

A STUDY AND EVALUATION OF IMAGE ANALYSIS TECHNIQUES APPLIED TO REMOTELY SENSED DATA

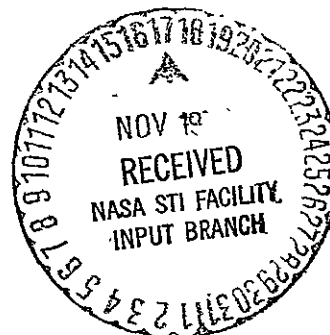
(NASA-CR-150041) A STUDY AND EVALUATION OF
IMAGE ANALYSIS TECHNIQUES APPLIED TO
REMOTELY SENSED DATA Final Report (Computer
Sciences Corp.) 205 p HC A10/ME A01.

N77-10614

Unclas

CSSL 05B G3/43 09513

October 1976



CSC

COMPUTER SCIENCES CORPORATION

COMPUTER SCIENCES CORPORATION

SYSTEMS DIVISION

(205) 883-1770

8300 SOUTH WHITESBURG DRIVE · HUNTSVILLE, ALABAMA

35802

October 28, 1976

National Aeronautics and Space Administration
George C. Marshall Space Flight Center
Huntsville, Alabama 35812

Attention: Dr. R. R. Jayroe, Jr., EF32

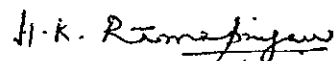
Reference: Contract No. NAS8-32107

Subject: Final Report

In accordance with the requirements of the referenced contract, a copy of the final report is attached herewith. This report covers a study of registration, geometric correction, classification, change detection and data compression as applied to several Landsat scenes covering the Mobile Bay, Alabama test area. The new developments needed were mainly in the areas of geometric correction, preparation of the ground truth map for point-by-point comparisons in assessing classifications and automatic change detection.

The report is in two parts, the first part presenting the descriptions of techniques and experimental results and the second part providing the formal documentation of the programs developed.

Sincerely,



H. K. Ramapriyan
Project Manager

Enclosures

:jg

COMPUTER SCIENCES CORPORATION

8300 South Whitesburg Drive
Huntsville, Alabama 35802

A STUDY AND EVALUATION OF IMAGE ANALYSIS
TECHNIQUES APPLIED TO REMOTELY SENSED DATA

October 1976

Contract No. NAS8-32107

Prepared By:

R. J. Atkinson
B. V. Dasarathy
M. Lybanon
H. K. Ramapriyan

Prepared For:

R. R. Jayroe, Jr., EF32
Information Management and Analysis Branch

NASA, GEORGE C. MARSHALL SPACE FLIGHT CENTER
MARSHALL SPACE FLIGHT CENTER, ALABAMA 35812

A STUDY AND EVALUATION OF IMAGE ANALYSIS TECHNIQUES
APPLIED TO REMOTELY SENSED DATA

By

R.J. Atkinson, B.V. Dasarathy, M. Lybanon, H.K. Ramapriyan

Abstract

Several areas of concern to users of remotely sensed data, are covered briefly in this report. An analysis of phenomena causing nonlinearities in the transformation from Landsat multispectral scanner coordinates to ground coordinates is presented. Experimental results comparing rms errors at ground control points indicate a slight improvement when a nonlinear (8-parameter) transformation is used instead of an affine (6-parameter) transformation. The improvements are expected to be more significant when the coordinates of the GCP's are determined more accurately. Using a preliminary ground truth map of a test site in Alabama covering the Mobile Bay area and six Landsat images of the same scene, several classification methods are assessed. The similarity measures indicate a slight improvement over those with a smaller test site in Tennessee used in an earlier report. However, due to the size of the present data set, registration has been a more difficult problem, and with improved GCP selection, it is expected that the similarity measures will show further increases. A methodology has been developed for automatic change detection using classification/cluster maps.

A sophisticated coding scheme is employed for generation of change depiction maps indicating specific types of changes such as from only the interior points of a given set of classes in map 1 to a set of classes in map 2. Inter- and intra-seasonal data of the Mobile Bay test area have been compared to illustrate the method. A beginning has been made in the study of data compression by applying a Karhunen-Loeve transform technique to a small section of the test data set.

The second part of this report provides a formal documentation of the several programs developed for the analysis and assessments presented here.

TABLE OF CONTENTS

	Page
1. INTRODUCTION	1
2. REGISTRATION AND GEOMETRIC CORRECTION	3
2.1 Introduction	3
2.2 Effects Leading to Geometric Distortions of Images	4
2.2.1 Effect of an Altitude Variation	4
2.2.2 Effect Due to Varying Position Along Scan Line	6
2.3 Discussion	6
2.4 Examples	8
2.5 Geometric Correction	10
2.5.1 Statement of the Problem	10
2.5.2 Determination of Bounds on the Output Image	11
2.5.3 Modification of the Transformation	12
2.5.4 Computation of Input Image Coordinates	12
2.5.5 Computation of Output Pixel Values	13
2.5.6 Experimental Comparison of Transformation	13
3. PROGRAM DESCRIPTIONS	14
3.1 Introduction	14
3.2 Resource Requirements	14
3.3 Supervised Nearest-Neighbor Establishing Histogram Approach (SNEHA)	15
3.3.1 Analysis Process	15
3.4 Parametric Recognition Imparting Trained Identification (PRITI)	16
3.4.1 Analysis Process	16
3.5 Piecewise Linear Classifier	17
3.5.1 Analysis Process	17
3.6 Performance Assessments	17
4. CLASSIFICATION TECHNIQUE ASSESSMENT	19
4.1 Introduction	19
4.2 Description of Data Sets	19
4.3 Preparation of the GTM	20
4.4 Comparisons with the GTM	24
4.4.1 HINDU Classifier	27
4.4.2 Table Look-up (ELLTAB)	30
4.4.3 Linear Sequential	30
4.4.4 Piecewise Linear (Histogram based)	30
4.4.5 Nearest Neighbor (Histogram based)	30
4.4.6 Maximum Likelihood (Histogram based)	30

TABLE OF CONTENTS (continued)

	Page
4.5 Comparisons Between Classification Maps	39
4.6 Chi-Square Tests for a Typical Case	39
4.7 General Remarks	43
4.7.1 Qualifiers	43
4.7.2 Conclusions	43
4.7.3 Suggestions for Further Work	46
5. TEMPORAL CHANGE DETECTION	47
5.1 Introduction	47
5.2 Overview of the System	47
5.3 Description of the Change Detection System	49
5.4 Experimental Results	51
5.5 Discussion of Results and Concluding Remarks	51
6. DATA COMPRESSION	58
6.1 Introduction	58
6.2 The Karhunen-Loeve Transform	58
6.3 A Fast Karhunen-Loeve Transformation	59
6.4 Implementation of the Algorithm	62
6.5 Results	63
REFERENCES	70
PART II - I. INTRODUCTION	72
2. GEOMETRIC CORRECTION	73
3. THINNING OF BOUNDARY IMAGES	92
4. FINDING DISCONTINUITIES IN BOUNDARY DATA	107
5. SMOOTHING BOUNDARY DATA	114
6. IDENTIFICATION OF CONNECTED REGIONS	126
7. DELETION OF BOUNDARY POINTS	159
8. COMPUTATION OF CONTINGENCY MATRICES	164
9. COMPARISON OF SUPERVISED CLASSIFICATION MAPS	177
10. TEMPORAL CHANGE DETECTION AND DISPLAY	184
REFERENCES	197

