    1 IORD(I) = I
      DO 2 K=1,N
      IF (K.EQ.N) GO TO 3
      TEST = DABS(A(K,K))
      KP1 = K + 1
      L = K
      DO 4 I=KP1,N
      IF (TEST.GE.DABS(A(I,K))) GO TO 4
      TEST = DABS(A(I,K))
      L = I
  4   CONTINUE
      IF (L.EQ.K) GO TO 3
      DO 5 J=1,N
      TEMP = A(L,J)
      A(L,J) = A(K,J)
    5 A(K,J) = TEMP
      J = IORD(L)
      IORD(L) = IORD(K)
      IORD(K) = J
      DET = -DET
    3 DET = DET*A(K,K)
      A(K,K) = 1.0 / A(K,K)
      DO 6 J=1,N
      IF (J.EQ.K) GO TO 6
      A(K,J) = A(K,J) * A(K,K)
  6   CONTINUE
      DO 7 I=1,N
      IF (I.EQ.K) GO TO 7
      FAC = A(I,K)
      A(I,K) = -A(I,K) * A(K,K)
      DO 8 J=1,N
      IF (J.EQ.K) GO TO 8
      W = FAC * A(K,J)
      D = A(I,J) - W
      IF (DABS(D).LT.0.00001*DABS(W)) D = 0.0
      A(I,J) = D
  8   CONTINUE
  7   CONTINUE
  2   CONTINUE
      NM1 = N-1
      DO 9 J=1,NM1
 12   CONTINUE
      IF (IORD(J).EQ.J) GO TO 9
      K = IORD(J)
      IORD(J) = IORD(K)
      IORD(K) = K
      DO 10 I=1,N
      TEMP = A(I,J)
      A(I,J) = A(I,K)
   10 A(I,K) = TEMP
      GO TO 12
 9    CONTINUE
      DO 15 I=2,N
      I1 = I - 1
      DO 15 J=1,I1
      A(I,J) = (A(I,J)+A(J,I))/2.0
   15 A(J,I) = A(I,J)
      RETURN
      END
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

MAXIMUM LIKELIHOOD CLASSIFIER

I. NAME

MAXLIK (training)
PCLASS (classification)

II. DESCRIPTION

The discriminant functions are the Gaussian probability distributions for each class which are defined by the mean values and covariance matrices of the training data. For an unknown data vector, it is possible to compute the probability of its belonging to each of the classes under consideration.

Additional a priori weighting factors (such as an estimate of the class populations) may be included. Assignment is made to the class for which the likelihood of belonging is a maximum.

III. CALLING SEQUENCE

CALL MAXLIK (X, CLASS, EM, EK, B, COVINV)
CALL PCLASS (S, MCLASS, CLASS, EM, B, COVINV)
where the arrays are dimensioned as follows:
X (NN, MM, NSC) - bytes of training data;
CLASS (MM) - double precision class names;
EM (NN, MM) - mean vectors of the classes;
EK (NN, NN) - double precision covariance matrix;
B (MM) - Gaussian function constant terms;
COVINV (MM X NN X (NN+1)/2) - triangular parts of the covariance matrices for the classes;
S (NN, NSAMP) - input buffer array; and,
MCLASS(NSAMP) - output buffer array.

The parameters defined in a common block named CLSSFR are the following:
NN - number of channels of data;
MM - number of classes;

NRECS - number of records;

NSAMP - number of pixels per record; and

NSC - number of training samples per class.

IV. INPUT/OUTPUT

1. INPUT

The input data set is sequential, with data samples arranged by measurement vectors (interleaved bands) and data length one byte.

2. OUTPUT

The output data set contains the class numbers in bytes corresponding to each input pixel.

V. DESCRIPTION OF SUBROUTINES

The subroutine linkages are given in the following table.

MAXIMUM LIKELIHOOD CLASSIFIER SUBROUTINES	
CALLING PROGRAM	PROGRAMS CALLED
MAXLIK	SUBLOP PTEST
SUBLOP	GASINV
PTEST	MALICA
PCLASS	MALICA

The subroutines are described in the following table.

DESCRIPTION OF SUBROUTINES

SUBROUTINE OR ENTRY	STORAGE (BYTES)	FUNCTION
MAXLIK	500	Calls routines to compute the Gaussian statistics and test on training data.

DESCRIPTION OF SUBROUTINES

SUBROUTINE OR ENTRY	STORAGE (BYTES)	FUNCTION
SUBLOP	2164	Compute the Gaussian distribution functions.
PTEST	1422	Test the classifier on training data.
PCLASS	1254	Classify the data set, compute and print class occupancies.
MALICA	1216	Classify data samples and return class numbers.
GASINV	1838	Double precision inversion of a symmetric matrix.

VI. PERFORMANCE SPECIFICATIONS

1. STORAGE

The program requires 90K bytes when classifying the Mobile Bay data set which has scan lines 1200 pixels long.

2. EXECUTION TIME

Approximately 1 second is required to compute the distribution functions. The classification speed for six classes is approximately 650 pixels/second.

VII. METHOD

It is assumed that the distribution of training data for a single class approximates the bell-shaped curve of the Gaussian or normal distribution.

The mathematical function for this curve is

$$P(x_j/c_i) = \frac{1}{\sqrt{2\pi}\,\sigma_j} \exp\left[-\frac{1}{2}\left(\frac{x_j - m_j}{\sigma_j}\right)^2\right]$$

where σ and m_j are the standard deviation and mean for measurement x_j belonging to class c_i.

Considering multichannel measurements, the joint probability function for a complete multivariate feature vector is

$$P(x_1,x_2,x_3,\ldots,x_n/c_i) = \frac{1}{\sqrt{(2\pi)^n D}} \exp\left[-\frac{1}{2}(X-M)^T K^{-1}(X-M)\right]$$

where (X-M) is the vector $\{x_1-m_1,\ x_2-m_2,\ \ldots,\ x_n-m_n\}$, K is the covariance matrix, and D is the determinant of K. The elements of the covariance matrix are a measure of the deviation of the corresponding x's from their mean values m:

$$K_{ij} = \frac{1}{N-1} \sum_{n=1}^{N} (x_{in} - m_i)(x_{jn} - m_j)$$

where N is the number of data samples used in the calculation.

The parameters--mean values and covariance matrices--completely define the Gaussian distribution functions. These parameters are easily determined for each class under consideration from the known set of training samples.

When the Gaussian parameters have been estimated, the Gaussian probability distribution for each class is completely defined. Thus, given any unknown feature vector, it is possible to compute the probability of this feature vector belonging to any one of the classes under consideration. Assignment is made to the class for which the probability is greatest; this is termed the maximum likelihood method of classification. For faster computation, the logarithm of the probability is computed and the decision function takes the form

$$G_i = \ln P_i - \frac{1}{2}\ln\left|K_i\right| - \frac{1}{2}(X-M_i)^T K_i^{-1}(X-M_i).$$

P_i is the probability of class i being present, M_i is the mean vector, and K_i is the covariance matrix. The decision point between two classes occurs when the probabilities are equal. This point is not midway between the means when the widths of the distributions are unequal.

The training and classification flow charts are given in Figures 21 and 22.

VIII. COMMENTS

If a priori probabilities are to be included, they are multiplied by the corresponding elements of array B after the call to MAXLIK.

IV. TESTS

The discriminant values and class assignments have been examined by printing the values.

X. LISTINGS

Listings of the routines follow.

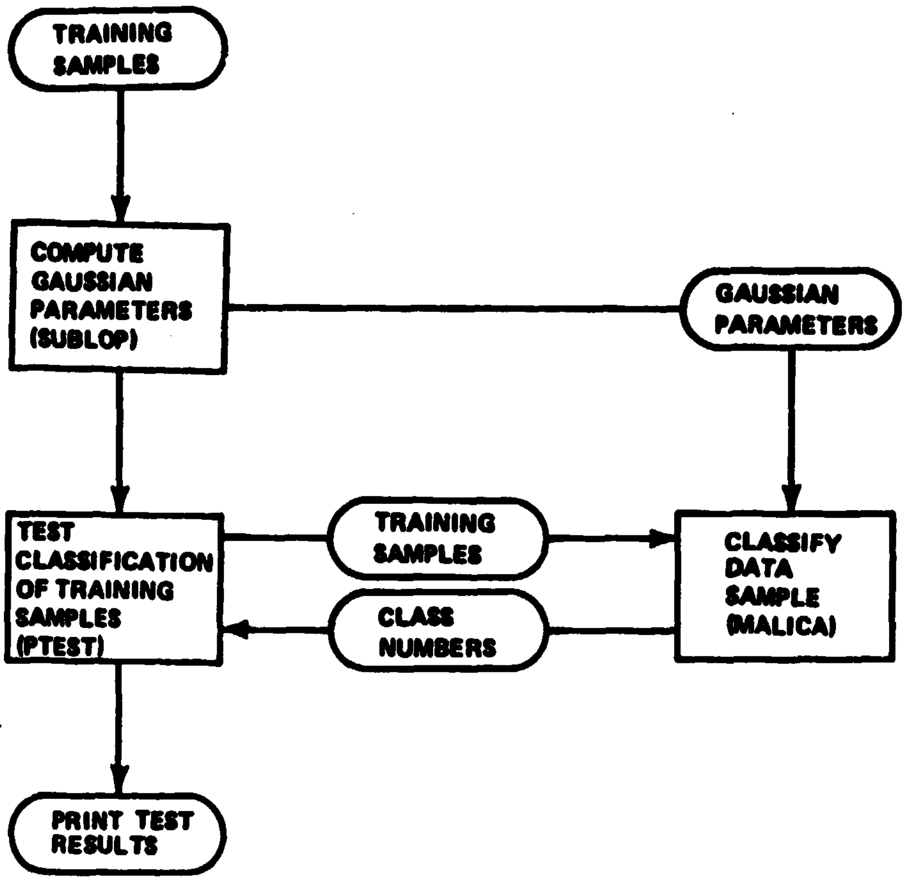

TRAINING SAMPLES
COMPUTE GAUSSIAN PARAMETERS (SUBLOP)
GAUSSIAN PARAMETERS
TEST CLASSIFICATION OF TRAINING SAMPLES (PTEST)
TRAINING SAMPLES
CLASSIFY DATA SAMPLE (MALICA)
CLASS NUMBERS
PRINT TEST RESULTS