LIST OF TABLES

		Page
Table 2.1	RMS Errors in Geographic Referencing Fits to Landsat Observations of Mobile Bay Area	9
Table 4.1	Landsat Scenes of Interest	19
Table 4.2	Subsets of Landsat Scenes	21
Table 4.3	Affine Transformation Parameters	21
Table 4.4	Information for Registration w. r. t GTM	22
Table 4.5	Registered Data Sets	23
Table 4.6	GTM in Various Stages of Preparation	25
Table 4.7	Joint Histogram Between GTM and HINDU Cluster Map	29
Table 4.8	Similarity Measures of HINDUCM's w. r. t. GTM	29
Table 4.9	Joint Histogram Between GTM and ELLTAB Classification Map	32
Table 4.10	Joint Histogram Between GTM and Linear Sequential Classification Map	34
Table 4.11	Similarity Measures of LCM's w. r. t. GTM	34
Table 4.12	Joint Histogram Between GTM and Piecewise Linear Classification Map	36
Table 4.13	Similarity Measures of PCM's w. r. t. GTM	36
Table 4.14	Joint Histogram Between GTM and Nearest Neighbor Classification Map	38
Table 4.15	Similarity Measures of SNEHACM's w. r. t. GTM	38
Table 4.16	Joint Histogram Between GTM and Histogram-based Maximum Likelihood Classification Map	41
Table 4.17	Similarity Measures of PRITICM's w. r. t. GTM	41
Table 4.18	Similarity Measures Between CM's for Dec. 73 Data Set	42
Table 4.19	Results of the χ^2 Tests	44
Table 5.1	Temporal Change Detection Results	52
Table 6.1	Statistics of the Input and Output Images	65

LIST OF ILLUSTRATIONS

		Page
Figure 4.1	Mobile Bay GTM	26
Figure 4.2	Mobile Bay HINDUCM (Jan. 1975)	28
Figure 4.3	Mobile Bay ELLTACM (Dec. 1973)	31
Figure 4.4	Mobile Bay LCM (April 1974)	33
Figure 4.5	Mobile Bay PCM (Oct. 1972)	35
Figure 4.6	Mobile Bay SNEHACM (Nov. 1973)	37
Figure 4.7	Mobile Bay PRITICM (June 1974)	40
Figure 5.1	Schematic Representation of Temporal Change Detection Methodology	48
Figure 5.2	Total Changes in all Classes (represented by dark pixels)	53
Figure 5.3	Total Changes in Interior Regions	54
Figure 5.4	Total Changes in Boundary Regions	55
Figure 5.5	Additions to the Urban Class	56
Figure 5.6	Deletions from the Interior of the Forest Class	57
Figure 6.1	Coefficients of Eight Nearest Neighbors, where $\alpha = \frac{p}{1+p^2}$	62
Figure 6.2	Histogram of Input Image	66
Figure 6.3	Histogram of Output Image	67
Figure 6.4	Histogram of Output Image Minus Input Image	68
Figure 6.5	Original and Reconstructed Images of the City of Mobile	69

1. INTRODUCTION

Earth related applications of space technology have been receiving considerable attention and support during the recent years. An important example in this area is the use of satellites such as Landsat and Skylab for remote sensing of the earth's resources. With the regular coverage of earth by Landsats 1 and 2, large quantities of data are collected and transmitted. A major concern for a user of such remotely sensed data is the accuracy of the results in terms of spatial registration and interpretation of multispectral signatures. A natural consequence of the repeated coverage of regions on earth by the Landsats is the ability to detect changes. An additional matter of interest to both the users and the agencies collecting and storing the data is the loss in accuracy that one might suffer if the data were compressed in order to achieve economies in transmission and archiving of the enormous volume of information.

The Image Coding Panel of the National Aeronautics and Space Administration addresses the problem of assessing existing digital computer techniques covering the above aspects of image analysis. Reference [1] is an example of the assessment of classification techniques applicable to remotely sensed multispectral images.

This report, a result of a three months' study* by Computer Sciences Corporation, is a continuation of the effort in [1] and covers a broader spectrum of analysis techniques.

A large region, of approximately 1.4 million Landsat pixels, covering the Mobile Bay area in Alabama was used in the study as a test site. Six Landsat images of the same region were used from Computer Compatible Tapes (CCT's) corresponding to six different passes. This was a much larger data set than that considered in [1] which had only 52,000 pixels. Therefore, the registration of the images to a common frame of reference was more difficult. The problems involved, mainly leading to a nonlinear (rather than linear) coordinate transformation, are discussed in Chapter 2. Also included in Chapter 2 is an approach to the implementation of nonlinear transformations on large data sets. Chapter 3 gives a brief description of the various classification techniques that were applied to the data sets.

The assessment methods employed here follow those in [1]. The classification maps are compared with a preliminary ground truth map (GTM) of the region. Due to the type of GTM available, certain special steps were necessary

* Supported by NASA Contract NAS8-32107.

in converting it into a digital format and preparing it for automatic point-by-point comparison with the classification maps. Chapter 4 presents the details of the preparation of the GTM and the results of the comparisons.

A methodology for automatic change detection is developed in Chapter 5 demonstrating the results of comparing inter- and intra- seasonal pairs of classification maps.

A beginning has been made in the evaluation of data compression techniques. Application of a typical technique (Karhunen-Loeve Transform) to a small section of the Mobile Bay test site is illustrated in Chapter 6.

2. REGISTRATION AND GEOMETRIC CORRECTION

2.1 Introduction

In performing multitemporal classifications, change detection, and assessment of classification accuracies relative to the ground truth by a point-by-point comparison with a ground truth map (GTM), it is necessary to register the various scenes under consideration relative to a common basis. It is useful to employ the UTM (Universal Transverse Mercator) System as the basis relative to which all images are corrected. The GTM is generally not expected to have any distortions other than a slight rotational error due to misorientation while digitizing. However, significant distortions are present in Landsat imagery and when several views of a given ground scene are to be handled it is possible that the distortions are different for different views.

Experiments have shown that for small subsets of a Landsat frame (say, less than 500 x 500 pixels), an affine transformation of the form

$$\begin{bmatrix} R \\ C \end{bmatrix} = A \begin{bmatrix} E \\ N \end{bmatrix} + \begin{bmatrix} R_0 \\ C_0 \end{bmatrix}$$

where (R, C) and (E, N) are the pixel and UTM coordinates, respectively, provides an adequate model for the distortions and results in errors of less than one pixel. However, for larger subsets, such a transformation produces larger errors. Therefore, either several affine transformations with image segmentation or a single transformation with additional terms should be used.