Figure 21. Classifier Training Phase of Maximum Likelihood Classification

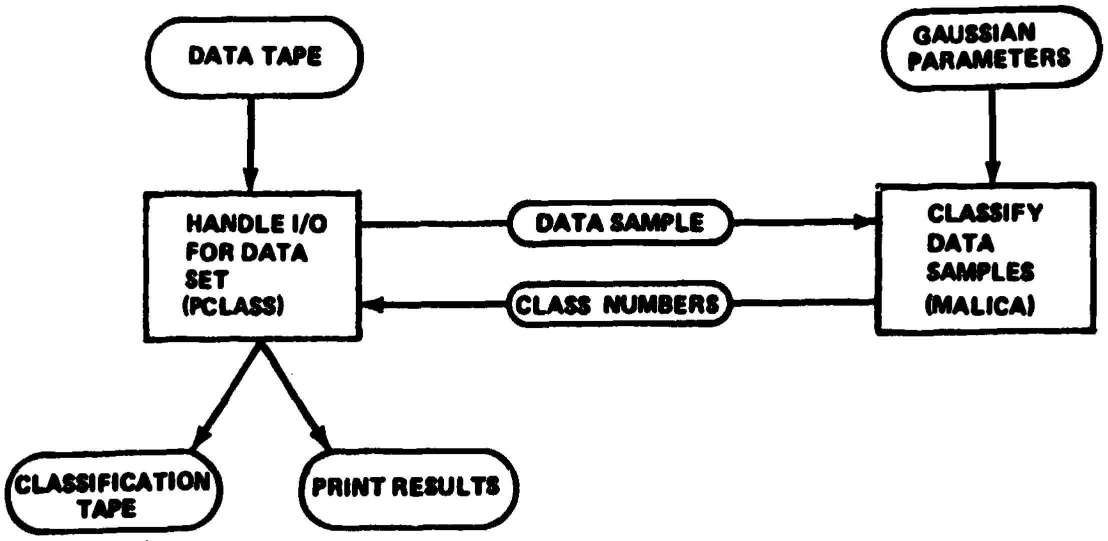


Figure 22. Classification Phase of Maximum Likelihood Classifier

```
      SUBROUTINE MAXLIK (X, CLASS, EM, EK, B, COVINV)
C
C  COMPUTE GAUSSIAN STATISTICS AND TEST USING TRAINING SAMPLES
C
      DIMENSION X(1), CLASS(1), EM(1), EK(1), B(1), COVINV(1)
      COMMON /CLSSFR/ NN, MM, NRECS, NSAMP, NSC
C
C  COMPUTE THE GAUSSIAN STATISTICS OF THE TRAINING SAMPLES
      CALL SUBLOP (X, CLASS, EM, EK, B, COVINV)
C
C  TEST THE CLASSIFICATION OF THE TRAINING SAMPLES
      CALL PTEST (X, CLASS, EM, B, COVINV)
      RETURN
      END
      SUBROUTINE SUBLOP (X, CLASS, EM, EK, B, COVINV)
C
C  SUPERVISED BATCH LEARNING OF PARAMETERS
C  COMPUTE MEAN VECTORS AND COVARIANCE MATRICES
C
      LOGICAL*1 X(NN,MM,LKK)
      DIMENSION EM(NN,MM), B(MM), COVINV(1)
      DOUBLE PRECISION CLASS(MM), EK(NN,NN), DET
      COMMON /CLSSFR/ NN, MM, NRECS, NSAMP, LKK
C
C     LOOP OVER CLASSES
      WRITE (6,4550)
      K = 0
      DO 9500 I3=1,MM
C
      DO 9000 I1=1,NN
      MEAN = 0
        DO 4000 LK=1,LKK
 4000   MEAN = MEAN + X(I1,I3,LK)
 9000 EM(I1,I3) = FLOAT(MEAN) / FLOAT(LKK)
C
      DO 9200 I1=1,NN
      DO 9200 I2=1,I1
      EK(I1,I2) = 0.0
        DO 9100 LK=1,LKK
 9100   EK(I1,I2) = EK(I1,I2) + (X(I1,I3,LK)-EM(I1,I3)) *
     *  (X(I2,I3,LK)-EM(I2,I3))
      EK(I1,I2) = EK(I1,I2) / (LKK-1)
      EK(I2,I1) = EK(I1,I2)
 9200 CONTINUE
C
      WRITE (6,4577) I3, CLASS(I3), LKK
      DO 9300 N1=1,NN
 9300 WRITE (6,4554) EM(N1,I3), (EK(N1,N2), N2=1,N1)
C
C  INVERT COVARIANCE MATRICES, COMPUTE GAUSSIAN FUNCTION CONSTANT TERMS
      CALL GASINV (EK, NN, DET)
      B(I3) = -0.5 * (NN*ALOG(2.0*3.14159265) + DLOG(DET))
      WRITE (6,4557) DET
C
C     PACK THE LOWER TRIANGULAR PART OF THE INVERSE COVARIANCE MATRIX
      DO 9400 N1=1,NN
      DO 9400 N2=1,N1
      K = K + 1
 9400 COVINV(K) = EK(N1,N2)
 9500 CONTINUE
      RETURN
C
 4550 FORMAT ('1'/20X,'ESTIMATED GAUSSIAN PARAMETERS'/20X,29('*')//5X,
     *'MEAN VECTORS',10X,'COVARIANCE MATRICES')
 4554 FORMAT (F15.2,(5X,16F7.2))
 4557 FORMAT (/20X,'DETERMINANT =',1PE10.3)
 4577 FORMAT (//I20,A10,I6,' SAMPLES')
      END
      SUBROUTINE PTEST (X, CLASS, EM, B, COVINV)
C
C  CLASSIFIES KNOWN DATA SAMPLES - PARAMETRIC CLASSIFICATION
C
      DIMENSION EM(1), B(1), COVINV(1), KS(20)
      LOGICAL*1 X(NN,MM,NSC), MCLASS
      DOUBLE PRECISION CLASS(MM)
      COMMON /CLSSFR/ NN, MM, NRECS, NSAMP, NSC
C
      WRITE (6,2008) CLASS
      TE = 0.0
C
      DO 1301 NC = 1,MM
      DO 1221 NW=1,MM
 1221 KS(NW) = 0
C
      DO 1400 NS1 = 1,NSC
 1400 CALL MALICA (X(1,NC,NS1), MCLASS, KS, EM, B, COVINV, 1)
C
      EFF = 100.0 * KS(NC) / NSC
      WRITE (6,2009) NC, CLASS(NC), NSC, KS(NC), EFF, (KS(N), N=1,MM)
 1301 TE = TE + EFF
C
      AVE = TE / FLOAT(MM)
      WRITE (6,2000) AVE
      RETURN
C
 2000 FORMAT (//30X,18HAVERAGE ACCURACY =,F6.1,8H PERCENT/30X,32(1H*))
 2008 FORMAT ('1'/30X,29('*')/30X,'* RESULTS OF CLASSIFICATION *'/30X,'*
     .      TRAINING SAMPLES       *'/30X,29('*')///14X,'NUMBER OF  NUMBER
     .      PERCENT     NUMBER OF SAMPLES CLASSIFIED AS'/6X,'CLASS    SAM
     .PLES   CORRECT    CORRECT',10A9/(43X,10A9))
 2009 FORMAT (/I4,A9,I7,I10,F12.1,10I9,(/42X,10I9))
      END
      SUBROUTINE PCLASS (S, MCLASS, CLASS, EM, B, COVINV)
C
C  CLASSIFIES UNKNOWN DATA SAMPLES - PARAMETRIC CLASSIFICATION
C
      LOGICAL*1 S(NN,NSAMP), MCLASS(NSAMP)
      DIMENSION EM(1), B(1), COVINV(1), KS(20)
      DOUBLE PRECISION CLASS(MM)
      COMMON /CLSSFR/ NN, MM, NRECS, NSAMP, NSC
C
      DO 10 NC=1,MM
   10 KS(NC) = 0
C
      DO 12 NREC=1,NRECS
      READ (10) S
      CALL MALICA (S, MCLASS, KS, EM, B, COVINV, NSAMP)
   12 WRITE (11) MCLASS
C
      NTOT = NSAMP * NRECS
      WRITE (6,2010)
      DO 20 NC=1,MM
      PCT = 100.0 * KS(NC) / NTOT
   20 WRITE (6,2011) NC, CLASS(NC), KS(NC), PCT
      WRITE (6,2012) NTOT
      RETURN
C
 2010 FORMAT ('1'/30X,29('*')/30X,'* RESULTS OF CLASSIFICATION *'/30X,
     ,'*',7X,'DATA SAMPLES',8X,'*'/30X,29('*')///20X,'CLASS',15X,'SAMPLE
     ,S',15X,'PERCENT'/20X,5('*'),2(15X,7('*')))
 2011 FORMAT (/I18,A9,I20,F21.2)
 2012 FORMAT (/14X,'TOTAL SAMPLES',I20/14X,13('*'))
      END
      SUBROUTINE MALICA (X1, KMAX, KS, EM, B, COVINV, NSS)
C
C            MAXIMUM LIKELIHOOD CLASSIFICATION
C
      DIMENSION KS(1), EM(NN,MM), B(MM), COVINV(1), DX(20), AX(20)
      LOGICAL*1 X1(NN,NSS), KMAX(NSS)
      COMMON /CLSSFR/ NN, MM, NRECS, NSAMP, NSC
C
      DO 2000 NS1=1,NSS
      DO 1000 NF1=1,NN
 1000 AX(NF1) = X1(NF1,NS1)
C
C     FIND MAXIMUM PROBABILITY OVER CLASSES
      GMAX = -1.0 E 50
      K = 0
      DO 1900 I=1,MM
      G = B(I)
C
C       COMPUTE GAUSSIAN EXPONENT (-1/2) (X-M) (KINV) (X-M)
        DO 6300 II=1,NN
        DX(II) = AX(II) - EM(II,I)
C
C           COMPUTE TERMS FROM LOWER TRIANGULAR MATRIX
            SUM = 0.0
            JJ = 0
 6200       CONTINUE
            JJ = JJ + 1
            IF (JJ.EQ.II) GO TO 6250
            K = K + 1
            SUM = SUM + DX(JJ)*COVINV(K)
            GO TO 6200
C
C         COMPUTE 1/2 DIAGONAL TERM OF (X-M) (KINV)
 6250     K = K + 1
          SUM = SUM + 0.5*DX(JJ)*COVINV(K)
C
 6300   G = G - SUM*DX(II)
C
      IF (G.LT.GMAX) GO TO 1900
      MAX = I
      GMAX = G
 1900 CONTINUE
      KS(MAX) = KS(MAX) + 1
 2000 KMAX(NS1) = MAX
      RETURN
      END
```

CHAPTER VI

PREPARATION OF GROUND TRUTH MAPS IN DIGITAL FORMAT

THINNING OF BOUNDARY IMAGES

I. <u>NAME</u>

PEELS

II. <u>DESCRIPTION</u>

This subroutine starts with the output (boundary lines) of a microdensitometer, applies a given threshold of density, and reduces the thickness of the boundary lines by "peeling" their outer layers while preserving the distinctness of regions separated by them.

III. <u>CALLING SEQUENCE</u>

CALL PEELS (NTAPI, NTAPO, NREC, NEL, IT, MPASS, ITYPE, LX, LY, IBDY)
where
NTAPI, NTAPO are the logical unit numbers of the input and output sequential data sets;
NREC, NEL are the number of records and the number of pixels (bytes) per record in the input image;
IT is a threshold on density; if IT is positive (negative) all points with densities $\geq$ IT ($\leq$ IT) will be regarded as boundary points;
MPASS is the maximum number of iterations permitted;
ITYPE determines the type of boundary connections (1 for diagonal, 2 for perpendicular; LX, LY, IBDY are scratch arrays with LX, LY dimensioned as indicated in the listing and IBDY dimensioned NEL.

IV. <u>INPUT/OUTPUT</u>

1. INPUT

The input image should be on a sequential data set with unit number NTAPI and consist of NREC records and NEL bytes per record, each record corresponding to a line of the digitized image and each byte, to a pixel. All other inputs are as indicated in the calling sequence.

2. OUTPUT

The output of this program will be on unit NTAPO as a sequential data set with NREC records. The records will be in SLIC (Scan Line Intersection Code) format. That is, the first word of the I'th record indicates the number of words that follow, and each subsequent word is a column coordinate of the intersection of the I'th scan line with the boundary image.

3. FILE STORAGE

This program requires two sequential scratch data sets to handle the intermediate iterations of the boundary data.

V. <u>SUBROUTINES CALLED</u>

The subroutines called by PEELS are given in the following table.

EXTERNAL LINKAGES

CALLING PROGRAMS	PROGRAMS CALLED
PEELS	SARN* SVSMLI@ VLTHR CMPRES SAWN* PEELER EXPBDY
CMPRES	ISTORE+
PEELER	SVSCI PEELRI PEELRO SAWN*
EXPBDY	ILOAD+
PEELRI	SARN BLSFTV BRSFTV°

TABLE EXTERNAL LINKAGES

CALLING PROGRAMS	PROGRAMS CALLED
PEELRO	ICOMP1+ IAND+ BLSFTV BRSFTV°
BLSFTV	ILOAD+ ISTORE+
BRSFTV	ILOAD+ ISTORE+

* Entry under DARN
@ Entry under SVSCI
+ Entry under LOGFUNC
o Entry under BLSFTV

A brief description of the function of each subroutine and its storage requirements are given in the following table.

DESCRIPTION OF SUBROUTINES

SUBROUTINE OR ENTRY	STORAGE (BYTES)	FUNCTION
PEELS	1526	Reduces the thickness of boundary lines.
SARN	Entry under DARN (882)	Reads N bytes from a sequential access unit into an array.
SVSML1	Entry under SVSCI (842)	Sets the Mth elt. of a LOGICAL *1 vector to zero.
VLTHR	400	Thresholds a vector of 8-bit integers to get a T-F vector.
CMPRES	412	Packs the information of an array into the first NEL bits of another array.
SAWN	Entry under DARN	Writes N bytes of an array onto a sequential access device.
PEELER	1856	Initiates the removal of one layer of thick boundaries from top, left, bottom, and right of an image.

DESCRIPTION OF SUBROUTINES

SUBROUTINE OR ENTRY	STORAGE (BYTES)	FUNCTION
EXPBDY	570	Senses each bit in a record and converts the record to scan line intersection format.
ISTORE	Entry under LOGFUNC (304)	Moves a right adjusted field of data from one word to a specified field of another word.
SVSCI	842	Sets all elements of a vector to a given number.
PEELRI	576	Reads one record of the input image and sets up two arrays, one with the bits of the record shifted one bit to the left, and the other with the bits shifted one bit to the right.
PEELRO	2720	Performs the peeling of one record.
ILOAD	Entry under LOGFUNC	Moves a field of data from a source word and right justifies it as the output argument.
BLSFTV	880	Generates array IY with the bits in IX shifted one bit to the left.
BRSFTV	Entry under BLSFTV	Generates array IY with the bits in IX shifted one bit to the right.
IOR	Entry under LOGFUNC	Performs an inclusive logical OR operation.
ICOMPI	Entry under LOGFUNC	Outputs a value which is the input value source with a specified field of bits 1's complemented.
IAND	Entry under LOGFUNC	Performs a logical AND operation.

VI. PERFORMANCE SPECIFICATIONS

1. STORAGE

Including a driver (whose size depends largely on the dimensions of LX, LY, and IBDY which are functions of NEL), the program needs approximately 50K for handling NEL = 1000.

2. EXECUTION TIME

The execution time is highly dependent on the size and complexity of the boundary image, the thickness of the boundary lines and the maximum number of passes (MPASS) requested. In the case of the LACIE GTM (a 820 x 1000 map with boundaries 2 and 3 pixels thick) it took about 4 minutes to thin the boundaries.

3. RESTRICTIONS

None.

VII. METHOD

A simplified flow diagram for thinning boundaries is shown in Figure 23.

The program has three major steps:

1. Thresholding, compressing, and writing on a sequential data set.
2. Iterating to "peel" boundaries.
3. Changing to SLIC format and writing on output sequential data set.

1. THRESHOLDING AND COMPRESSING

The routine SARN reads each record (of NEL bytes) of the input data set into the array LX. The routine VLTHR thresholds each of the NEL bytes in LX. A logical vector LY is defined as follows:

```
IF (IT.GE.0) LX(I).GE.IT→LY(I)=T
IF (IT.LT.0) LX(I).LE.IABS(IT)→LY(I)=T
```

for I = 1, NEL.

The routine CMPRES is then used to pack the information in LY into the first NEL bits of the array LX. The I'th bit of LX is "set" if and only if LY (I) is .TRUE..

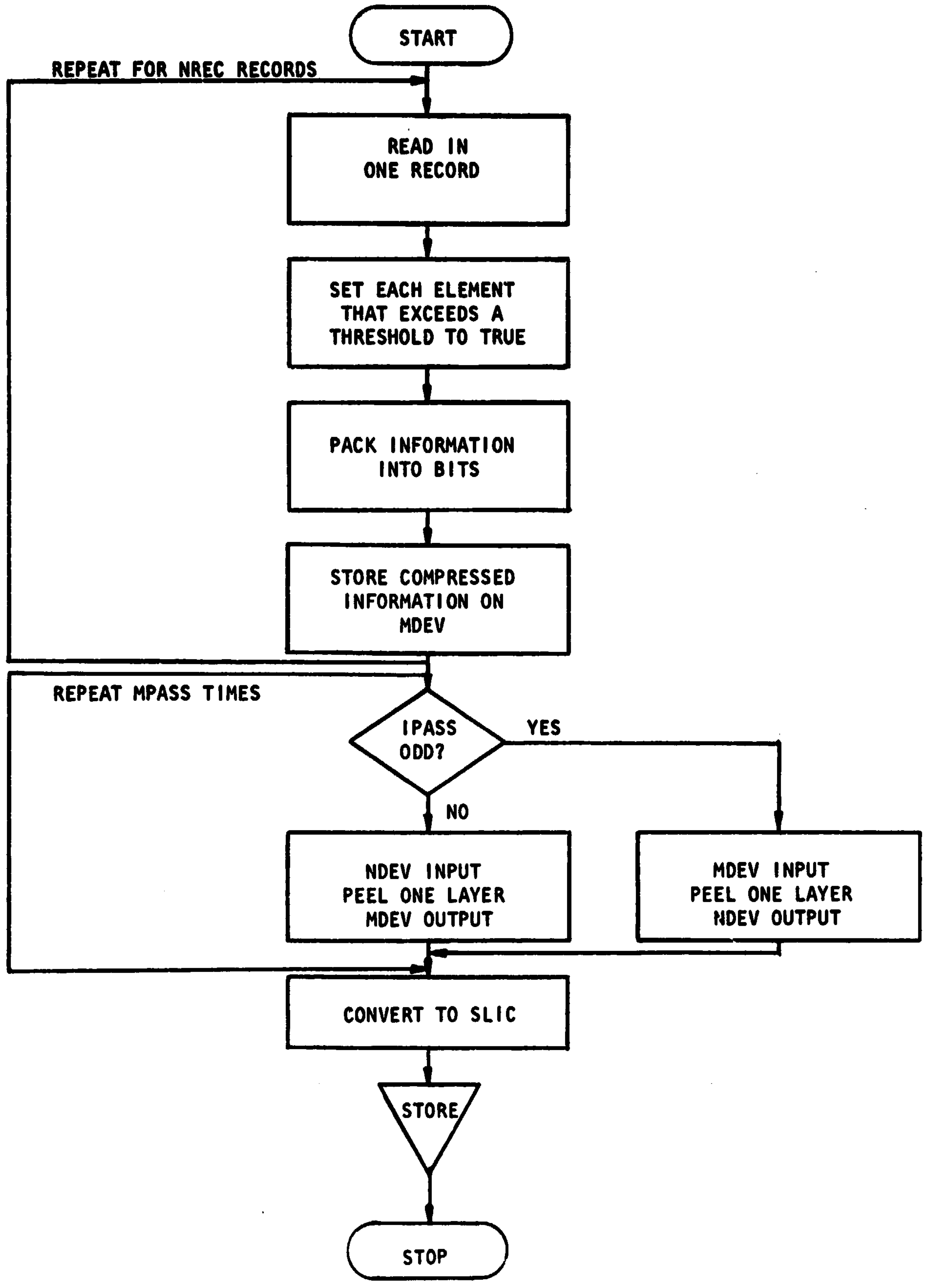

START
REPEAT FOR NREC RECORDS
READ IN ONE RECORD
SET EACH ELEMENT THAT EXCEEDS A THRESHOLD TO TRUE
PACK INFORMATION INTO BITS
STORE COMPRESSED INFORMATION ON MDEV
REPEAT MPASS TIMES
IPASS ODD?
YES
NO
NDEV INPUT PEEL ONE LAYER MDEV OUTPUT
MDEV INPUT PEEL ONE LAYER NDEV OUTPUT
CONVERT TO SLIC
STORE
STOP

FIGURE 23. SIMPLIFIED FLOW DIAGRAM FOR THINNING BOUNDARIES.

The compressed boundary information is then written on the sequential access data set MDEV using the routine SAWN.

2. ITERATING TO PEEL

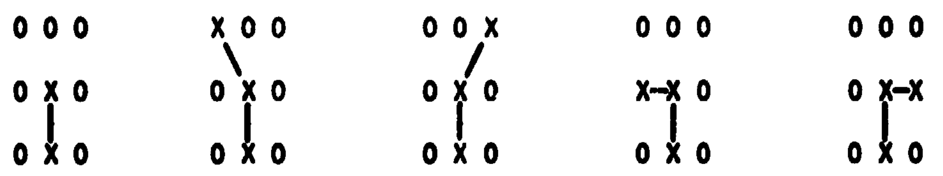
The main peeling routine is called PEELER. The input to this routine is from MDEV whenever IPASS, the iteration number, is odd and the output then will be written on NDEV. When IPASS is even, the input and output designations are interchanged. One call to PEELER removes one "layer" of the thick boundaries from top, left, bottom, and right.

To decide whether a particular boundary point should be deleted (i.e., the bit corresponding to it changed to 0), a 3 x 3 neighborhood centered around the point is examined. Consider the array

$$\begin{array}{ccc} a & b & c \\ d & e & f \\ g & h & i \end{array}$$

where each letter represents a binary pixel. It is to be decided whether e which is presently equal to 1 should be changed to 0. The conditions for a 'top peel' will be derived below and those for peeling from the other directions followed by symmetry.

First of all, e should be a top boundary point. That is, there should be no boundary point directly above e and there should be a boundary point below e. Therefore, $b = 0$ and $h = 1$ are necessary conditions. Suppose $\overline{b}\,h = 1$. (Here, $\overline{b}$ denotes the complement of b.) Then, it is only necessary to check whether e is essential to a boundary line through h and e. The line may proceed diagonally or at right angles from e. The conditions for various configurations are given below.



$\bar{b}h = 1$ $a\bar{d} = 1$ $c\bar{f} = 1$ $d\bar{g} = 1$ $f\bar{i} = 1$

DIAGONAL PERPENDICULAR

Thus, e is essential if and only if $\bar{b}h = 1$ or ($a\bar{d} = 1$ or $c\bar{f} = 1$) or ($d\bar{g} = 1$ or $f\bar{i} = 1$). Therefore, the condition for a top peel is that

$$\bar{b}h(\bar{a}+d)(\bar{c}+f) = 1 \qquad \text{(diagonal connection)}$$

Equivalently, to perform a top peel set

$$e = e(b+\bar{h}+a\bar{d}+c\bar{f})$$

It is convenient to implement the above equation by employing bit manipulation routines operating on pairs of 32-bit words, thereby performing the top-peel operation in parallel on 32 pixels. This is done by using the "current" array in place of e, the "previous" array for b, the "next" array in place of h. Also, the previous, current, and next arrays are right (left) shifted by one bit and used for a, d, and g (c, f and i) respectively in the peeling formulas.

The routine PEELER minimizes the movement of data in core by using circular buffers for storing the "previous, current, and next" arrays. An array J dimensioned 3 is used to store the indices pointing to these arrays ($J(1) \longrightarrow$ previous, $J(2) \longrightarrow$ current, $J(3) \longrightarrow$ next) and after finishing each record, only the array J is updated.

Also, top, left, bottom, and right peels are performed one after the other by just one pass through the data (thus minimizing I/O) by storing the intermediate results in core and operating with a phase lag.

When the I'th record LX is read from the input data set (see PEELR1), BLSFTV and BRSFTV are used to generate arrays LXL and LXR with the bits in LX shifted by one bit to the left and right, respectively.

Next, the (I-1)th record is peeled from the top. The top-peeled output of the (I-2)nd record is peeled from the left. The top- and left-peeled output of the (I-3)rd record is peeled from the bottom. The top-, left-, and bottom-peeled output of the (I-4)th record is right-peeled and written on the output data set. Also, whenever any peeling is done other than from the right, the output is shifted to the left and right by one bit and the results are stored in the appropriate core locations pointed by J(3).

The routine PEELRO with the appropriate ISIDE will perform the peeling of one record. The above operations performed for I=1, NREC+4 will complete one iteration of peeling, constituting one call to PEELER. The number, NP, or words of input that were changed is counted during each call to PEELER. If NP=0 or the number of calls to PEELER has been MPASS, the iterations are stopped.

3. CONVERTING TO SLIC

Each record is read from the last scratch unit on which the output image was created. The routine EXPBDY is used to sense each bit in the record. The bit number of each 1-bit is stored in IBDY. The total number, N, of 1-bits followed by N words of the array IBDY are written on unit NTAPO.

VIII. COMMENTS

None.

IX. TESTS

The program was tested on a small portion of a boundary image and was found to work satisfactorily. Figures 24 (a) and (b) show computer line printer plots of the image before and after peeling, respectively.

X. LISTINGS

The listings of PEELS and the associated routines are attached at the end of this section.

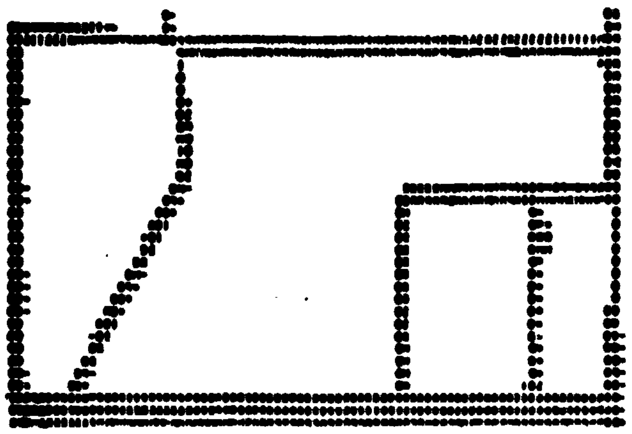
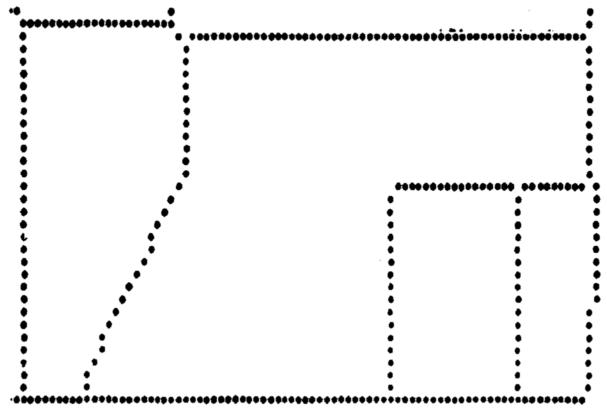
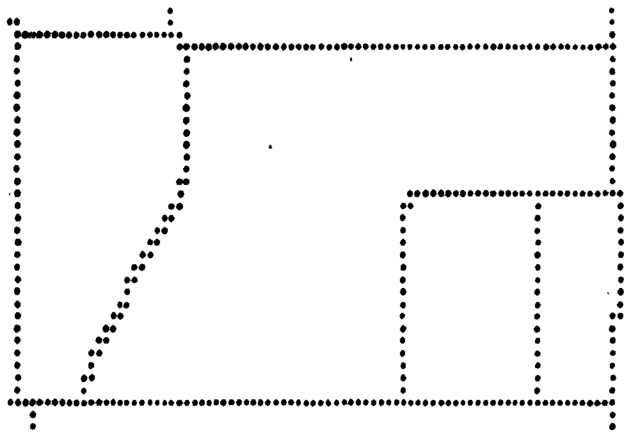