This chapter provides an analysis of the causes of the geometric distortion, showing why the linear model is not adequate. Also, experimental results are shown to illustrate the improvements in rms errors at a given set of Ground Control Points (GCP) when EN terms are included in the transformation. It is found, however, that the gain in accuracy is not sufficiently significant to justify application of the more complicated transformation in the present work. With greater accuracy in the determination of the GCP coordinates, the differences in rms errors between the linear and nonlinear models could be more pronounced.

The results presented in the subsequent chapters of this report are based on affine transformations applied to all the data sets. But, a system of programs has been developed for applying the nonlinear transformation so that when the GCP's are identified more accurately, the results can be updated with the new geometric corrections. The philosophy of these programs is slightly different from that of the affine transformation routines and is described briefly in Section 2.5.

2.2 Effects Leading to Geometric Distortions of Images

2.2.1 Effect of an Altitude Variation

Some useful equations relating image coordinates to ground locations are given in Ref. 1. The coordinates (x, y) of a point in the image are given by

$$\begin{aligned}x &= (f \tan \eta) \sin \psi \\y &= (f \tan \eta) \cos \psi\end{aligned}\quad (1)$$

where y is along the (image of the) heading line, x is in the perpendicular direction, and the origin is at the image center (the image of the subpoint). (Eq. (1) actually defines the instantaneous field of view of the sensor. Effects such as the skew produced as a result of the finite time required for the Landsat multispectral scanner to form an image are not included.) The other quantities are defined as follows: f is a scale factor, η is the nadir angle subtended at the satellite by the great circle arc connecting the subpoint and the observed point and ψ is the azimuth of the observed point measured from the heading line (vertex at the subpoint). A spherical Earth is assumed, and the boresight direction of the scanner is assumed to be straight downward.

The nadir angle η is given by

$$\tan \eta = \frac{R \sin \delta}{R (1 - \cos \delta) + H} \quad (2)$$

where R is the Earth's radius, H is the orbital altitude, and δ is the geocentric angle subtended by the great circle arc connecting the subpoint and the observed point. Then the equation for position along a scan line, measured in the image, is

$$x = \frac{fR \sin \delta}{R (1 - \cos \delta) + H} \quad (3)$$

where $R\delta$ measures the distance on the ground from the subpoint to a point on the scan line. Here $x=0$ gives the center pixel of the line.

The dependence of η upon time is a property of the scanner mechanism. The question of interest here is: For a given η (therefore a given x), how does (angular) position on the ground vary with changes in altitude H ? Differentiation of Eq. (3) produces

$$\frac{d\delta}{dH} = \frac{\sin \delta}{R (\cos \delta - 1) + H \cos \delta} \quad (4)$$

Let $u=R\delta$ denote a distance on the ground. Then the shift in ground position corresponding to a given pixel x , due to a variation in satellite altitude H , is

$$\frac{du}{dH} = \frac{R \sin (u/R)}{R [\cos (u/R) - 1] + H \cos (u/R)} \quad (5)$$

From Eq. (5), for a 10 km change in altitude (a reasonable figure) the ground position corresponding to the end pixel on a line shifts by 1.02 km, the equivalent of 17.9 pixels. However, $u/R=0.0145$ radians for this case (and smaller for other points), so small-angle approximations for the trigonometric functions are valid. If terms no higher than the first degree in u/R are retained, Eq. (5) reduces to

$$\frac{du}{dH} = \frac{u}{H} \quad (6)$$

Therefore, although this effect may be large it is approximately linear. So the linear transformation of Ref. (1) should account for it.

For the dimension normal to scan lines, the analysis is different. The Landsat scanner sweeps out scan lines in groups of six simultaneously. Within each group of six scan lines the aspect angle effect prevails. However the ground position of each successive group of scan lines depends on the along-track velocity and the constant time interval. So for points more than a few scan lines apart in the along-track direction the latter effect predominates.

From celestial mechanics it is known that (neglecting non-central gravitational terms and comparable effects) the orbital period is proportional to the $3/2$ power of the semimajor axis. For circular orbits, the velocity is constant and inversely proportional to the period. (Assuming a spherical earth, this is also true of the ground track velocity, which is of interest here.) So the average along-track distance v on the ground between scan lines is

$$v = C a^{-3/2} \quad (7)$$

Since the time interval between scan lines (1/6 the time between scans of successive groups of six lines) is constant. In Eq. (7) C is a constant (a combination of several factors) and a is the orbit's semimajor axis - or radius, since the orbit is assumed circular. Therefore the fractional change in v due to a change in altitude is

$$\frac{dv}{v} = \frac{-3/2 da}{a} = \frac{-3/2 dH}{a} \quad (8)$$

For a scan line increment nominally encompassing the same interval along track as the previous cross-track example, the shift due to a 10-km altitude variation is 190 m, the equivalent of 2.41 scan lines. Therefore an altitude change produces a smaller effect along-track than cross-track. Also, again the effect is a linear scale change.

2.2.2 Effect Due to Varying Position Along Scan Line

The nominal Landsat orbit has a ground track that repeats identically every eighteen days. However, because of small orbit variations the actual ground track may shift laterally, for several passes over the same region. Therefore a landmark that is directly beneath the satellite's path during one pass may be off to the side during another. Because of this situation, it is of interest to consider the variation in the mapping from the ground to the image plane as a function of position along a scan line.

To consider this effect, we return to Eq. (3). The quantity of interest is $dx/du = (1/R) dx/d\delta$. From Eq. (3), it is given by

$$\frac{dx}{du} = f \left\{ \frac{H - 2(R+H) \sin^2(u/2R)}{[2R \sin^2(u/2R) + H]^2} \right\} \quad (9)$$

For points within Landsat images the trigonometric terms are much smaller than the others. So, to a good approximation,

$$\frac{dx}{du} = \frac{f}{H} \quad (10a)$$

$$x = (f/H) u = (\text{constant}) u \quad (10b)$$

In this approximation, the mapping from u to x is linear; projective effects do not cause any distortion. The error in this approximation is less than 1 percent; small, but not negligible.

The above analysis should also apply, at least approximately, to a sensor boresight (pointing) error due to a nonzero satellite roll attitude. Therefore, the nonlinearity may be increased by some factor greater than unity.

2.3 Discussion

The distortion-producing effects on Landsat imagery of some physical processes that can be analyzed easily have been studied. It was found that the effects of orbital altitude variations and the projection into the image plane from various points along a scan line can be accounted for very nearly by a linear transformation, as was concluded in Ref. 1.

However, it can be seen that the approximations are not perfect. First the effects of altitude variation will be considered. It should be noted that Eq. (6), giving du/dH , is a small-angle approximation to the more accurate expression, Eq. (5). For the numerical example considered, the discrepancy between the two formulas amounts to slightly less than 1m - about 1-1/2 percent of a pixel. Clearly the small-angle approximation is valid in this case. Another-more serious-approximation has been made, however. The right-hand side of Eq. (6) (as well as Eq. (5)) depends on H . So the Taylor series giving the change in u induced by a change in H contains nonzero terms beyond first order. The amount of the nonlinearity is estimated by the next term, which has a magnitude of roughly 0.2 pixel for the case used as an example. This is small, but possibly of borderline significance if one is attempting to register images to subpixel accuracy.