Figure 24(a). Boundaries before Thinning



Diagonal Connections



Perpendicular Connections

Figure 24(b) Thinned Boundaries

```
C
      SUBROUTINE PEELS (NTI,NTO,NREC,NEL,IT,MPASS,ITYPE,LX,LY,IBDY)
C
C  THIS SUBROUTINE PEELS THE OUTER LAYERS FROM THE BOUNDARY LINES OF AN
C  IMAGE WHILE PRESERVING THE DISTINCTNESS OF REGIONS SEPARATED BY
C  THE BOUNDARY LINES
C
C     DIMENSION LX(36*((NEL-1)/32+1)),LY((NEL-1)/4+1),IBDY(NEL)
      DIMENSION LX(1),LY(1),IBDY(1)
      DATA MDEV, NDEV /1, 2/
C
      N=(NEL-1)/32+1
      REWIND MDEV
      DO 10 I=1,NREC
      CALL SARN(NTI,LX,NEL)
      IF(I.EQ.1.OR.I.EQ.NREC) CALL SVSCL1(LX,NEL,0)
      LX(1) = 0
      LX(NEL) = 0
      CALL VLTHR(LX,NEL,IT,LY)
      LX(N)=0
      CALL CMPRES(LY,NEL,LX)
      CALL SAWN(MDEV,LX,N*4)
10    CONTINUE
C
      DO 20 IPASS=1,MPASS
      REWIND MDEV
      REWIND NDEV
      IF (MOD(IPASS,2).EQ.1)
     .CALL PEELER(MDEV,NDEV,NREC,ITYPE,N,LX,LX(12*N+1),LX(24*N+1),LY,NP)
      IF (MOD(IPASS,2).EQ.0)
     .CALL PEELER(NDEV,MDEV,NREC,ITYPE,N,LX,LX(12*N+1),LX(24*N+1),LY,NP)
      WRITE(6,100) IPASS,NP
      IF(NP.EQ.0)GO TO 30
20    CONTINUE
      IPASS=MPASS
C
30    IF(MOD(IPASS,2).EQ.1)JDEV=NDEV
      IF(MOD(IPASS,2).EQ.0)JDEV=MDEV
      REWIND JDEV
      DO 40 I=1,NREC
      CALL SARN(JDEV,LX,N*4)
      CALL EXPBDY (LX,N,I,NREC,NEL,IBDY,J)
      WRITE (NTO) J, (IBDY(L), L=1,J)
40    CONTINUE
      RETURN
C
100   FORMAT(5X'DURING PASS NUMBER'I3,' THROUGH PEELER'I6,' WORDS OF COM
     -PRESSED BOUNDARY INFORMATION WERE CHANGED.')
      END
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```
      SUBROUTINE DARN(IDEV,IREC,X,N)
C
C     THIS SUBROUTINE READS N BYTES FROM DIRECT ACCESS DEVICE IDEV
C     STARTING AT RECORD IX INTO ARRAY X
C
      LOGICAL*1 X(N)
      READ(IDEV'IREC)X
      RETURN
C
      ENTRY DAWN(IDEV,IREC,X,N)
C
C  THIS ENTRY WRITES N BYTES OF ARRAY X ONTO RECORD IX OF DIRECT ACCESS
C  DEVICE IDEV
      WRITE(IDEV'IREC)X
      RETURN
C
      ENTRY SARN(NTAPI,X,N)
C
C  THIS ENTRY READS N BYTES FROM SEQUENTIAL ACCESS UNIT NTAPI INTO
C  ARRAY X
      READ(NTAPI)X
      RETURN
C
      ENTRY SAWN(NTAPO,X,N)
C
C  THIS ENTRY WRITES N BYTES OF ARRAY X ONTO SEQUENTIAL ACCESS
C  UNIT NTAPI
      WRITE(NTAPO)X
      RETURN
      END
      SUBROUTINE VLTHR(LX,N,IT,LY)
C
      LOGICAL*1 LX(N),LY(N),F/.FALSE./,T/.TRUE./
C
C     THRESHOLD A VECTOR LX OF 8 BIT INTEGERS TO GET A T=F VECTOR.
C     IF 'IT' IS POSITIVE, LY(I) IS TRUE FOR LX(I) .GE. IT.
C     IF 'IT' IS NEGATIVE, LY(I) IS TRUE FOR LX(I) .LE. IABS(IT).
C
      ITT=IABS(IT)
      IF(IT.LT.0)GO TO 10
      DO 20 I=1,N
      LY(I)=F
      IF(LX(I).GE.ITT)LY(I)=T
20    CONTINUE
      RETURN
C
10    DO 30 I=1,N
      LY(I)=F
      IF(LX(I).LE.ITT)LY(I)=T
30    CONTINUE
      RETURN
      END
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
      SUBROUTINE CMPRES(LX,NEL,LY)
C
C  THIS SUBROUTINE IS USED TO PACK THE INFORMATION IN LX INTO THE FIRST
C  NEL BITS OF THE ARRAY LY
C
      LOGICAL*1 LX(NEL)
      DIMENSION LY(1)
C
      JWRD=1
      JBIT=33
      DO 10 I=1,NEL
      JBIT=JBIT-1
      IF(JBIT.NE.0)GO TO 20
      JBIT=32
      JWRD=JWRD+1
20    IX=LX(I)
      LY(JWRD)=ISTORE(IX,LY(JWRD),JBIT,1)
10    CONTINUE
      RETURN
      END
      SUBROUTINE PEELER (MDEV,NDEV,NREC,ITYPE,N,LX,LXR,LXL,LY,NP)
C
C  THIS SUBROUTINE INITIATES THE REMOVAL OF ONE LAYER OF THICK BOUNDARIE
C  FROM TOP, LEFT, BOTTOM AND RIGHT OF AN IMAGE
C
      DIMENSION LX(N,3,4),LXR(N,3,4),LXL(N,3,4),LY(N),J(3)
C
      NREC1=NREC+1
      NREC2=NREC+2
      NREC3=NREC+3
      NREC4=NREC+4
      J(1)=1
      J(2)=2
      J(3)=3
      CALL SVSCI(LX ,12*N,0)
      CALL SVSCI(LXR,12*N,0)
      CALL SVSCI(LXL,12*N,0)
      NP=0
C
      DO 10 I=1,NREC4
      DO 20 K=1,4
      IF(I.LE.NREC+K)GO TO 20
      CALL SVSCI(LX (1,J(3),K),N,0)
      CALL SVSCI(LXR(1,J(3),K),N,0)
      CALL SVSCI(LXL(1,J(3),K),N,0)
20    CONTINUE
      IF (I.LE.NREC) CALL PEELR1 (MDEV,LX,LXR,LXL,J,N)
      IF(I.GT.1.AND.I.LE.NREC1)
     .   CALL PEELRO(LX(1,1,1),LXR(1,1,1),LXL(1,1,1),J,N,1,
     .      LX(1,J(3),2),LXR(1,J(3),2),LXL(1,J(3),2),NP,ITYPE)
      IF(I.GT.2.AND.I.LE.NREC2)
     .   CALL PEELRO(LX(1,1,2),LXR(1,1,2),LXL(1,1,2),J,N,2,
     .      LX(1,J(3),3),LXR(1,J(3),3),LXL(1,J(3),3),NP,ITYPE)
      IF(I.GT.3.AND.I.LE.NREC3)
     .   CALL PEELRO(LX(1,1,3),LXR(1,1,3),LXL(1,1,3),J,N,3,
     .      LX(1,J(3),4),LXR(1,J(3),4),LXL(1,J(3),4),NP,ITYPE)
      IF(I.GT.4)
     .   CALL PEELRO(LX(1,1,4),LXR(1,1,4),LXL(1,1,4),J,N,4,
     .      LY,0,0,NP,ITYPE)
      IF(I.GT.4) CALL SAWN(NDEV,LY,4*N)
      DO 30 K=1,3
      J(K)=MOD(J(K),3)+1
30    CONTINUE
10    CONTINUE
      RETURN
      END
      SUBROUTINE FXPBDY(LX,N,IREC,NREC,NEL,IBDY,J)
C
C  THIS SUBROUTINE SENSES EACH BIT IN A RECORD AND CONVERTS THE RECORD
C  TO SCAN LINE INTERSECTION CODE FORMAT
C
      DIMENSION LX(N),IBDY(1)
      LOGICAL ILOAD
C
      IF(IREC.EQ.1.OR.IREC.EQ.NREC) GO TO 10
      GO TO 30
   10 J=NEL
      DO 20 I=1,NEL
      IBDY(I)=I
   20 CONTINUE
      GO TO 60
   30 JWRD=1
      JBIT=33
      J=0
      DO 50 I=1,NEL
      JBIT=JBIT-1
      IF(I.EQ.1.OR.I.EQ.NEL) GO TO 70
      IF(JBIT.NE.0) GO TO 40
      JBIT=32
      JWRD=JWRD+1
   40 IF(.NOT.ILOAD(LX(JWRD),JBIT,1)) GO TO 50
   70 J=J+1
      IBDY(J)=I
   50 CONTINUE
   60 RETURN
      END
      SUBROUTINE SVSCI(IX,N,IS)
C
C  THIS SUBROUTINE SETS ALL ELEMENTS OF AN INTEGER*4 VECTOR TO A GIVEN
C  NUMBER
C
      DIMENSION IX(N)
      DO 10 I=1,N
      IX(I)=IS
10    CONTINUE
      RETURN
C
      ENTRY SVSCI2(I2,N,IS)
C
C  THIS ENTRY SETS ALL ELEMENTS OF AN INTEGER*2 VECTOR TO A GIVEN NUMBER
      INTEGER*2 I2(N)
      DO 20 I=1,N
20    I2(I)=IS
      RETURN
C
      ENTRY SVSCL1(I3,N,L)
C
C  THIS ENTRY SETS ALL ELEMENTS OF A LOGICAL*1 VECTOR TO A GIVEN NUMBER
      LOGICAL*1 I3(N)
      DO 30 I=1,N
      I3(I)=L
30    CONTINUE
      RETURN
      END
      SUBROUTINE PEELR1 (MDEV, LX, LXR, LXL, J, N)
C
C  THIS SUBROUTINE READS ONE RECORD OF THE INPUT IMAGE AND SETS UP TWO
C  ARRAYS, ONE WITH THE BITS OF THE RECORD SHIFTED ONE BIT TO THE LEFT,
C  AND THE OTHER WITH THE BITS SHIFTED ONE BIT TO THE RIGHT
C
      DIMENSION LX(N,3), LXR(N,3), LXL(N,3), J(3)
C
      CALL SARN (MDEV, LX(1,J(3)), 4*N)
C
      CALL BLSFTV (LX(1,J(3)), N, LXL(1,J(3)))
      CALL BRSFTV (LX(1,J(3)), N, LXR(1,J(3)))
      RETURN
      END
      SUBROUTINE PEELRO (LX,LXR,LXL,J,N,ISIDE,LY,LYR,LYL,NP,ITYPE)
C
C  THIS SUBROUTINE PERFORMS THE PEELING OF ONE RECORD
C
      DIMENSION LX(N,3),LXR(N,3),LXL(N,3),LY(N),LYR(N),LYL(N),J(3)
C
      DO 60 I=1,N
      LY(I)=LX(I,J(2))
      IF(LY(I).EQ.0)GO TO 60
      GO TO (1, 2), ITYPE
C
C  ** TYPE 1 ALGORITHM - DIAGONAL CONNECTIONS
1     GO TO (10,20,30,40),ISIDE
C
C  TOP PEEL
10    IW1 = IOR  (LX (I,J(1)), ICOMP1(LX (I,J(3)),32,32))
      IW2 = IAND (LXR(I,J(1)), ICOMP1(LXR(I,J(2)),32,32))
      IW3 = IAND (LXL(I,J(1)), ICOMP1(LXL(I,J(2)),32,32))
      GO TO 50
C
C  LEFT PEEL
20    IW1 = IOR  (LXR(I,J(2)), ICOMP1(LXL(I,J(2)),32,32))
      IW2 = IAND (LXR(I,J(1)), ICOMP1(LX (I,J(1)),32,32))
      IW3 = IAND (LXR(I,J(3)), ICOMP1(LX (I,J(3)),32,32))
      GO TO 50
C
C  BOTTOM PEEL
30    IW1 = IOR  (LX (I,J(3)), ICOMP1(LX (I,J(1)),32,32))
      IW2 = IAND (LXR(I,J(3)), ICOMP1(LXR(I,J(2)),32,32))
      IW3 = IAND (LXL(I,J(3)), ICOMP1(LXL(I,J(2)),32,32))
      GO TO 50
C
C  RIGHT PEEL
40    IW1 = IOR  (LXL(I,J(2)), ICOMP1(LXR(I,J(2)),32,32))
      IW2 = IAND (LXL(I,J(1)), ICOMP1(LX (I,J(1)),32,32))
      IW3 = IAND (LXL(I,J(3)), ICOMP1(LX (I,J(3)),32,32))
      GO TO 50
C
C  ** TYPE 2 ALGORITHM - PERPENDICULAR CONNECTIONS
2     GO TO (11,22,33,44),ISIDE
C
C  TOP PEEL
11    IW1 = IOR  (LX (I,J(1)), ICOMP1(LX (I,J(3)),32,32))
      IW2 = IAND (LXR(I,J(2)), ICOMP1(LXR(I,J(3)),32,32))
      IW3 = IAND (LXL(I,J(2)), ICOMP1(LXL(I,J(3)),32,32))
      GO TO 50
C
C  LEFT PEEL
22    IW1 = IOR  (LXR(I,J(2)), ICOMP1(LXL(I,J(2)),32,32))
      IW2 = IAND (LX (I,J(3)), ICOMP1(LXL(I,J(3)),32,32))
      IW3 = IAND (LX (I,J(1)), ICOMP1(LXL(I,J(1)),32,32))
      GO TO 50
C
C  BOTTOM PEEL
33    IW1 = IOR  (LX (I,J(3)), ICOMP1(LX (I,J(1)),32,32))
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```
      IW2 = IAND (LXL(I,J(2)), ICOMP1(LXL(I,J(1)),32,32))
      IW3 = IAND (LXR(I,J(2)), ICOMP1(LXR(I,J(1)),32,32))
      GO TO 50
C
C  RIGHT PEEL
44    IW1 = IOR  (LXL(I,J(2)), ICOMP1(LXR(I,J(2)),32,32))
      IW2 = IAND (LX (I,J(1)), ICOMP1(LXR(I,J(1)),32,32))
      IW3 = IAND (LX (I,J(3)), ICOMP1(LXR(I,J(3)),32,32))
C
50    IW1 = IOR (IW1, IW2)
      IW1 = IOR (IW1, IW3)
      LY(I) = IAND (LY(I), IW1)
      IF (LX(I,J(2)).NE.LY(I)) NP=NP+1
60    CONTINUE
C
      IF(ISIDF.EQ.4)RETURN
      CALL BLSFTV(LY,N,LYL)
      CALL BRSFTV(LY,N,LYR)
      RETURN
      END
      SUBROUTINE BLSFTV(IX,N,IY)
C
C  THIS SUBROUTINE GENERATES ARRAY IY WITH THE BITS IN IX SHIFTED ONE
C  BIT TO THE LEFT
C
      DIMENSION IX(N),IY(N)
      N1=N-1
      DO 10 I=1,N1
C
C     MOVE ONE BIT FROM BIT POSITION 32 OF IX(I+1) INTO IY(I) AND
C     RIGHT JUSTIFY
      IY(I)=ILOAD(IX(I+1),32,1)
C
C     MOVE RIGHTMOST 31 BITS FROM IX(I) INTO IY(I) STARTING AT BIT
C     POSITION 32
      IY(I)=ISTORE(IX(I),IY(I),32,31)
10    CONTINUE
      IY(N)=0
      IY(N)=ISTORE(IX(N),IY(N),32,31)
      RETURN
C
C ********************************************************************
C ********************************************************************
C
      ENTRY BRSFTV(IX,N,IY)
C
C  THIS ENTRY GENERATES ARRAY IY WITH THE BITS IN IX SHIFTED ONE BIT
C  TO THE RIGHT
C
C     MOVE 31 BITS FROM BIT POSITION 32 OF IX(I) INTO IY(I) AND RIGHT
C     JUSTIFY
      IY(1)=ILOAD(IX(1),32,31)
      DO 20 I=2,N
      IY(I)=ILOAD(IX(I),32,31)
C
C     MOVE RIGHTMOST BIT FROM IX(I-1) INTO IY(I) STARTING AT BIT
C     POSITION 32
      IY(I)=ISTORE(IX(I-1),IY(I),32,1)
20    CONTINUE
      RETURN
      END
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
ADDR2  STMT   SOURCE STATEMENT                                       F01FEB73

          3 *
          4 * ENTRY POINTS TO MACHINE LANGUAGE FUNCTIONS
          5 *
          6             ENTRY  ILOAD          RIGHT JUSTIFY BINARY FIELD OF WORD
          7             ENTRY  ISTORE         PLACE FIELD IN BINARY FIELD OF WORD
          8             ENTRY  ICOMPL         1'S COMPLIMENT OF FIELD
          9             ENTRY  IAND           LOGICAL 'AND'
         10             ENTRY  IOR            LOGICAL INCLUSIVE 'OR'
         11             ENTRY  IEOR           LOGICAL EXCLUSIVE 'OR'


         13 *
         14 * SYMBOLIC REGISTERS 0 - 15
         15 *
         16 R0          EQU    0
         17 R1          EQU    1
         18 R2          EQU    2
         19 R3          EQU    3
         20 R4          EQU    4
         21 R5          EQU    5
         22 R6          EQU    6
         23 R7          EQU    7
         24 R8          EQU    8
         25 R9          EQU    9
         26 R10         EQU    10
         27 R11         EQU    11
         28 R12         EQU    12
         29 R13         EQU    13
         30 R14         EQU    14
         31 R15         EQU    15
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
ADDR2  STMT   SOURCE STATEMENT                                                     FO1FEB73

         33 *
         34 *
         35 ***********************************************************************
         36 ****************************   ILOAD   *********************************
         37 ***********************************************************************
         38 *
         39 *
         40 * ILOAD
         41 *
         42 *
         43 *          THE ILOAD FUNCTION WILL MOVE A FIELD OF DATA FROM THE SOURCE
         44 * WORD AND WILL RIGHT JUSTIFY IT AS THE OUTPUT ARGUMENT.  THE REST OF
         45 * THE OUTPUT ARGUMENT WORD WILL BE ZEROED.
         46 *
         47 *
         48 * ENTRY IS BY      FUNCTION ILOAD(SOURCE,SB,NB)
         49 *
         50 *          WHERE SOURCE = INTEGER, THE SOURCE WORD FROM WHICH TO OBTAIN
         51 *                               THE DATA FIELD
         52 *
         53 *                SB     = INTEGER DENOTING THE LEFT MOST BIT OF THE
         54 *                               SOURCE WORD FROM WHICH TO OBTAIN THE
         55 *                               FIELD.
         56 *                                          LEAST SIGNIFICANT BIT = 1
         57 *
         58 *                NB     = INTEGER DENOTING THE NUMBER OF BITS OF THE
         59 *                               SOURCE WORD TO RETAIN AND RIGHT JUSTIFY.
         60 *
         61 LOGFUNC  CSECT                    NAME OF LOAD MODULE
         62 ILOAD    EQU    *
         63          SAVE   (14,12),,ILOAD
0000A    64+         B      10(0,15) BRANCH AROUND ID
         65+         DC     AL1(5) LENGTH OF IDENTIFIER
         66+         DC     CL5'ILOAD' IDENTIFIER
0000C    67+         STM    14,12,12(13) SAVE REGISTERS
         68          USING  ILOAD,R15     R15 IS BASE REGISTER
00000    69          LM     R1,R3,0(R1)   R1 = ADR OF SOURCE WORD
         70 *                             R2 = ADR OF SB
         71 *                             R3 = ADR OF NB
00000    72          L      R0,0(,R1)     R0 = SOURCE WORD
00020    73          LA     R1,32
00002    74          SH     R1,2(,R2)     R1 = 32 - SB
00000    75          SLL    R0,0(R1)      LEFT JUSTIFY SOURCE WORD
00020    76          LA     R1,32
00002    77          SH     R1,2(,R3)     R1 = 32 - NB
00000    78          SRL    R0,0(R1)      RIGHT JUSTIFY SOURCE WORD
00118    79          B      ENDFUNC       GO TO END SEQUENCE
         80 *
         81 *
```

```
*
*
*****************************************************************
***************************  ISTORE  ****************************
*****************************************************************
*
* ISTORE
*
*
*            THE ISTORE FUNCTION ACTS AS A FUNCTION AND MOVES A RIGHT
* ADJUSTED FIELD OF DATA FROM WORD SRC1 AND SCALES IT AS SPECIFIED.  IT
* THEN REPLACES THE SPECIFIED PORTION OF SRC2 WITH THAT FIELD. THE WORD
* THUS FORMED BECOMES THE OUTPUT ARGUMENT.
*
*
* ENTRY IS BY       FUNCTION ISTORE(SRC1,SRC2,SB,NB)
*
*          WHERE SRC1 = WORD FROM WHICH TO GET THE RIGHT ADJUSTED FIELD.
*
*                SRC2 = WORD IN WHICH TO REPLACE THE SPECIFIED FIELD
*                       WITH THE ADJUSTED FIELD OBTAINED FROM SRC1.
*
*                SB   = STARTING BIT OF FIELD BEING MOVED TO THE
*                       OUTPUT ARGUMENT.  BIT 1 = LEAST SIGNIFICANT BIT
*
*                NB   = NUMBER OF BITS TO MOVE IN THE FIELD
*
*
*
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```
ADDR2  STMT    SOURCE STATEMENT                                          F01FEB73

        114 ISTORE   EQU    *
        115          SAVE   (14,12),,ISTORE
000000  116+         B      12(0,15) BRANCH AROUND ID
        117+         DC     AL1(6) LENGTH OF IDENTIFIER
        118+         DC     CL6'ISTORE' IDENTIFIER
00000C  119+         STM    14,12,12(13) SAVE REGISTERS
        120          USING  ISTORE,R15      R15 IS BASE REGISTER
00000   121          LM     R1,R4,0(R1)     R1 = ADR OF RJ FIELD
        122 *                               R2 = ADR OF DESTINATION
        123 *                               R3 = ADR OF SB
        124 *                               R4 = ADR OF NB
0012C   125          L      R7,MSKALLF
00000   126          L      R6,0(,R1)       R6 = RIGHT ADJUSTED FIELD
00000   127          L      R0,0(,R2)       R0 = DESTINATION WORD
00020   128          LA     R1,32
00002   129          SH     R1,2(,R4)       R1 = 32 - NB
00000   130          SLL    R6,0(R1)        LEFT JUSTIFY RJ FIELD
00000   131          SLL    R7,0(R1)        LEFT JUSTIFY MASK
00020   132          LA     R1,32
00002   133          SH     R1,2(,R3)       R1 = 32 - SB
00000   134          SRDL   R6,0(R1)        POSITION FIELD AND MASK
0012C   135          X      R7,MSKALLF      CHANGE BITS IN MASK FOR INSERTION
        136          NR     R0,R7           CLEAR FIELD IN DESTINATION WORD
        137          OR     R0,R6           INSERT NEW FIELD IN DESTINATION WORD
00118   138          B      ENDFUNC         GO TO END SEQUENCE
        139 *
        140 *
```

```
ADDR2  STMT   SOURCE STATEMENT                                               F01FEB73

       142 *
       143 *
       144 ***********************************************************************
       145 ******************************   ICOMP1   ******************************
       146 ***********************************************************************
       147 *
       148 *          THE FUNCTION ICOMP1 WILL OUTPUT A VALUE WHICH IS THE INPUT
       149 * VALUE SOURCE WITH THE FIELD STARTING WITH SB BIT FOR NB BITS
       150 * 1'S COMPLEMENTED.
       151 *
       152 * ENTRY IS BY         CALL ICOMP1(SOURCE,SB,NB)
       153 *
       154 *          SOURCE = INTEGER = THE INPUT WORD IN WHICH A FIELD IS TO BE
       155 *                              1'S COMPLEMENTED.  SOURCE WILL NOT BE
       156 *                              ALTERED.
       157 *
       158 *          SB     = INTEGER = THE STARTING (LEFTMOST) BIT OF WRDIN TO
       159 *                              1'S COMPLEMENT.
       160 *                              LEAST SIGNIFICANT BIT = 1
       161 *
       162 *          NB     INTEGER = THE NUMBER OF BITS TO 1'S COMPLEMENT.
       163 *
       164 *
       165 ICOMP1    EQU   *
       166           SAVE  (14,12),,ICOMP1
0000C  167+          B     12(0,15) BRANCH AROUND ID
       168+          DC    AL1(6) LENGTH OF IDENTIFIER
       169+          DC    CL6'ICOMP1' IDENTIFIER
0000C  170+          STM   14,12,12(13) SAVE REGISTERS
       171           USING ICOMP1,R15        R15 IS BASE REGISTER
00000  172           LM    R1,R3,0(R1)       R1 = ADR OF SOURCE WORD
       173 *                                 R2 = ADR OF SB
       174 *                                 R3 = ADR OF NB
00000  175           L     R0,0(,R1)         R0 = SOURCE WORD
       176           LR    R4,R0             R4 = R0
0012C  177           L     R5,MSKALLF        R5 = FFFF FFFF
00020  178           LA    R1,32
00002  179           SH    R1,2(,R2)         R1 = 32 - SB
00020  180           LA    R2,32
00002  181           SH    R2,2(,R3)         R2 = 32 - NB
       182           LR    R3,R5             R3 = R5
00000  183           SLL   R5,0(R2)
00000  184           SRL   R5,0(R1)          POSITION FIELD
       185           XR    R4,R3             COMPLIMENT FULL WORD
       186           NR    R4,R5             LEAVE ONLY FIELD IN WORD
       187           XR    R5,R3             CHANGE MASK
       188           NR    R0,R5             ZERO OUT FIELD IN DESTINATION SOURCE
       189           OR    R0,R4             INSERT 1'S COMPLIMENT FIELD IN WORD
00118  190           B     ENDFUNC           GO TO END SEQUENCE
       191 *
       192 *
       193 *
```