When the same analysis is applied to dv/dH , given by Eq. (8), the magnitude of the nonlinearity is smaller. For the same numerical example considered previously, the nonlinear correction amounts to a fraction of a meter. So it may reasonably be treated as negligible.

The above analysis has considered the distortions in components (u,v) of locations on the Earth's surface mapped into given positions in the image, where u is in the scanning direction and v is parallel to the motion of the subsatellite point. Standard geographic coordinates used, such as the E and N of the UTM system, are not parallel to u and v . So both E and N contain contributions from u , which the above analysis shows may lead to small but possibly significant nonlinearities in the mapping from the ground into the image.

The other effect considered, the (projective) effect due to varying position along a scan line, does not involve changes in altitude. So Eq. (10a) actually describes a linear effect. But this equation is a small-angle approximation to the more accurate Eq. (9). Differences between the two equations give the sought-for nonlinearity. As has been pointed out, Eq. (10a) is a very good approximation, in error by only a fraction of a percent at the ends of a scan line. However, this translates into an equivalent ground distance of about 200 m at these points, a significant nonlinearity equivalent to several pixels. And, as discussed above, the effect of this nonlinearity is felt in both E and N.

It has been seen that, while the linear coordinate transformation is a very good approximation, the effects considered lead to some possibly significant nonlinearities. And only a few effects have been analyzed. It is possible that other causes may lead to even greater nonlinearities. Therefore it appears that a nonlinear transformation might be used with profit.

The simplest nonlinear transformation is obtained by adding terms proportional to the product (EN) to both the x and y equations of the linear transformation derived in Ref. 1. This simplicity is consistent with the relatively small magnitude of the expected nonlinearities. Also, Rifman et al reported success in using such a transformation with Landsat data. [2]

2.4 Examples

Least-squares fits were made to several sets of data resulting from Landsat passes over the Mobile Bay, Alabama area, using both a linear coordinate transformation and the nonlinear transformation described above. The linear transformation involved six parameters: the four elements of a matrix accounting for rotation, skew and scale factors, and the two components of an origin shift vector. The nonlinear transformation added two more parameters: coefficients of cross-product (EN) terms. The program used made several fits for each data set, successively discarding fit points possessing fit errors too large to pass a test.

The same 36 ground control points (GCPs) were used for each data set. Table 1 shows a sample of the results, for three data sets. RMS errors are given for fits involving all 36 GCPs and fits using 18 points (in some cases, linear interpolation was used to obtain the latter). Shown are errors computed at fit points only, and for all the points.

The results show that the nonlinear transformation always produced smaller errors at fit points. (They can never be larger, since the nonlinear transformation was produced by adding terms to the linear transformation.) The situation is less clear-cut when errors computed for all 36 GCPs are compared for the 18-point fits. In order to clarify the situation, several steps may be useful:

- (1) Recheck the coordinates of all the GCPs.
- (2) Obtain better estimates of GCP image coordinates by first making enlargements (using digital resampling techniques) of regions surrounding the points.
- (3) Experiment with more data sets.
- (4) Experiment with other nonlinear transformations.

Because all of these will involve significant effort and time, an estimate of the expected improvement over present results should be made and balanced against the expense required to achieve that improvement.

Table 2-1. RMS Errors in Geographic Referencing Fits to Landsat Observations of Mobile Bay Area

Date of Observation	Number of GCPs*	6-Parameter Linear Transformation		8-Parameter Nonlinear Transformation	
		Error at Fit Points	Error at All Points	Error at Fit Points	Error at All Points
10/17/72	36	1.7912	1.7912	1.7831	1.7831
	18	0.80430	1.9095	0.69475	2.0202
11/17/73	36	1.3308	1.3308	1.2320	1.2320
	18	0.73284	1.4361	0.57837	1.4444
12/5/73	36	1.3264	1.3264	1.2970	1.2970
	18	0.58774	1.4614	0.53989	1.4512

Rms errors are given in pixel units.

*GCP: ground control point.

2.5 Geometric Correction

In this section we shall discuss a method of applying a nonlinear geometric transformation to a large image.

If the main memory of the computer is large enough to contain all the lines of the input image required to compute any one output line, then the implementation of the geometric correction is a simple matter. But, this is not the case in general. A method of handling large images using segmentation of the input image has been described in detail in [3] for the case of the affine transformation. This technique is applicable to general nonlinear transformations, but the computations of the number of segments required and the portions of each output record that can be computed with the data from an input segment are somewhat cumbersome. Therefore, a slightly different approach is employed which will be described in the sequel.

2.5.1 Statement of the Problem

We are given a transformation of the form

$$\begin{bmatrix} X \\ Y \end{bmatrix} = A \begin{bmatrix} U \\ V \end{bmatrix} + \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix} + \begin{bmatrix} \alpha \\ \beta \end{bmatrix} UV \quad (11)$$

where X, Y are the coordinates of a point in the input image reference system and U, V are those of the same point in the output coordinate system. The input image function (say classification numbers or reflected intensity values in a given spectral band) is defined for all integral values of X, Y satisfying

$$1 \leq X \leq \bar{X}; 1 \leq Y \leq \bar{Y} \quad (12)$$

The output image function should be stored as $(\bar{U} - \underline{U} + 1)$ records (lines) with $(\bar{V} - \underline{V} + 1)$ samples in each record, the values being computed for all integral values of U, V satisfying

$$\underline{U} \leq U \leq \bar{U}; \underline{V} \leq V \leq \bar{V} \quad (13)$$

where $\underline{U}, \bar{U}, \underline{V}, \bar{V}$ are the integral bounds on U and V such that (11) and (12) are satisfied.

The following steps are involved in computing the output image function.

1. Determination of \underline{U} , \bar{U} , \underline{V} , \bar{V} .
2. Deciding whether it is convenient to implement the given transformation or a slight modification thereof.
3. Computing the input image coordinates for each output point in a record.
4. Finding the nearest integer values to the input coordinates and obtaining the output image function by a suitable interpolation method.

2.5.2 Determination of Bounds on the Output Image

The values of \underline{U} , \bar{U} , \underline{V} , \bar{V} should be found such that, for all (U, V) satisfying (13), the solution (X, Y) to equation (11) satisfies (12).

If $\alpha = \beta = 0$ the values of \underline{U} , \bar{U} , \underline{V} , \bar{V} are simply found by using

$$\begin{bmatrix} U \\ V \end{bmatrix} = A^{-1} \begin{bmatrix} X - X_0 \\ Y - Y_0 \end{bmatrix} \quad (14)$$

and finding (U, V) corresponding to the four vertices of the rectangle defined by (12).

However, equation (11) results in quadratic equations for U and V in terms of X and Y . The equation for U is:

$$(a_{11}\beta - a_{21}\alpha)U^2 + [a_{11}a_{22} - a_{12}a_{21} + \alpha(Y - Y_0) - \beta(X - X_0)]U + [a_{12}(Y - Y_0) - a_{22}(X - X_0)] = 0 \quad (15)$$

This yields two solutions for U .