C-4

```
ADDR2  STMT   SOURCE STATEMENT                                                F01FE873

       195 *
       196 *
       197 ***********************************************************************
       198 ****************************** LOGFUN    **** IAND, IOR, IEOR *********
       199 ***********************************************************************
       200 *
       201 * LOGICAL FUNCTIONS
       202 *          IAND      LOGICAL AND
       203 *          IOR       INCLUSIVE LOGICAL OR
       204 *          IEOR      EXCLUSIVE LOGICAL OR
       205 *
       206 *          THE LOGICAL FUNCTIONS PERFORM THE REQUIRED OPERATION ON THE
       207 * FIRST INPUT PARAMETER BY THE BIT PATTERN OF THE SECOND PARAMETER AND
       208 * RETURN THE RESULT TO THE CALLING ROUTINE.
       209 *
       210 * ENTRY IS BY         FUNCTION (WORD,MASK)
       211 *
       212 *                     WHERE FUNCTION = IAND, IOR, OR IEOR
       213 *                              WORD     = INPUT SOURCE WORD
       214 *                              MASK     = PATTERN FOR FUNCTION
       215 *
       216 *
       217 *          THESE LOGICAL FUNCTION FOUTINES WILL PERFORM THE REQUIRED
       218 * FUNCTION AND RETURN THE VALUE IN REGISTER 0. REGISTER 14 IS USED FOR
       219 * THE RETURN. ALL REGISTERS USED ARE RESTORED TO THEIR ORIGINAL VALUE.
       220 *
       221 *
       222 *
       223 IAND     EQU   *
       224          SAVE  (14,12),,IAND
0000A  225+         B     10(0,15) BRANCH AROUND ID
       226+         DC    AL1(4) LENGTH OF IDENTIFIER
       227+         DC    CL4'IAND' IDENTIFIER
0000C  228+         STM   14,12,12(13) SAVE REGISTERS
       229          USING IAND,R15          R15 IS BASE REGISTER
```

```
ADDR2  STMT   SOURCE STATEMENT                                                   F0[illegible]873

       231 *
       232 *
C0000  233            LM     R2,R3,0(R1)      R2 = ADR OF WORD
       234 *                                  R3 = ADR OF MASK
00000  235            L      R0,0(,R2)        R0 = WORD
00000  236            N      R0,0(,R3)        AND WORD WITH MASK
00118  237            B      ENDFUNC          GO TO END SEQUENCE
       238 *
       239 *
       240 *
       241 *
       242 *
       243 *
       244 IOR        EQU    *
       245            SAVE   (14,12),,IOR
00000  246+           B      8(0,15) BRANCH AROUND ID
       247+           DC     AL1(3) LENGTH OF IDENTIFIER
       248+           DC     CL3'IOR' IDENTIFIER
C000C  249+           STM    14,12,12(13) SAVE REGISTERS
       250            USING  IOR,R15          R15 IS BASE REGISTER
00000  251            LM     R2,R3,0(R1)      R2 = ADR OF WORD
       252 *                                  R3 = ADR OF MASK
00000  253            L      R0,0(,R2)        R0 = WORD
00000  254            O      R0,0(,R3)        OR WORD WITH MASK
00118  255            B      ENDFUNC          GO TO END SEQUENCE
       256 *
       257 *
       258 *
       259 *
       260 IEOR       EQU    *
       261            SAVE   (14,12),,IEOR
0000A  262+           B      10(0,15) BRANCH AROUND ID
       263+           DC     AL1(4) LENGTH OF IDENTIFIER
       264+           DC     CL4'IEOR' IDENTIFIER

0000C  265+           STM    14,12,12(13) SAVE REGISTERS
       266            USING  IEOR,R15         R15 IS BASE REGISTER
00000  267            LM     R2,R3,0(R1)      R2 = ADR OF WORD
       268 *                                  R3 = ADR OF MASK
00000  269            L      R0,0(,R2)        R0 = WORD
00000  270            X      R0,0(,R3)        EXCLUSIVE 'OR' WORD WITH MASK
00118  271            B      ENDFUNC          GO TO END SEQUENCE
       272 *
       273 *
```

```
ADDR2  STMT    SOURCE STATEMENT                                                  F01FEB73

         275 *
         276 *
         277 ENDFUNC  EQU     *
00014    278          ST      R0,20(,R13)     STORE RESULT BEFORE LOADING REGISTERS
         279          RETURN (14,12),T,RC=0
0000C    280+         LM      14,12,12(13) RESTORE THE REGISTERS
         281+         MVI     12(13),X'FF' SET RETURN INDICATION
00000    282+         LA      15,0(0,0) LOAD RETURN CODE
         283+         BR      14 RETURN
         284 *
         285 *
         286          CNOP    0,4
         287 MSKALLF  DC      X'FFFFFFFF'     MASK OF ALL 1'S (MINUS 1)
         288 *
         289          END
```

IDENTIFICATION OF CONNECTED REGIONS

I. **NAME**

REGION

II. **DESCRIPTION**

This subroutine identifies all distinct connected regions in an image given the boundary data in SLIC (scan line intersection code) format and produces a map with a number at each point showing the region to which it belongs. The region numbers will be in descending order of area.

III. **CALLING SEQUENCE**

CALL REGION (IX, IH, ISEQ, IW1, IW2, ITABL, IS, LW, IDENT, NREC, NEL, MR)

where

IX, IH, ISEQ, IW1, IW2, ITABL, IS, LW, and IDENT are arrays dimensioned as follows:

IX (MAX (NEL, NR))	
IH (MR, 20)	FULL WORDS
ISEQ (NR+1)	
IW1 (NEL)	
IW2 (NEL)	
ITABL (MR, 20, 2)	HALF WORDS
IS (MR)	
LW (MR)	
IDENT (MR, MR)	BYTES

and NREC, NEL are the number of records and the number of pixels (bytes) per record in the input image;

MR is the maximum number of region identifiers permissible in a segment; and

NR is the maximum number of regions expected.

IV. <u>INPUT/OUTPUT</u>

1. INPUT

The input to this program is a sequential data set on logical unit 8, having NREC records stored as N, (IX(J), J=1, N) in unformatted FORTRAN mode.

2. OUTPUT

The output of this program will be a sequential data set on logical unit 12, having NREC records with NEL pixels each with one half-word (2 bytes) per pixel.

3. FILE STORAGE

This program requires a sequential access data set with NREC records and NEL half-words per record.

V. <u>DESCRIPTION OF SUBROUTINES</u>

The subroutines called by REGION are given.

EXTERNAL LINKAGES

CALLING PROGRAM	PROGRAMS CALLED
REGION	RIDER 1 SORTLS
RIDER1	RIDER2 RIDER3 RIDER4
RIDER4	RIDER5 RIDER6 RIDER7

A brief description of the function of each subroutine and its storage requirements are given below.

DESCRIPTION OF SUBROUTINES

SUBROUTINE OR ENTRY	STORAGE (BYTES)	FUNCTION
REGION	2112	Driver program for identifying regions; rearranges region numbers in population order and writes output.
RIDER1	2012	Initialize arrays, read input, compute histogram.
RIDER2	878	Find current array containing region identification numbers.
RIDER3	562	Changes region numbers such that consecutive region numbers are used.
RIDER4	1500	Handles processing to preserve information when a new segment is begun.
RIDER5	818	Generate a lookup table based on the IDENT matrix.
RIDER6	472	Find the set of distinct region numbers in the last record of a segment.
RIDER7	564	Modify lookup tables for earlier segments based on new connectivities.
SORTSL (SORTLS)	1684	Sort one array in decreasing order and arrange a second array accordingly.

VI. PERFORMANCE SPECIFICATION

1. STORAGE

For a given number of regions, the storage required is dependent on the data record length and the number of regions allocated to a segment. For 300 regions and a record length of 1000, the program requires 190K when all the regions are placed in one segment. If only 50 regions are allocated to a segment, the core required is reduced to 64K.

2. EXECUTION TIME

The data set used was a digitized LACIE ground truth map of 820 lines by 1000 samples, consisting of 260 regions. For one segment, the time required was 50 seconds, and for 50 regions per segment, the time increased to 53 seconds.

3. RESTRICTIONS

In order to ensure that all regions are closed, the edge pixels on each side of the input image should be boundary pixels.

VII. METHOD

A simplified flow diagram for subroutine RIDER2 which identifies new regions of an image is shown in Figure 25 .

The total program has four major sections:

(i) Finding a preliminary set of region identifiers;

(ii) Finding the areas of each of the regions;

(iii) Generating a mapping such that the region numbers are used in the order of decreasing areas;

(iv) Modifying the region number by table lookup.

1. FINDING PRELIMINARY REGION IDENTIFIERS

This is the most important step in the program. The subroutine RIDER2 is used for this purpose. The present version can handle up to MR*MSEG distinct regions while still using a "region identity matrix" of size MR by MR. (MSEG is currently set to 20.)

This routine uses the arrays IW1 and IW2 as the previous and current records of region identifiers. By convention, region number 0 indicates the boundary points. The MR by MR array IDENT is used to store information about the identity of regions, IDENT(I,J) = .TRUE. meaning that region numbers I and J refer to the same connected region.

Initally, the array IW1 is set to all 1's and IDENT is set to all .FALSE.. Each of the input records is read and the following operations are performed.

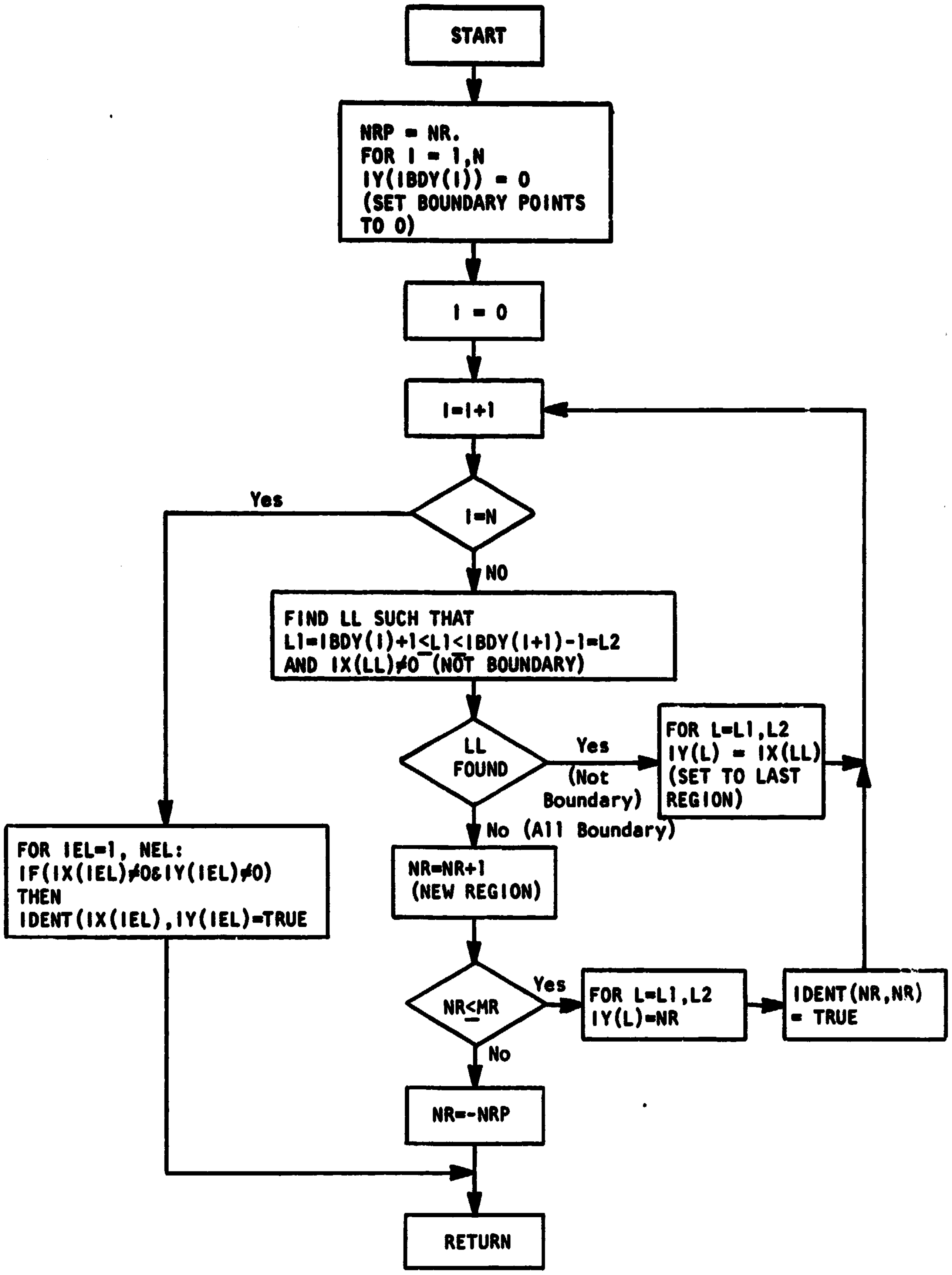


Figure 25. Simplified Flow Diagram for Subroutine RIDER2

The boundary coordinates in the input record are arranged in ascending order. The routine RIDER2 is used to generate, in IW2, the region identification numbers corresponding to the present row. First, all the elements of IW2 corresponding to the boundary coordinates are set to zero. Each interval between the zeros is compared with the corresponding segment of IW1. If all of the elements in that section of IW1 are boundary points, a new region number is started and assigned to the interval in IW2. If there is a break in the boundary connection to a region, that number is propagated to all elements in the interval. Finally, IDENT(IW1(I), IW2(I)) is set to .TRUE. for I=1, NEL wherever IW1(I)≠0 and IW2(I)≠0, indicating that IW1(I) and IW2(I) refer to the same region. Also, when new region identifiers are to be used, the routine RIDER2 verifies whether the number of identifiers exceeds MR. If so, the value of NR, the total number identifiers, is set to -NRP, the total number up to the previous record and the control goes back to the routine RIDER1.

Now if RIDER2 returns a positive NR, the array IW2 is written as the I'th record on the sequential access data set (unit number NDEVM) and IW2 is moved into IW1 (so that it becomes the "previous" record while handling the next record).

If RIDER2 returns a negative NR, then NR is changed to -NR and the routine RIDER4 is called. The set of records handled between any two calls of RIDER4 will be referred to as a segment. Associated with each segment, a table is defined which gives a mapping from the set of region identifiers obtained in that segment to a new set reflecting the connectivities discovered up to the most recent segment handled. Also, the initial record number for each of the segments is stored in an array. The functions of the routine RIDER4 are to:

(i) Reduce the matrix IDENT (using RIDER5) examining all of the available connectivity information in it and obtain a lookup table for the current segment;

(ii) Modify the tables for the previous segments to reflect the newly found connectivities, if any;

(iii) Find all the distinct region numbers occurring in the last record IW1 of the current segment and change the numbers there which are greater than 0 to consecutive numbers starting with 1 (let NR be the largest number in IW1);

(iv) Set up an array IS consisting of the distinct region numbers in IW1 and then change IS(I) to ITABL(IS(I), ISEG) where ITABL is the lookup table for the current segment;

(v) Set all elements of IDENT to .FALSE. except when IS(I) = IS(J) for I, J in the range 1 through NR.

After each call to RIDER4, the segment count ISEG is incremented, and the initial record number for the next segment (which is really the number at which RIDER4 had to be called) is stored in IRES(ISEG). If MSEG is exceeded by ISEG, or if NR>MR (which means there are more than MR distinct regions in the last record) the routine RIDER1 prints an error message, sets NR = 0 and exits. Otherwise, RIDER2 is called again, IW2 is found and written on NDEVM, and the program proceeds normally to the next input record.

After the NREC input records have been processed, the routine RIDER4 is called to get the lookup table for the final segment. A call to RIDER3 changes the lookup tables for all the segments such that consecutive region numbers are used.

2. FINDING AREAS

A histogram of the region identification maps is found, giving the total number of occurrences of each of the region identifiers 0 through NR. These numbers indicate the areas of the regions.

3. FINDING THE FINAL LOOKUP TABLE

A sequence of natural numbers is used as a secondary array with the histogram as the primary array in a descending sort operation (routine SORTLS). The resulting secondary array then gives the sequence of original region identifiers corresponding to decreasing areas. An inverse mapping (inverse mapping of (IX(J) J = 1, N) is defined as (IY(J) J=1,N) if IY (IX(J))=J) of this sequence gives the final lookup table. The actual coding follows these principles but is slightly different in detail to preserve the identity of region 0 which has special significance (boundary lines).

4. DERIVING THE FINAL REGION IDENTIFICATION

The lookup tables generated above are used to modify the region identifiers on NDEVM, record by record, and write out the final sequential data set on unit 12.

VIII. COMMENTS

Another approach could be used instead of the one described above. With that method, the processing would be identical, except that the matrix IDENT is not defined. Instead, a table is updated every time a new connectivity is discovered. While this saves storage, it appears to take more execution time than the present method.

IX. TESTS

This program has been tested on the LACIE GTM and found to work satisfactorily. A segment of the output is shown in the following figure.

X. LISTINGS

The listing of the main program and the associated routines are attached at the end of this section.

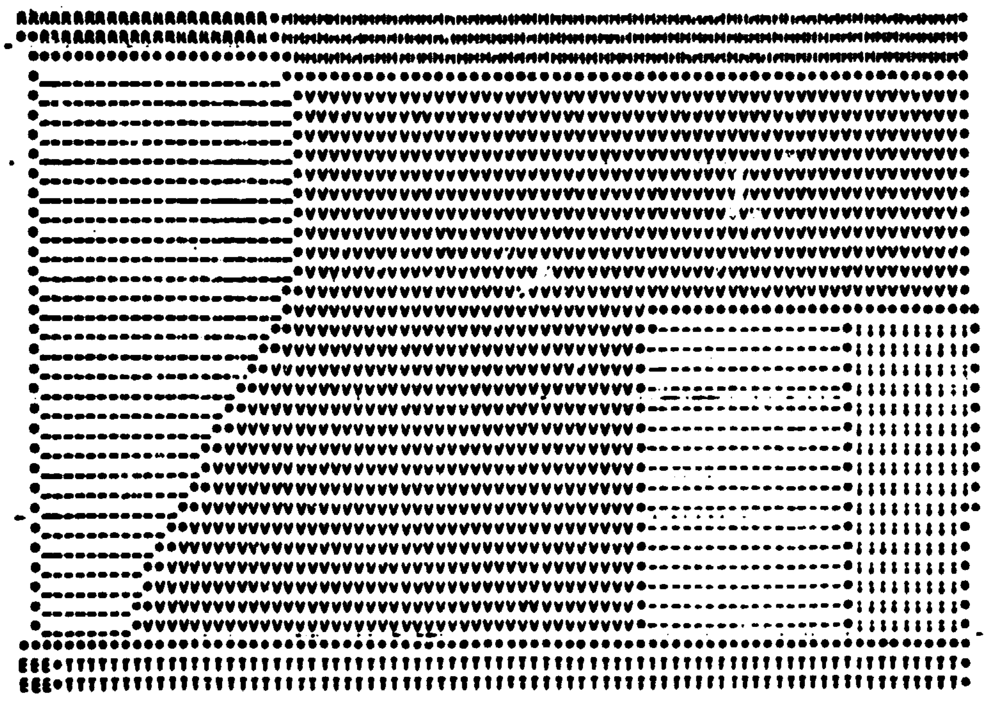


Figure 26. Plot of Distinct Regions as Determined by Subroutine REGION

```
      SUBROUTINE REGION (IX, IH, ISEG, IW1, IW2, ITABL, IS, LW, IDENT,
     *                  NREC, NEL, MR)
C
C  THIS SUBROUTINE IDENTIFIES ALL DISTINCT REGIONS IN AN IMAGE AND
C  PRODUCES A MAP WITH A NUMBER AT EACH POINT SHOWING THE REGION TO
C  WHICH IT BELONGS
C
C     D'N IX(MAX(N,NR)),ISEQ(NR) WHERE NR=MAX.NO.OF REGIONS EXPECTED AND
C         N= MAX NO. OF BOUNDARY POINTS EXPECTED IN ANY RECORD
C     D'N ITABL(MR,MSEG),IS(MCR) WHERE MSEG IS THE MAXIMUM
C     NUMBER OF SEGMENTS EXPECTED FOR HANDLING THE GIVEN BOUNDARY IMAGE
C     MCR=MAX NUMBER OF REGION NUMBERS EXPECTED TO OCCUR IN ANY RECORD.
      DIMENSION IX(1), IH(MR,1), ISEQ(1), IRES(20)
      INTEGER*2 IW1(NEL), IW2(NEL), ITABL(MR,1), IS(MR), LW(MR)
      LOGICAL*1 IDENT(MR,MR)
      DATA NDEVI, NDEVM, NDEVO /8, 10, 12/
C
C     CALL REGION IDENTIFICATION ROUTINE
      WRITE(6,100) NREC,NEL
      CALL RIDER1 (IX, IH, IRES, IW1, IW2, ITABL, IS, LW, IDENT, NREC,
     *             NEL, MR, NR, ISEG, NDEVI, NDEVM)
C
C     COMPUTE FINAL HISTOGRAM FOR ALL SEGMENTS USING 'ITABL'
      NR1 = NR + 1
      DO 5 I=2,NR1
5     IX(I) = 0
      NTOT = 0
      DO 10 I=1,MR
      DO 10 J=1,ISEG
      NREG = ITABL(I,J) + 1
      IF (NREG.GT.1) IX(NREG) = IX(NREG) + IH(I,J)
      NTOT = NTOT + IH(I,J)
10    CONTINUE
      IX(1) = NREC*NEL - NTOT
C
C     PRINT HISTOGRAM OF REGION IDENTIFICATION MAP
      WRITE(6,200)
      DO 40 I=1,NR1
      J=I-1
      WRITE (6,350) J, IX(I)
40    CONTINUE
C
C     REARRANGE REGION NUMBERS IN DESCENDING ORDER OF POPULATIONS.
C     LEAVE 0 UNCHANGED SINCE IT CORRESPONDS TO BOUNDARY POINTS
      DO 20 I=1,NR1
      ISEQ(I)=I-1
20    CONTINUE
      CALL SORTLS (IX(2), ISEQ(2), 1, NR)
      WRITE(6,400)
      DO 50 I=1,NR1
      J=I-1
      WRITE (6,350) J, IX(I), ISEQ(I)
50    CONTINUE
      DO 60 I=1,NR1
      NSORT = ISEQ(I)
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```
      IX(NSORT+1) = I-1
60    CONTINUE
C
C     MODIFY REGION NUMBERS ACCORDING TO TABLES 'ITABL' AND 'IX'
      REWIND NDEVM
      JSEG = 0
      DO 70 IREC=1,NREC
      READ (NDEVM) IW1
      IF (IREC.EQ.IRES(JSEG+1)) JSEG = JSEG + 1
      DO 80 IEL=1,NEL
      I=IW1(IEL)
      IF (I.EQ.0) J = 0
      IF (I.NE.0) J = ITABL(I,JSEG)
      IW1(IEL) = IX(J+1)
80    CONTINUE
      WRITE (NDEVO) IW1
70    CONTINUE
      RETURN
C
100   FORMAT(//' IMAGE SIZE=('I5,',',I5,')')
200   FORMAT(//10X'REGION NO.'10X'NO. OF PIXELS')
350   FORMAT(11XI6,16XI9,17XI6)
400   FORMAT('1',10X,'REGIONS AFTER REASSIGNMENTS:'//10X,'REGION NO.',
     ,10X,'NO. OF PIXELS',10X,'OLD REGION NO.')
      END
      SUBROUTINE RIDER1 (IBDY, IH, IRES, IW1, IW2, ITABL, IS, LW, IDENT,
     .                   NREC, NEL, MR, NR, ISEG, NDEVI, NDEVM)
C
C     TO IDENTIFY ALL DISTINCT CONNECTED REGIONS IN A PICTURE SEPARATED
C     BY BOUNDARY LINES,THE BOUNDARY DATA ARE GIVEN AS NREC RECORDS ON A
C     SEQUENTIAL FILE NDEVI, EACH RECORD BEING WRITTEN AS
C     N,(IBDY(I),I=1,N)
C     THE OUTPUT OF THE PROGRAM IS AN NREC*NEL SEQUENTIAL ACCESS FILE ON
C     NDEVO CONSISTING OF 0' S FOR BOUNDARY POINTS AND DISTINCT REGION
C     NUMBERS FOR EACH OF THE CONNECTED REGIONS.
C
      DIMENSION IBDY(1), IH(MR,1), IRES(20)
      INTEGER*2 IW1(NEL), IW2(NEL), ITABL(MR,1), IS(MR), LW(MR)
      LOGICAL*1 IDENT(MR,MR)
      DATA MSEG /20/
C
C     INITIALIZE A WORK ARRAY IW1 WITH 1'S AND IDENT WITH .FALSE.
      DO 2 I=1,NEL
    2 IW1(I) = 1
      DO 4 I=1,MR
      DO 4 J=1,MR
    4 IDENT(I,J) = .FALSE.
      DO 6 I=1,MR
      DO 6 J=1,MSEG
    6 IH(I,J) = 0
      REWIND NDEVM
      ISEG=1
      IRES(1)=1
      NR=0
C
C     LOOP ON RECORDS
      DO 10 IREC=1,NREC
C
C     READ ONE RECORD OF BOUNDARY INFORMATION
      READ(NDEVI)N,(IBDY(I),I=1,N)
C
C     DESIGNATE ALL BOUNDARY POINTS AS 'REGION 0'
      DO 8 I=1,N
      J=IBDY(I)
    8 IW2(J) = 0
C
C     USE IW1 AND IBDY TO SET ARRAY IW2 AND MATRIX IDENT
      N1=N-1
   30 CONTINUE
      CALL RIDER2 (IBDY, IW1, IW2, IDENT, ISEG, N1, NEL, MR, NR)
      IF(NR.GT.-1) GO TO 20
      WRITE(6,200) IREC,NR
      IF(IREC.EQ.IRES(ISEG))GO TO 40
      NR=-NR
      CALL RIDER4(IDENT,LW,MR,NR,ITABL,ISEG,I-1,NEL,IS,.FALSE.)
      ISEG=ISEG+1
      IF(ISEG.GT.MSEG)GO TO 40
      IRES(ISEG)=IREC
      IF(NR.LE.MR)GO TO 30
C
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```
   20 WRITE (NDEVM) IW2
      DO 21 I=1,NEL
   21 IW1(I) = IW2(I)
      DO 25 I=1,N1
      L1 = IBDY(I) + 1
      L2 = IBDY(I+1) - 1
      IF (L1.GT.L2) GO TO 25
      KR = IW2(L1)
      IH(KR,ISEG) = IH(KR,ISEG) + L2 - L1 + 1
   25 CONTINUE
   10 CONTINUE
C
C     OBTAIN LOOKUP TABLE FOR THE FINAL SEGMENT
      CALL RIDER4(IDENT,LW,MR,NR,ITABL,ISEG,IW1,NEL,IS,.TRUE.)
      CALL RIDER3 (ITABL, MR*ISEG)
C
C     FIND HIGHEST REGION NUMBER
      NR=0
      DO 35 I=1,MR
      DO 35 J=1,ISEG
   35 IF (ITABL(I,J).GT.NR) NR = ITABL(I,J)
      RETURN
C
   40 WRITE (6,100) IREC
      NR=0
      RETURN
C
100   FORMAT(' ERROR CONDITION IN RIDER,  SUPPLIED MR OR MSEG WAS EXCEED
     .ED AT RECORD NUMBER'I6,',  RETURNING WITH NR=0')
200   FORMAT(' (IREC, NR) = '2I6)
      END
      SUBROUTINE RIDER2 (IBDY, IX, IY, IDENT, ISEG, N1, NEL, MR, NR)
C
C     GIVEN CURRENT SET OF BOUNDARY ADDRESSES (IBDY(I),I=1,N) AND THE
C     LAST ARRAY IX, FIND CURRENT ARRAY IY CONTAINING REGION IDENTIFICA-
C     TION NUMBERS.  ALSO, IF THE NONBOUNDARY ELEMENTS IN CURRENT ROW
C     ARE CONTIGUOUS WITH ANY NONBOUNDARY POINTS OF THE LAST ROW, SET
C     THE CORRESPONDING ELEMENTS IN IDENT MATRIX.
C
      DIMENSION IBDY(1)
      INTEGER*2 IX(NEL),IY(NEL)
      LOGICAL*1 IDENT(MR,MR)
C
C     FOR I=1,N-1 EXAMINE IX(J) FOR IBDY(I).LT.J.LT.IBDY(I+1)
C     AND SET IY ACCORDINGLY.  A NEW REGION NUMBER IS STARTED WHEN IX(J)
C     IS 0 FOR ALL J IN THE ABOVE RANGE.
      NRP=NR
      DO 40 I=1,N1
      L1=IBDY(I)+1
      L2=IBDY(I+1)-1
      IF(L1.GT.L2)GO TO 40
C
C     LOOP OVER PIXELS BETWEEN BOUNDARY POINTS AT L1 AND L2
      DO 50 L=L1,L2
      IF(IX(L).EQ.0)GO TO 50
C
C     GO TO 60 IF IX(L) IS NOT A BOUNDARY
      GO TO 60
   50 CONTINUE
C
C  START NEW REGION IF PREVIOUS LINE (IX) IS ALL BOUNDARY POINTS
      NR=NR+1
      IF (NR.GT.MR) GO TO 90
      DO 55 K=L1,L2
   55 IY(K) = NR
      IDENT(NR,NR)=.TRUE.
      GO TO 40
C
C  IF IX(L) IS NOT A BOUNDARY, SET IY FROM L1 TO L2 TO IX(L)
   60 CONTINUE
      DO 61 K=L1,L2
   61 IY(K) = IX(L)
C
   40 CONTINUE
C
C     SET IDENT MATRIX TO INDICATE REGION NUMBERS CORRESPONDING TO
C     IDENTICAL REGIONS.
   20   CONTINUE
      DO 80 IEL=1,NEL
      I=IX(IEL)
      IF(I.EQ.0)GO TO 80
      J=IY(IEL)
      IF(J.EQ.0)GO TO 80
      IDENT(I,J)=.TRUE.
   80 CONTINUE
      RETURN
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
C
   90 CONTINUE
      NR=-NRP
      RETURN
      END
      SUBROUTINE RIDER3 (ITABL, NRSEG)
C
C     CHANGE REGION NUMBERS SUCH THAT CONSECUTIVE NUMBERS ARE USED.
C
      INTEGER*2 ITABL(NRSEG,2)
C
C     FIND THE SET OF NUMBERS IN ITABL(*,1).
      DO 5 I=1,NRSEG
      ITABL(I,2)=0
5     CONTINUE
      DO 10 I=1,NRSEG
      J=ITABL(I,1)
      IF (J.NE.0) ITABL(J,2) = 1
10    CONTINUE
C
C     CHANGE ITABL(*,2) TO GET A LOOKUP TABLE FOR ITABL(*,1).
      J=0
      DO 20 I=1,NRSEG
      IF(ITABL(I,2).EQ.0)GO TO 20
      J=J+1
      ITABL(I,2)=J
20    CONTINUE
C
C     CHANGE ITABL(*,1).
      DO 30 I=1,NRSEG
      ITABL(I,1)=ITABL(ITABL(I,1),2)
30    CONTINUE
      RETURN
      END
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
      SUBROUTINE RIDER4(IDENT,LW,MR,NR,ITABL,ISEG,IW1,NEL,IS,LAST)
C
C     THIS ROUTINE IS CALLED FROM RIDER1 WHEN ALL RECORDS ARE PROCESSED
C     (LAST=.TRUE.) OR WHEN THE NUMBER OF REGION NUMBERS FOUND WHILE
C     TESTING (IREC+1)'TH RECORD EXCEEDS MR. (LAST=.FALSE.).  THEN,
C          1. THE REGION CONNECTIVITY MATRIX IDENT IS REDUCED TO GET A
C     LOOKUP TABLE FOR THE CURRENT SEGMENT.
C         2. THE LOOK-UP TABLES CORRESPONDING TO EARLIER SEGMENTS ARE
C     MODIFIED BASED ON NEWLY FOUND CONNECTIVITIES, IF ANY.
C          3. THE DISTINCT REGION NUMBERS OCCURRING IN THE IREC'TH SEG.
C     ARE FOUND; A CORRESPONDENCE ARRAY IS BETWEEN CURRENT AND NEXT
C     SEGMENT SET UP.  THE LAST RECORD(IW1) IS MODIFIED TO MATCH THE
C     NUMBERING OF THE NEXT SEGMENT.
C        4. THE CONNECTIVITY MATRIX IS MODIFIED TO PRESERVE THE INFORMA-
C     TION ON THE CONNECTIONS BETWEEN REGIONS IN IREC'TH RECORD.
C
      INTEGER*2 ITABL(MR,1),LW(MR),IS(1),IW1(NEL)
      LOGICAL*1 IDENT(MR,MR)
      LOGICAL LAST
C
C                SECTION 1.
      DO 10 I=1,NR
      DO 10 J=1,NR
      IDENT(I,J)=IDENT(I,J).OR.IDENT(J,I)
10    CONTINUE
      CALL RIDER5(IDENT,MR,NR,ITABL(1,ISEG),LW)
      ISEGT=(ISEG-1)*MR
      DO 20 I=1,NR
      ITABL(I,ISEG) = ITABL(I,ISEG) + ISEGT
20    CONTINUE
      IF (MR.LE.NR) GO TO 50
      NR1 = NR + 1
      DO 30 I=NR1,MR
      ITABL(I,ISEG) = 0
30    CONTINUE
C
C               SECTION 2.
50    CONTINUE
      IF(ISEG.EQ.1)GO TO 60
      ISEG1=ISEG-1
      CALL RIDER7(ITABL(1,ISEG),IS,NCR,ITABL,MR*ISEG1)
C
C               SECTION 3.
60    IF(LAST)RETURN
      CALL RIDER6(IW1,NEL,IS,NR)
      NCR=NR
      DO 70 I=1,NR
      IS(I)=ITABL(IS(I),ISEG)
70    CONTINUE
C
C               SECTION 4.
C     CONNECTIVITIES BETWEEN NEW REGIONS I,J IN THE LAST RECORD ARE FOUN
C     BY TESTING WHETHER IS(I).EQ.IS(J)
      DO 80 I=1,MR
      DO 80 J=1,MR
```

```
80      IDENT(I,J) = .FALSE.
        DO 100 I=1,NR
        IDENT(I,I)=.TRUE.
        IF (I.EQ.NR) GO TO 100
        I1=I+1
        DO 90 J=I1,NR
90      IDENT(I,J)=IS(I).EQ.IS(J)
100     CONTINUE
        RETURN
        END
```

```
      SUBROUTINE RIDERS(IDENT,MD,N,IT,M)
C
C     TO GENERATE A TABLE IT MAPPING J=1,...,N TO I=IT(J)= SMALLEST K
C     SUCH THAT THERE EXISTS A SEQUENCE (K(ID),ID=1,...,L) WITH K(I)=I,
C     K(L)=J AND IDENT(K(ID),K(ID+1))=.TRUE.
C
      INTEGER*2 IT(N),M(1)
      LOGICAL*1 IDENT(MD,N)
C
      DO 100 I=1,N
      IT(I)=I
100   CONTINUE
      I=0
10    I=I+1
      IF(I.LE.N)GO TO 20
      RETURN
20    IF(IT(I).LT.I)GO TO 10
      J=0
      K=0
30    J=J+1
      IF(J.LE.N)GO TO 40
      L=0
50    L=L+1
      IF(L.GT.K)GO TO 10
      J=0
70    J=J+1
      IF(J.GT.N)GO TO 50
      IF(.NOT.IDENT(M(L),J))GO TO 70
      IF(IT(J).EQ.I)GO TO 70
      IT(J)=I
      K=K+1
      M(K)=J
      GO TO 70
40    IF(.NOT.IDENT(I,J))GO TO 30
      IT(J)=I
      K=K+1
      M(K)=J
      GO TO 30
      END
      SUBROUTINE RIDER6 (IX, NEL, IS, N)
C
C     FIND A SET IS OF DISTINCT NONZERO ELEMENTS IN IX.  THE NUMBER OF
C     SUCH ELEMENTS IS N.
C
      INTEGER*2 IX(NEL), IS(1)
C
      N = 0
      DO 10 I=1,NEL
      IF (IX(I).EQ.0) GO TO 10
      IF(N.EQ.0)GO TO 20
C
C     CHECK ELEMENTS OF IS ALREADY LOADED
      DO 30 J=1,N
      IF(IS(J).EQ.IX(I))GO TO 40
30    CONTINUE
C
C     CURRENT ELEMENT OF IX IS DISTINCT
20    N=N+1
      IS(N)=IX(I)
      IX(I)=N
      GO TO 10
C
C     CURRENT ELEMENT OF IX EQUALS J-TH DISTINCT ELEMENT
40    IX(I)=J
10    CONTINUE
      RETURN
      END
      SUBROUTINE RIDER7(IX,IS,N,IY,M)
C
C  MODIFY RELEVANT ENTRIES IN IY ACCORDING TO CONNECTIVITIES FOUND IN IS
C
      INTEGER*2 IX(N),IS(N),IY(M)
C
      IF (N.EQ.0) RETURN
      MAX = IS(1)
      MIN = IS(1)
      DO 5 I=1,N
      IF (IS(I).GT.MAX) MAX = IS(I)
    5 IF (IS(I).LT.MIN) MIN = IS(I)
C
C     CHECK FOR REGION NUMBERS OUTSIDE RANGE OF IY (PREVIOUS SEGMENTS)
      DO 10 J=1,M
      IF(IY(J).LT.MIN.OR.IY(J).GT.MAX)GO TO 10
C
C     MODIFY PREVIOUS TABLE (IY) USING CURRENT TABLE (IX)
      DO 20 I=1,N
      IF(IY(J).NE.IS(I))GO TO 20
      IY(J)=IX(I)
      GO TO 10
20    CONTINUE
10    CONTINUE
      RETURN
      END
      SUBROUTINE SORTSL (A1, A2, II, JJ)
C
C  SORT ARRAY A1 AND ARRANGE ARRAY A2 CORRESPONDINGLY
C  DO FROM ELEMENTS II TO JJ
C
      DIMENSION A1(1), A2(1), IU(16), IL(16)
      INTEGER A1, A2, T1, T2, TT1, TT2
      LOGICAL*1 SL
      SL = .TRUE.
      GO TO 1
C
      ENTRY SORTLS (A1, A2, II, JJ)
C
      SL = .FALSE.
    1 ND = JJ - II + 1
      M=1
      I=II
      J=JJ
    5 IF(I.GE.J) GOTO 70
   10 K=I
      IJ=(J+I)/2
      T1 = A1(IJ)
      T2 = A2(IJ)
      IF (A1(I).LE.T1) GO TO 20
      A1(IJ) = A1(I)
      A2(IJ) = A2(I)
      A1(I) = T1
      A2(I) = T2
      T1 = A1(IJ)
      T2 = A2(IJ)
   20 L=J
      IF (A1(J).GE.T1) GO TO 40
      A1(IJ) = A1(J)
      A2(IJ) = A2(J)
      A1(J) = T1
      A2(J) = T2
      T1 = A1(IJ)
      T2 = A2(IJ)
      IF (A1(I).LE.T1) GO TO 40
      A1(IJ) = A1(I)
      A2(IJ) = A2(I)
      A1(I) = T1
      A2(I) = T2
      T1 = A1(IJ)
      T2 = A2(IJ)
      GOTO 40
   30 A1(L) = A1(K)
      A2(L) = A2(K)
      A1(K) = TT1
      A2(K) = TT2
   40 L=L-1
      IF (A1(L).GT.T1) GO TO 40
      TT1 = A1(L)
      TT2 = A2(L)
   50 K=K+1
```