A choice is made of the sign to be used in the formula for the solution of the quadratic equation (15) based on the absolute value of the solutions for the case $(X, Y) = (1, 1)$. The sign yielding the smaller absolute value is chosen for convenience and is used henceforth. Given a U , the solution for V is unique.

Due to the nonlinear nature of the transformation, the four vertices of the rectangle defined by (12) do not necessarily yield the extremal values of U and V . However, an examination of the quadratic curves represented by the equation (11) in the (U, V) domain for various values of (X, Y) reveals that the sides of the rectangle defined by (12) should map into curves bounding the output image.

Therefore, \underline{U} , \bar{U} , \underline{V} , \bar{V} are obtained by finding the extremal values of U and V over the sets

$$\begin{aligned}
 X = 1; \quad 1 \leq Y \leq \bar{Y} \\
 X = \bar{X}; \quad 1 \leq Y \leq \bar{Y} \\
 1 \leq X \leq \bar{X}; Y = 1 \\
 1 \leq X \leq \bar{X}; Y = \bar{Y}
 \end{aligned} \tag{16}$$

2.5.3 Modification of the Transformation

It is sometimes convenient to implement a modified transformation and generate an image which is equivalent to the desired output image from the point of view of resampling (that is, integer coordinates in the generated output image correspond to integer coordinates in the image desired). The permissible modifications are interchanges of the U and V axes and/or changes in the signs of U and/or V.

The maximum number of input records required to compute an output record is given by

$$R_1 = \text{Max} \left[\left| a_{12} + \alpha \underline{U} \right| (\bar{V} - \underline{V}), \left| a_{12} + \alpha \bar{U} \right| (\bar{V} - \underline{V}) \right]$$

if the given transformation is implemented. If U and V axes are interchanged, this changes to

$$R_2 = \text{Max} \left[\left| a_{11} + \alpha \underline{V} \right| (\bar{U} - \underline{U}), \left| a_{11} + \alpha \bar{V} \right| (\bar{U} - \underline{U}) \right]$$

If $R_1 \leq R_2$, then U and V need not be interchanged. Otherwise, the columns of A, $(\underline{U}, \underline{V})$ and (\bar{U}, \bar{V}) are interchanged.

Also, it is desirable to have the output record and sample numbers increase with respect to those in the input image. Since α and β are generally much smaller than the elements of A, $a_{11} \geq 0$ and $a_{22} \geq 0$ imply that the above requirement is met. If $a_{11} < 0$, then the first column of A, α , β , \underline{U} , \bar{U} are negated and \underline{U} , \bar{U} are interchanged. If $a_{22} < 0$, then the second column of A, α , β , \underline{V} , \bar{V} are negated and \underline{V} , \bar{V} are interchanged.

2.5.4 Computation of Input Image Coordinates

For each line in the output image (that is, for a given U) there are $\bar{V} - \underline{V} + 1$ output pixel values to be computed. The (X, Y) coordinates for each of these points are computed and stored in two arrays. Also, some of the computation is saved by checking whether the computed value of X is between 1 and \bar{X} . If not, the Y value is set to 0 without any computation.

2.5.5 Computation of Output Pixel Values

Suppose $\{X_i | i=1, 2, \dots, N\}$ and $\{Y_i | i=1, 2, \dots, N\}$ are the coordinate arrays with (X_i, Y_i) corresponding to $(U, i=\underline{V}-1)$ and $N=\overline{V}-\underline{V}+1$. Also, whenever $1 > X_i$ or $X_i > \overline{X}$, $Y_i=0$. Then, the input records needed to compute the current record of output range from R_1 through R_2 where

$$R_1 = \text{Min } \{X_i | 1 \leq i \leq N; Y_i \neq 0\}$$

$$R_2 = \text{Max } \{X_i | 1 \leq i \leq N; Y_i \neq 0\}.$$

Now, if the storage available for the input image can contain $R_2 - R_1 + 1$ records, it is possible to compute the entire output record by reading the appropriate input records into storage. However, if only a part of the records needed can be held in core at a time, the data handling becomes more complicated. The method employed here is iterative.

Assume that records S_1 through S_2 are available in core at a given stage and R_1 through R_2 are needed for computing the remaining part of the current record. Then, all the output pixels that can be computed using S_1 through S_2 are computed, a code array indicating the computed pixels is updated, as many as possible of the new input records in the range R_1 through R_2 are read into core and S_1, S_2 are modified to reflect the present state of the storage. This procedure is repeated until all the output pixels of the current record have been computed.

The above method works employing a circularly addressed input buffer to avoid unnecessary movement of data within the core storage and assumes that the input image is available on a direct access device so that the updating of storage can be performed using both forward and backward reading of input records.

2.5.6 Experimental Comparison of Transformation

An experiment was performed using both the affine and the above 8-parameter transformation on a six class classification map of the Mobile Bay, Alabama region obtained by using a Linear Sequential classifier. The transformations used here were both obtained such that the mean squared errors over the same set of ground control points were minimized. The resulting output images were overlaid and their joint histogram was found. This is shown in Figure 2.1. The "similarity measure" between the two images, defined as in [4] is seen to be 84.6 percent. Also, as expected, most of the deviations occur at boundaries between classes. In fact the differences occurring at locations interior to the classes in the output of the affine transformation amount only to 2.79 percent.

3. PROGRAM DESCRIPTIONS

3.1 Introduction

In this study, three classification techniques in addition to those described previously [1] were employed. These additional techniques are supervised, in that a set of training samples, whose classification is known, is input to the system. These samples are then used to evaluate the required parameters of the classification algorithm, such as the Gaussian parameters of the distributions or the coefficients of linear discriminant functions, for use in a table look-up classification scheme.

The next step is the table formulation phase, which consists of creating the table or array of class labels corresponding to the set of all the independent feature vectors expected in the data environment. In the approach considered here, this effort is limited to generating the cluster or class labels corresponding to a smaller set of prototypes only instead of all possible feature measurements. The prototypes are defined here as the centroids of the contents of all the occupied cells resulting from a multidimensional histogram analysis. When being deployed with the HINDU [2] system, there is no additional effort needed for identifying these prototypes as these are already available being the output of the histogram generator.

The classification, or table lookup, phase requires the use of the incoming measurement vector to locate the proper element of the table, which contains the class number to which the pixel is assigned.

In the approach developed here for use with the HINDU system, each input sample has already been flagged during the histogram analysis with an index number which identifies its address on an address array, this address denoting the cell corresponding to the input sample. This information being available on external storage is read in (instead of the feature vectors themselves) during the table lookup phase and the entries in this array are replaced by the corresponding entry of the label array or table. Hence, no additional searching of the array is involved in this phase (equivalent effort having been expended earlier in the learning phase).

3.2 Resource Requirements

The computer resources employed by these classifiers are similar to those given previously for the HINDU system, viz. a core memory requirement of 150K bytes, two tape drives for input and output, and external storage for the histogram cell addresses. An additional input requirement is a set of labeled training samples.

3.3 Supervised Nearest-Neighbor Establishing Histogram Approach (SNEHA) [3]

3.3.1 Analysis Process

In this method, each class is represented not by a single description (e. g. its centroid), but by a set of descriptors. Here, the centroids of all the histogram cells within each class are identified and designated as the descriptors of the corresponding clusters. The prototypes are then classified on the basis of the label of the nearest descriptor (cell centroid).

The steps in the analysis are as follows:

- (i) Training Set Definition: This phase is common to all supervised classification techniques and accordingly details of these efforts are omitted here, except to state that training samples corresponding to all of the pattern classes expected in the data environment are selected through ground truth and/or photo-interpretation techniques. These samples along with their labels form one of the major inputs to the ensuing phases.
- (ii) Prototypes Identification: Through a multidimensional histogram analysis of the entire data set, the centroids of all the occupied histogram cells are extracted. These centroids then represent a set of prototypes of the given unclassified data set. These prototypes are treated as a pseudo sample set to be classified according to the nearest neighbor rule. Further, each input sample is also identified during this process by the relative address of the multidimensional histogram cell occupied by the sample. This information is stored for all samples externally for later retrieval. Here, the computational effort is a function of several variables: The dimensionality and size of the data set, the grid size of the histogram. While the first two are dictated by the data set, the latter is an option available to the user. This has been discussed at length in the HINDU System wherein a similar histogram analysis is performed prior to unsupervised learning. The grid size is therefore chosen in accordance with the guidelines and empirical relationships developed for the HINDU system. It is to be noted that in general, the computational effort increases inversely with grid size and, hence, a trade off study between computational effort and accuracy may be necessary.
- (iii) Label Table Formulation: The training sample set along with its labels forms the data base for the nearest neighbor classifier. The prototypes, i. e., the centroids of all the occupied histogram cells are classified on the basis of nearness to one or the other of

these training samples, and the corresponding labels are stored in the form of a table or array of labels to be used in the table lookup phase. The computation involved in this phase is proportional to the size of the training sample set as well as the size of the prototype set, but is independent of the size of the total data set to be labeled. This independence is the key to the success of this approach in the processing of large data sets.

- (iv) Table Lookup and Labeling: In this phase, the relative address of each input sample is retrieved from the external memory (wherein it was stored during the histogram analysis process) and the corresponding entry in the label table is looked up to identify the label of the input sample. Repetition of this operation for the entire data set leads to the output label set corresponding to the input unclassified sample set.

3.4 Parametric Recognition Imparting Trained Identification (PRITI) [4]

3.4.1 Analysis Process

The classical maximum likelihood method is used to classify the centroids of the occupied histogram cells. The steps in the analysis are as follows:

- (i) Training Set Definition: This phase, being common to all supervised classification schemes, is not discussed here at length. The approach is, of course, based on the availability of well defined ground truth (i.e., labeled training sample set) and a knowledge of the probabilistic description of the pattern classes (i.e., distributions underlying the data). These form the input to the next phase.
- (ii) Parameters Estimation: This is the so-called learning phase, wherein the Gaussian parameters of the distributions underlying the different pattern classes are estimated using the labeled training sample data.
- (iii) Prototype Identification: The input data set consisting of all the unclassified samples is processed through a multidimensional histogram analysis package to identify the centroids of all the occupied histogram cells. These centroids are then considered as prototypes of the incoming data set to be classified by the classical parametric recognition method employing the maximum likelihood classifier. During this histogram generation, the relative address of the histogram cell corresponding to each input sample is developed and stored externally for future retrieval. The computations involved in this phase are proportional to the

size of the data set and represent the major part of the total computational expense. However, the histogram grid size has an equally significant effect on this computational load. This size is presently dictated by certain semi-empirical considerations developed earlier in the HINDU system of unsupervised learning.

- (iv) Table Formulation: The prototypes of the data set as discerned by histogram analysis are classified here using the classical maximum likelihood classifier with the estimated parameter values. The labels corresponding to the prototypes are stored as a table or array of labels for the ensuing table lookup phase. The computations involved in this phase are proportional to the number of prototypes (and hence the grid size and spread in the sample set), square of the dimensionality of the data, and the number of pattern classes.
- (v) Table Lookup and Labeling: Here the relative address of each sample in the histogram space is recalled from external memory (where it was stored earlier in the histogram generation phase) and used to lookup the table entry to identify the label of the input sample. This process of table lookup is repeated for all the input unclassified samples to derive the output label set.

3.5 Piecewise Linear Classifier

3.5.1 Analysis Process

The occupied cells are classified on the basis of their centroid positions by a set of discriminant hyperplanes. The parameters of the discriminants are determined by the set of training samples. Otherwise, the procedure is identical to that described in the preceding section regarding parametric recognition. The positions of the discriminant hyperplanes are computed by the Discriminant Hyperplane Abstracting Residuals Minimization Algorithm (DHARMA) [5] which has been described in conjunction with the HINDU system. Consequently, the analysis process is not repeated in detail here.

3.6 Performance Assessments

The CPU time required is divided between the histogram analysis time and the table creation and lookup time. Typically, processing a four-dimensional data set consisting of a quarter million pixels required 84 seconds for identifying the significant clusters in terms of their centroids.

The CPU time required for creating the table containing the class numbers of the prototype cells is on the order of a few seconds, with a maximum of

approximately three seconds for the piecewise linear method. This is due to the longer time required to determine the linear discriminant functions, when compared to the other methods.

The table lookup phase is very rapid. The computational demands of this phase, in terms of CPU time, are only nominal with most of the effort being I/O operation (which is proportional to the number of records or scan lines of data being processed). Even compared to other table lookup approaches, the savings achieved here is significant as the number of prototypes which are classified by the classifier is still much smaller than the number of nonidentical feature vectors one comes across in a large Landsat scene. Approximately 1 1/2 seconds of CPU time is expended in looking up the table to identify the class labels of the quarter million pixels.

4. CLASSIFICATION TECHNIQUE ASSESSMENT

4.1 Introduction

The approach used in this report for assessing the classification methods closely parallels that of [1]. The data sets used here are much larger in size, with approximately 1.4×10^6 pixels compared to 51000 in [1]. The details of the data sets are given in the next section.

The ground truth map (GTM), supplied by the U. S. Geological Survey* is a second level classification map in which each region is denoted by a two digit number. For the purposes of the present work this map was converted into digital format with only the first level classifications. The details of rendering the GTM to a digital form are presented in Section 4.3.

Various tests have been applied to the classification maps described in Chapter 3. The similarity measures relative to the GTM, typical contingency matrices, χ^2 - tests and inventory comparisons are shown in Section 4.4.

The description of the data sets given below include tape numbers containing the data. These refer to the IBM 360 tape library in Building 4708, Marshall Space Flight Center, Huntsville, Alabama, and are provided only for the sake of completeness of the record.