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

```
      IF (A1(K).LT.T1) GO TO 50
      IF(K.LE.L) GOTO 30
      IF(L-I.LE.J-K) GOTO 60
      IL(M)=I
      IU(M)=L
      I=K
      M=M+1
      GOTO 80
   60 IL(M) = K
      IU(M)=J
      J=L
      M=M+1
      GOTO 80
   70 M=M-1
      IF (M.NE.0) GO TO 75
      IF (SL) RETURN
      ND2 = ND/2
      I1 = II
      I2 = JJ
      DO 110 I=1,ND2
      T1 = A1(I1)
      T2 = A2(I1)
      A1(I1) = A1(I2)
      A2(I1) = A2(I2)
      A1(I2) = T1
      A2(I2) = T2
      I1 = I1 + 1
  110 I2 = I2 - 1
      RETURN
   75 I=IL(M)
      J=IU(M)
   80 IF(J-I.GE.II) GOTO 10
      IF(I.EQ.II) GOTO 5
      I=I-1
   90 I=I+1
      IF(I.EQ.J) GOTO 70
      T1 = A1(I+1)
      T2 = A2(I+1)
      IF (A1(I).LE.T1) GO TO 90
      K=I
  100 A1(K+1) = A1(K)
      A2(K+1) = A2(K)
      K=K-1
      IF (T1.LT.A1(K)) GO TO 100
      A1(K+1) = T1
      A2(K+1) = T2
      GOTO 90
      RETURN
      END
```

REMOVAL OF BOUNDARIES

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

I. <u>NAME</u>

DBOUND (Delete Boundaries)

II. <u>DESCRIPTION</u>

This subroutine modifies each of the boundary pixels in an image to the most frequently occurring number in its 3 by 3 neighborhood. This is useful, for example, in suppressing all the boundary points in a GTM and replacing them with reasonable class labels.

III. <u>CALLING SEQUENCE</u>

CALL DBOUND (NREC, NEL, NEL2, NTAPI, NTAPO, IX, IY)

where

NREC is the number of records in the input image;

NEL is the number of pixels per record;

NEL2 = NEL+2;

NTAPI, NTAPO are the logical unit numbers of input and output sequential data sets;

IX, IY are work arrays to be dimensioned (NEL2,3) and NEL bytes.

All the calling arguments except IX and IY are inputs.

IV. <u>INPUT/OUTPUT</u>

The input and output data sets are in unformatted FORTRAN. The number of records is NREC and the number of pixels per record is NEL. Each pixel is represented by a byte.

V. <u>DESCRIPTION OF SUBROUTINES</u>

The subprograms required by this routine are:

SARN, a sequential access array read routine entry under subroutine DARN; VMOV2, a routine to move a vector in core requiring 300 bytes of storage; and MAJOR, a function (requiring 834 bytes of storage) which gives the most frequently occurring number in a 3 by 3 neighborhood.

VI. PERFORMANCE SPECIFICATIONS

1. STORAGE

This subroutine is 1008 bytes long.

2. EXECUTION TIME

The time required to process an image whose size is 820 lines x 1000 pixels is approximately 36 seconds, and is 378 seconds for a large map of 4000 x 2100 pixels, giving a processing speed on typical maps in excess of 20,000 pixels per second.

3. I/O LOAD

None.

4. RESTRICTIONS

None.

VII. METHOD

This program uses a circular buffer IX with pointers I1, I2, I3 indicating the previous, present and next records under consideration. Initially, I1, I2, and I3 are set at 1, 2, and 3, respectively. After each record is processed, the pointers I1, I2, and I3 are "rolled" upward. The processing of each record consists of checking the eight neighbors of each pixel whose value is zero. The function subprogram MAJOR is employed to determine the number most frequently occurring in the set of eight. (If such a number is not unique, the first encountered number is taken.) Neighboring boundary values are ignored in the count, to prevent MAJOR returning a boundary value when these are most frequent in the neighborhood.

Records 0 and NREC+1 are defined to be identical to records 1 and NREC respectively. Also, pixels 0 and NEL+1 in any record are defined to be the same as pixels 1 and NEL in the same record.

VIII. COMMENTS

None.

IX. TESTS

This program was used to remove the extraneous boundary points from the LACIE class map. Figure 27 shows the selected section of the LACIE class map after the boundary points have been removed.