4.2 Description of Data Sets

The data sets used here were obtained from the computer compatible tapes of the following six Landsat Multispectral Scanner (MSS) images, each with four bands of information:

Table 4.1. Landsat Scenes of Interest

Tape Number	ID	Date
A0769	1086-15562	Oct. 17, 1972
A1469	1482-15541	Nov. 17, 1973
A1459	1500-15535	Dec. 5, 1973
A0691	1626-15510	April 10, 1974
A0484	1698-15490	June 21, 1974
A1310	1896-15420	Jan. 5, 1975

*This map was preliminary and had a disclaimer indicating that it was not ready for public release.

The geographic region covered by the data sets is the Mobile Bay area in Southern Alabama.

From each of the Landsat images, a 1200 by 1200 image was extracted to cover this region. Also, the "synthetic" pixels (SP) were removed after the extraction resulting in slight reduction in the number of pixels per line of the images. The coordinates of the top left corners of the data extracted relative to the corresponding Landsat images (before removal of SP's) and the image sizes (after removal of SP's) are shown in Table 4.2.

These data sets were classified using various classification methods. Both the data sets and the classification maps were geometrically corrected to the UTM coordinate system using the affine transformations defined by the parameters shown in Table 4.3, and using a scale factor of 20 pixels per kilometer. Also, to register the geometrically corrected images with respect to the GTM prepared as described in Section 4.3, it was necessary to extract subsets of the images starting at the locations shown in Table 4.4. Finally, the registered images were all 1624 lines by 866 pixels in size.

The tape numbers for the geometrically corrected data and the classification maps (GC) and the corresponding 1624 x 866 data sets overlaying the GTM (OV) are shown in Table 4.5.

4.3 Preparation of the GTM

The goal in preparing the GTM is to generate a digital data set consisting of the class numbers at each point of a rectangular array matching the UTM coordinates with 20 pixels/km (That is, each sample representing a 50 x 50 m² area). Table 4.6 shows the various stages involved in the preparation of the GTM. In the following description GTM_i will be used to denote the ith stage GTM.

A preliminary map, GTM₀, was supplied by J. R. Anderson, U. S. Geological Survey (with a disclaimer indicating that it was not ready for public release). It consisted of interclass boundaries with the class identification numbers (CIN) written inside the respective regions. Being a level II classification map it was very detailed. Each CIN was a two digit number representing the level I land use class and the level II subclass.

The portion of interest in GTM₀ was photographically copied and the annotations were masked out manually. The resulting image was photographically reduced to 4" x 3" (GTM₁) for digitization on a microdensitometer. The image was digitized at a scanning interval of 25 microns to ensure that the boundaries were recorded properly. The part of interest in the map, designated as GTM₂, has 4000 x 2100 pixels.

Table 4.2. Subsets of Landsat Scenes

Date	Initial Row	No. of Rows	Initial Column	No. of Columns	Feature Vector Tape Numbers	
					Bytes	Floating Point
Oct. 17, 72	156	1200	551	1194	A0923	A0198
Nov. 17, 73	7	1200	665	1193	A1048	A0787
Dec. 5, 73	62	1200	751	1194	A0052	A0454
Apr. 10, 74	84	1200	788	1194	A1119	A1235
Jun. 21, 74	148	1200	900	1193	A0872	A0705
Jan. 5, 75	7	1200	836	1193	A0906	A1347

Table 4.3. Affine Transformation Parameters

Date	No. of GCP's	PARAMETERS						RMS Error
		a_{11}	a_{12}	a_{21}	a_{22}	R_0	C_0	
Oct. 17, 72	18	-2.4037	-12.2023	16.6449	-4.2916	42758.3	8863.9	0.80
Nov. 17, 73	20	-2.3846	-12.2867	16.6352	-4.2836	43098.4	8838.8	0.74
Dec. 5, 73	25	-2.3416	-12.3164	16.6034	-4.2711	43192.2	8798.4	0.88
Apr. 10, 74	17	-2.3816	-12.1802	16.6193	-4.1945	42790.4	8541.5	0.83
Jun. 21, 74	24	-2.4133	-12.2759	16.8070	-4.4704	43138.9	9376.3	0.79
Jan. 5, 75	24	-2.3975	-12.3053	16.5819	-4.2947	43199.3	8939.1	0.78

Table 4.4. Information for Registration w. r. t. GTM

Date	Coordinates Matching (1, 1) on GTM*	
	Row	Column
Oct. 17, 72	248	605
Nov. 17, 73	336	577
Dec. 5, 73	345	564
April 10, 74	430	553
June 21, 74	431	514
Jan. 5, 75	396	615

*The GTM here refers to GTM10 described in Section 4.3 (Tape Number A0944)

Table 4.5. Registered Data Sets

Date	Tape Numbers												
	Raw Data	HINDUCM		SNEHACM		PRITICM		LCM		PCM		ETCM	
	GC	GC	OV	GC	OV	GC	OV	GC	OV	GC	OV	GC	OV
Oct. 17, 72	A0620	A0869	A0969	A0557	A0912	A0575	A0147	A0528	A0517	A0390	A0413	-	-
Nov. 17, 73	A0671	A0784	A1209	A0335	A0492	A0894	A1437	A0219	A0202	A0085	A1498	-	-
Dec. 5, 73	A0063	A0286	A0200	A0676	A0633	A0820	A0822	A0798	A0790	A0865	A0831	A1152	A1153
Apr. 10, 74	A1325	A0966	A0197	A0259	A0495	A0209	A0438	A1434	A1423	A1222	A1352	-	-
June 21, 74	A1287	A0868	A0846	A0626	A0374	A0440	A0331	A1457	A1464	A0573	A0340	-	-
Jan. 5, 75	A0828	A0109	A0016	A0279	A0287	A1388	A1368	A1315	A0535	A0498	A0402	-	-

Now, GTM2 had boundary lines with thickness greater than one pixel. (While thinner boundary lines could have been obtained by digitizing at 50 μ resolution, undesirable discontinuities would have resulted in that case). The boundary lines in GTM2 were thinned by a "peeling" algorithm which removes outer layers of "thick" lines while ensuring that connectivities are preserved. The resulting image, GTM3, was stored using the "scan line intersection code (SLIC)" wherein only the column coordinates of the intersections of each scan line with the boundary image are recorded.

The information in GTM3 was converted into a Region Identification Map GTM4 wherein unique numbers (RIN) were used to identify each connected region, the boundaries being denoted by 0. A table of correspondences was established manually between the CIN's of GTM0 and RIN's of GTM4. It was found, however, that there were some small discontinuities in the digitized boundaries which caused some large distinct regions to be merged. Therefore, the discontinuities in GTM3 were automatically sensed and patched by generating straight line segments. The result, GTM5, was converted into a Region Identification Map GTM6. The regions which were in error in GTM4 were now corrected as far as possible by a new table of RIN to CIN correspondences. The two correspondence tables were used on GTM4 and GTM6 to get a map GTM7, storing the CIN's at all the points in the properly identified regions, 0's at the boundary points and -1 at all the points in the regions still in doubt. The percentage of -1's in GTM7 is 2.8. These points represent locations in the image which, due to errors in GTM0, indicated nonunique or unavailable assignments of CIN's. The corrections to the GTM0 were received, but not in time to be incorporated into the present work.