X. LISTINGS

The computer listings for DBOUND and its subroutines are included next.

```
444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444477
444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444477
444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444444477
244444444444444444444414444444444444444444444444444444444444444444444444444444444444444444444777
211111111111111111111111444444444444444444444444444444444444444444444444444444444444444444444777
211111111111111111111111144444444444444444444444444444444444444444444444444444444444444444444777
211111111111111111111111144444444444444444444444444444444444444444444444444444444444444444444477
211111111111111111111111144444444444444444444444444444444444444444444444444444444444444444444477
211111111111111111111111144444444444444444444444444444444444444444444444444444444444444444444477
211111111111111111111111144444444444444444444444444444444444444444444444444444444444444444444477
211111111111111111111111144444444444444444444444444444444444444444444444444444444444444444444477
211111111111111111111111144444444444444444444444444444444444444444444444444444444444444444444477
211111111111111111111111144444444444444444444444444444444444444444444444444444444444444444444477
211111111111111111111111144444444444444444444444444444444444444444444444444444444444444444444477
211111111111111111111111444444444444444444444444444444444444444444444444444444444444444444444477
211111111111111111111111444444444444444444444444444444444444444444444444444444444444444444444477
211111111111111111111111444444444444444444444444444444444444444444444444444444444444444444442277
211111111111111111111144444444444444444444444444444444444444444444444444444444444422222222222277
211111111111111111111144444444444444444444444444444444444444444444444444444444444422222222222227
211111111111111111111444444444444444444444444444444444444444444444444444444444444422222222222227
211111111111111111114444444444444444444444444444444444444444444444444444444444444422222222222227
211111111111111111144444444444444444444444444444444444444444444444444444444444444422222222222227
211111111111111111444444444444444444444444444444444444444444444444444444444444444422222222222227
211111111111111114444444444444444444444444444444444444444444444444444444444444444422222222222227
211111111111111114444444444444444444444444444444444444444444444444444444444444444422222222222227
211111111111111444444444444444444444444444444444444444444444444444444444444444444422222222222277
211111111111114444444444444444444444444444444444444444444444444444444444444444444422222272227277
211111111111144444444444444444444444444444444444444444444444444444444444444444444422222222222277
211111111111444444444444444444444444444444444444444444444444444444444444444444444422222222222277
211111111114444444444444444444444444444444444444444444444444444444444444444444444422222222222277
211111111114444444444444444444444444444444444444444444444444444444444444444444444422222222222277
211111111144444444444444444444444444444444444444444444444444444444444444444444444422222222222277
211111111144444444444444444444444444444444444444444444444444444444444444444444444422222222222777
511111115554444444444444444444444444444444444444444444444444444555444444444444444555222222222777
555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555777
```

Figure 27. Plot of Class Numbers after Removal of Boundaries

```
      SUBROUTINE DBOUND(NREC,NEL,NEL2,NTAPI,NTAPO,IX,IY)
C
C  THE PURPOSE OF THIS PROGRAM IS TO MODIFY POINTS IN THE IMAGE ON NTAPI
C  WITH NUMBER 0 TO THE NUMBER OCCURRING MOST FREQUENTLY IN A 3 BY 3
C  NEIGHBORHOOD
C
      LOGICAL*1 IX(NEL2,3), IY(NEL)
C
C     INITIALIZE ARRAY IX.
      I1=1
      I2=2
      I3=3
      CALL SARN(NTAPI,IX(2,I1),NEL*2)
      IX(1,I1)=IX(2,I1)
      IX(NEL2,I1)=IX(NEL+1,I1)
      DO 5 J=1,NEL2
    5 IX(J,I2) = IX(J,I1)
C
C     IF(I.LT.NREC) READ (I+1)'ST RECORD INTO IX(*,I3).
C     FOR LAST RECORD, MOVE IX(*,I2) INTO IX(*,I3).
      DO 10 I=1,NREC
      IF (I.LT.NREC) GO TO 8
      DO 6 J=1,NEL2
    6 IX(J,I3) = IX(J,I2)
      GO TO 9
    8 CALL SARN (NTAPI,IX(2,I3),NEL*2)
      IX(1,I3)=IX(2,I3)
      IX(NEL2,I3)=IX(NEL+1,I3)
    9 CONTINUE
C
C     NOW, THE PREVIOUS, CURRENT AND NEXT ROWS ARE IN IX(*,I1),IX(*,I2)
C     AND IX(*,I3) RESPECTIVELY.  MODIFY EACH 0 IN IX(*,I2) TO THE MAJO-
C     RITY CLASS NUMBER IN THE 3 BY 3 NEIGHBORHOOD OF IT.
      DO 20 J=1,NEL
      IY(J)=IX(J+1,I2)
   20 IF(IY(J).EQ.0)IY(J)=MAJOR(IX,NEL2,I1,I2,I3,J+1)
      WRITE(NTAPO) IY
C
C     MODIFY I1,I2,I3 IN PREPARATION FOR THE NEXT RECORD.
      IW=I1
      I1=I2
      I2=I3
   10 I3=IW
      RETURN
      END
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
      FUNCTION MAJOR(IX,NEL,I1,I2,I3,J)
C
C     FIND THE MOST FREQUENTLY OCCURRING NUMBER AMONG THE EIGHT
C     NEIGHBORS OF IX(J,I2).  NOTE THAT 1.LT.J.LT.NEL.
C
      LOGICAL*1 IX(NEL,3)
      DIMENSION LABEL(8),NUMBER(8)
C
      LABEL(1)=IX(J-1,I1)
      NUMBER(1)=1
      N=1
      J2=J-2
C
      DO 30 I=1,3
      IF(I.EQ.1)II=I1
      IF(I.EQ.2)II=I2
      IF(I.EQ.3)II=I3
      KM=1
      IF(I.EQ.1)KM=2
      INC=1
      IF(I.EQ.2)INC=2
C
      DO 10 K=KM,3,INC
      DO 20 L=1,N
      IF(IX(J2+K,II).EQ.LABEL(L))GO TO 40
20    CONTINUE
      N=N+1
      LABEL(N)=IX(J2+K,II)
      NUMBER(N)=1
      GO TO 10
40    NUMBER(L)=NUMBER(L)+1
10    CONTINUE
30    CONTINUE
C
      DO 60 I=1,N
      IF(LABEL(I).EQ.0) NUMBER(I)=0
60    CONTINUE
C
      MAX=0
      DO 50 I=1,N
      IF(NUMBER(I).LE.MAX)GO TO 50
      MAJOR=LABEL(I)
      MAX=NUMBER(I)
50    CONTINUE
      RETURN
      END
      SUBROUTINE DARN(IDEV,IREC,X,N)
C
C     THIS SUBROUTINE READS N BYTES FROM DIRECT ACCESS DEVICE IDEV
C     STARTING AT RECORD IX INTO ARRAY X
C
      LOGICAL*1 X(N)
      READ(IDEV'IREC)X
      RETURN
C
      ENTRY DAWN(IDEV,IREC,X,N)
C
C  THIS ENTRY WRITES N BYTES OF ARRAY X ONTO RECORD IX OF DIRECT ACCESS
C  DEVICE IDEV
      WRITE(IDEV'IREC)X
      RETURN
C
      ENTRY SARN(NTAPI,X,N)
C
C  THIS ENTRY READS N BYTES FROM SEQUENTIAL ACCESS UNIT NTAPI INTO
C  ARRAY X
      READ(NTAPI)X
      RETURN
C
      ENTRY SAWN(NTAPO,X,N)
C
C  THIS ENTRY WRITES N BYTES OF ARRAY X ONTO SEQUENTIAL ACCESS
C  UNIT NTAPI
      WRITE(NTAPO)X
      RETURN
      END
```

REPLACEMENT OF REGION IDENTIFICATION NUMBERS WITH CLASS IDENTIFICATION NUMBERS

I. NAME

RINCIN

II. DESCRIPTION

The output of the REGION program is a data set where each data value is a region number. For comparison with other maps, it is desirable that each data value be a class number. This subroutine replaces each region number in the REGION output with the class number corresponding to that region.

III. CALLING SEQUENCE

Call RINCIN (IW, ICLS, LINE, IPIX, NUM, NREC, NEL, NCLS, NREG, NDEVM, NDEVO, IDEV, and IX)

where

IW, ICLS, LINE, IPIX, and IX are arrays to be dimensioned as indicated in the listings.

NUM = class number.

NREC = number of records in the input image.

NEL = number of pixels per record.

NCLS = number of coordinate-class number values.

NREG = number of regions in the map.

NDEVM and NDEVO are logical unit numbers of input and output sequential data sets.

IDEV = logical unit number of a direct access data set.

IV. INPUT/OUTPUT

The input and output data sets are in unformatted FORTRAN. There are NREC records and NEL pixels per record. Each input pixel is represented by one byte. The table of coordinates and class numbers is input from cards.

V. DESCRIPTION OF SUBROUTINES

No additional subroutines are called.

VI. PERFORMANCE SPECIFICATIONS

1. STORAGE

This subroutine is 1306 bytes long.

2. EXECUTION TIME

The time required to process an image whose size is 820 lines x 1000 pixels per line is approximately 22 seconds on the IBM 360/75 computer.

3. RESTRICTION

None.

VII. METHOD

A simplified flow diagram for the program to change region numbers to class numbers is shown in Figure 28. First, the region map is placed on a direct access device to avoid numerous rewinds when creating a class table. The class table is created by placing a class number in the table at the element corresponding to the region number. Region numbers at the given set of coordinates (at least one set per region) are determined by reading the direct access file at the coordinates. This process is repeated for all coordinate pairs. After the class table has been set up, each record of the image is read into a sequential device. Then each pixel is changed from a region number to the class table as specified by the region number.

VIII. COMMENTS

None

IX. TESTS

This program was used to convert the region numbers of the LACIE GTM to class numbers.

X. LISTINGS

The listings of the main program and associated subroutines are attached at the end of this section.

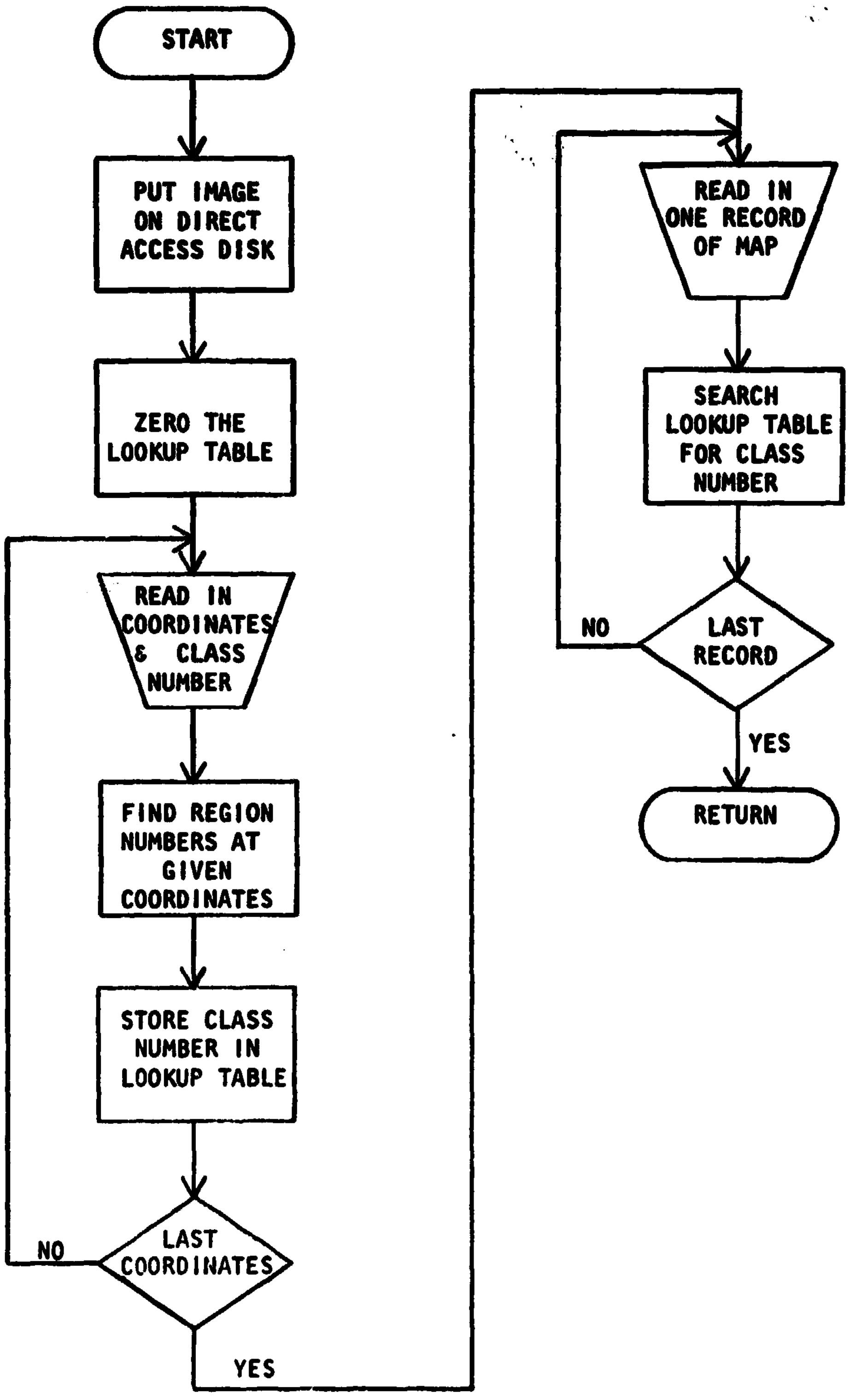

START
PUT IMAGE ON DIRECT ACCESS DISK
ZERO THE LOOKUP TABLE
READ IN COORDINATES & CLASS NUMBER
FIND REGION NUMBERS AT GIVEN COORDINATES
STORE CLASS NUMBER IN LOOKUP TABLE
LAST COORDINATES
NO
YES
READ IN ONE RECORD OF MAP
SEARCH LOOKUP TABLE FOR CLASS NUMBER
LAST RECORD
NO
YES
RETURN

FIGURE 28. PROGRAM TO CHANGE REGION NUMBERS TO CLASS NUMBERS

```
      SUBROUTINE RINCIN(IW,ICLS,LINE,IPIX,NUM,NREC,NEL,NCLS,NREG,
     .   NDEVM,NDEVO,IDEV,IX)
C
C     THIS SUBROUTINE CONVERTS REGION IDENTIFICATION NUMBERS (RIN'S)
C     TO CLASS IDENTIFICATION NUMBERS (CIN'S)
C     DEFINE FILE IDEV (NREC, NEL, L, IV) REQUIRED IN CALLING PROGRAM
C
      DIMENSION LINE(NCLS),IPIX(NCLS),NUM(NCLS)
      INTEGER*2 IW(NEL),ICLS(NREG)
      LOGICAL*1 IX(NEL)
C
C     PUT IMAGE ON DISK
      DO 10 I=1,NREC
      READ(NDEVM) IW
      WRITE(IDEV'I) IW
   10 CONTINUE
C
C     ZERO THE LOOKUP TABLE
      DO 15 I=1,NREG
   15 ICLS(I) = 0
C
C     READ COORDINATE VALUES AND CORRESPONDING CLASS NUMBERS
      READ (5,100) (LINE(I), IPIX(I), NUM(I), I=1,NCLS)
      WRITE (6,200) (LINE(I), IPIX(I), NUM(I), I=1,NCLS)
C
C     STORE CLASS NUMBERS IN LOOKUP TABLE
      DO 30 I=1,NCLS
      READ(IDEV'LINE(I)) IW
      IREG=IW(IPIX(I))
      ICLS(IREG)=NUM(I)
   30 CONTINUE
C
C     REPLACE REGION NUMBERS WITH CLASS NUMBERS
      REWIND NDEVM
      DO 60 I=1,NREC
      READ(NDEVM) IW
      DO 50 J=1,NEL
      IF(IW(J).EQ.0) GO TO 46
      IF(IW(J).EQ.1) GO TO 45
      IW(J)=ICLS(IW(J))
      GO TO 46
   45 IW(J)=0
   46 IX(J)=IW(J)
   50 CONTINUE
      WRITE(NDEVO) IX
   60 CONTINUE
      RETURN
C
  100 FORMAT(5(3I5))
  200 FORMAT(5(10X,3I5))
      END
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

REFERENCES

1. Jayroe, Robert R., Jr., "A Fast Routine for Computing Multidimensional Histograms," NASA TM-78133, October 1977.

2. Jayroe, Robert R., Jr., "Evaluation Criteria for Software Classification Inventories, Accuracies, and Maps," NASA TMX-73347, September 1976.

3. Gupta, J. N., Wintz, P. A., "Multi-Image Modeling", Technical Report TR-EE-74-24, School of Electrical Engineering, Purdue University, West Lafayette, Indiana.

4. Gupta, J. N., Wintz, P. A., "A Boundary Finding Algorithm and Its Applications", IEEE CAS-22, No. 4, April 1975, 351-362.

APPROVAL

EVALUATION OF REGISTRATION, COMPRESSION AND CLASSIFICATION ALGORITHMS – VOLUME II (DOCUMENTATION)

By R. Jayroe, R. Atkinson, L. Callas,
J. Hodges, B. Gaggini, and J. Peterson

The information in this report has been reviewed for technical content Review of any information concerning Department of Defense or nuclear energy activities or programs has been made by the MSFC Security Classification Officer. This report, in its entirety, has been determined to be unclassified.

[illegible]

fr J. T. POWELL
Director, Data Systems Laboratory

✿ U.S. GOVERNMENT PRINTING OFFICE 1979–640 081/560 REGION NO. 4