Next, GTM7 was corrected to UTM coordinates using an affine transformation determined to minimize the mean squared error over a set of ground control points. The resulting image, GTM8 was arranged to have 20 pixels/km in both coordinate directions. The GTM, at this stage, was still a level II classification map. For the present analysis, a level I map was required. Therefore, GTM8 was modified by table look-up to GTM9 with only 6 classes (1-Urban, 2-Agriculture, 3-Forest, 4-Water, 5-Wet land, 6-Vacant, 0-Boundary, Unknown or exterior of the image region after geometric correction). Next, each of 0's in GTM9 was replaced by CIN occurring most frequently in the 3 x 3 region centered at it. Thus, 0's representing the boundaries between subclasses of level I classes were eliminated while retaining the 0's representing the unknown and exterior regions. The image, GTM10, is shown in Figure 4.1.

4.4 Comparisons With the GTM

The map GTM10 was used as the standard for evaluating the classification/cluster maps produced by the various methods. Only the points in GTM10 with class numbers 1 through 6 were used for determining the similarity measures. The following definitions of similarity measures parallel those in [1].

Table 4.6. GTM in Various Stages of Preparation

Stage	Description
0	Preliminary map (with disclaimer) supplied by U. S. Geological Survey.
1	Annotations removed and photographically reduced.
2	Digitized at 25 μ scanning interval.
3	Boundary lines thinned.
4	Regions identified by unique numbers (RIN) based on connectivity.
5	Discontinuities in boundary lines patched.
6	Regions identified by unique numbers (RIN) based on connectivity.
7	RIN's replaced by CIN's using a manually determined correspondence table with regions in doubt flagged with CIN = -1.
8	Geometrically corrected to the orientation of UTM coordinates with 20 pixels/km.
9	Reduced to Level I map by modifying the CIN's.
10	Boundary points replaced by CIN's occurring the largest number of times in their immediate neighborhoods.



Figure 4.1. Mobile Bay GTM

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

All the supervised classification maps (CM) have 6 classes - Urban, Agriculture, Forest, Water, Wet land and Vacant - with the exception of ELLTABCM which has a seventh "reject" class.

With A defined as the joint histogram between the GTM (rows) and a CM (columns), the similarity measure is defined as

$$\text{Tr (A) / S (A)}$$

$$\text{where Tr (A) = } \sum_{i=1}^6 a_{ii}$$

and S (A) = Sum of all elements of A.

In the case of unsupervised CM's, the columns of A are interchanged and/or merged according to a reassignment rule and then the above definition is used to find the similarity measure.

Due to uncertainties in registration and determination of geometric correction transformations, it is likely that the locations of the boundary pixels between classes in the GTM are unreliable. Therefore, the errors at the boundary and interior pixels are reckoned separately and similarity measures considering only the interior pixels are shown in addition to the "normal" similarity measures which consider the entire map. Also, "inventory" similarity measures are shown in each case indicating the closeness of the estimates of percentage occupancies of each Land Use Class in the CM's and the GTM. As in [1], this similarity measure is defined as $[1 - \sum |p_{1i} - p_{2i}| / 2\sum p_{1i}] 100$ percent. In the case of the unsupervised maps the similarity measure is based on the same measurements as for "normal".

In the following subsections, the CM's and the similarity matrices are presented for each of the algorithms discussed in Chapter 3. With the exception of ELLTAB, the classification methods have been applied to all six data sets described in Section 4.2. However, in each case, only one typical CM and a similarity matrix are shown, with the similarity measures tabulated for all six cases.

4.4.1 HINDU Classifier

This histogram dependent unsupervised method generated classification maps with different numbers of clusters for the six data sets. The map for the January 1975 data set, consisting of 8 classes, is shown in Figure 4.2. The joint histogram of GTM v/s HINDUCM for January 1975 is shown in Table 4.7. This being an unsupervised map, the class numbers were assigned to numbers 1 through 6 of the GTM classes and the similarity measures were evaluated. The similarity measures for the six HINDUCM's are shown in Table 4.8.



Figure 4.2. Mobile Bay HINDUCM (Jan. 1975)

Table 4.7. Joint Histogram Between GTM and HINDU Cluster Map

CLASS. NO.	1	2	3	4	5	6	7	8
1	72346	35816	2679	378	2814	1567	237	5093
2	81189	81745	203	31	6634	845	11401	1742
3	464658	47892	300	367	2315	3562	1632	1652
4	4439	1631	190250	184327	62	12020	10	1230
5	78792	3436	2656	502	42	8899	259	297
6	4856	2006	400	84	331	100	100	1912
GTM V/S HINDUCM(010575)								

Table 4.8. Similarity Measures of HINDUCM's w. r. t. GTM

Date	No. of Classes	Similarity Measures (%)		
		Normal	Boundaries Ignored	Inventory
Oct. 17, 72	9	67.42	69.90	76.15
Nov. 17, 73	8	70.96	73.54	90.45
Dec. 5, 73	9	73.37	76.08	83.82
Apr. 10, 74	8	69.67	72.39	78.80
June 21, 74	10	69.06	71.65	83.63
Jan. 5, 75	8	71.36	74.06	85.78

4.4.2 Table Look-up (ELLTAB)

The classification map using this table look-up implementation of the maximum likelihood method for the December 1973 data set is shown in Figure 4.3. It has 7 classes, the last being a "reject" class. The joint histogram of this map with the GTM is shown in Table 4.9. This method was used for the classification of only the December 1973 data. The similarity measures in this case were found to be:

Normal: 73.48%

Boundaries Ignored: 76.51%

Inventory: 98.28%

4.4.3 Linear Sequential

The output, LCM, of this classifier is shown in Figure 4.4 for the April 1974 data set. The joint histogram with GTM is shown in Table 4.10, and the similarity measures are given in Table 4.11 for all the six data sets.

4.4.4 Piecewise Linear (Histogram based)

This supervised classifier is implemented using a table look-up approach wherein, for a given grid size, the histogram cells corresponding to each of the feature vectors in the data set are identified first, a table of classifications of the occupied histogram cells is formed using training samples and a piecewise linear classification rule and, finally, the entire data set is classified by table look-up. The result, PCM, of applying this method to the October 1972 data set is shown in Figure 4.5. The corresponding joint histogram is shown in Table 4.12 and the similarity measures are given in Table 4.13.

4.4.5 Nearest Neighbor (Histogram based)

This is a supervised classifier similar in implementation to the piecewise linear classifier above. The table of classifications of the multidimensional histogram cells is produced using training samples and a nearest neighbor rule of class assignment. The result, SNEHACM, of this method is shown in Figure 4.6 for the November 1973 data set. The joint histogram and the similarity measures are shown in Tables 4.14 and 4.15 respectively.

4.4.6 Maximum Likelihood (Histogram based)

This classifier differs from the conventional maximum likelihood and ELLTAB in that it uses a multidimensional histogram with a specified grid size as in the case of PCM and SNEHACM. The resulting histogram cells are classified

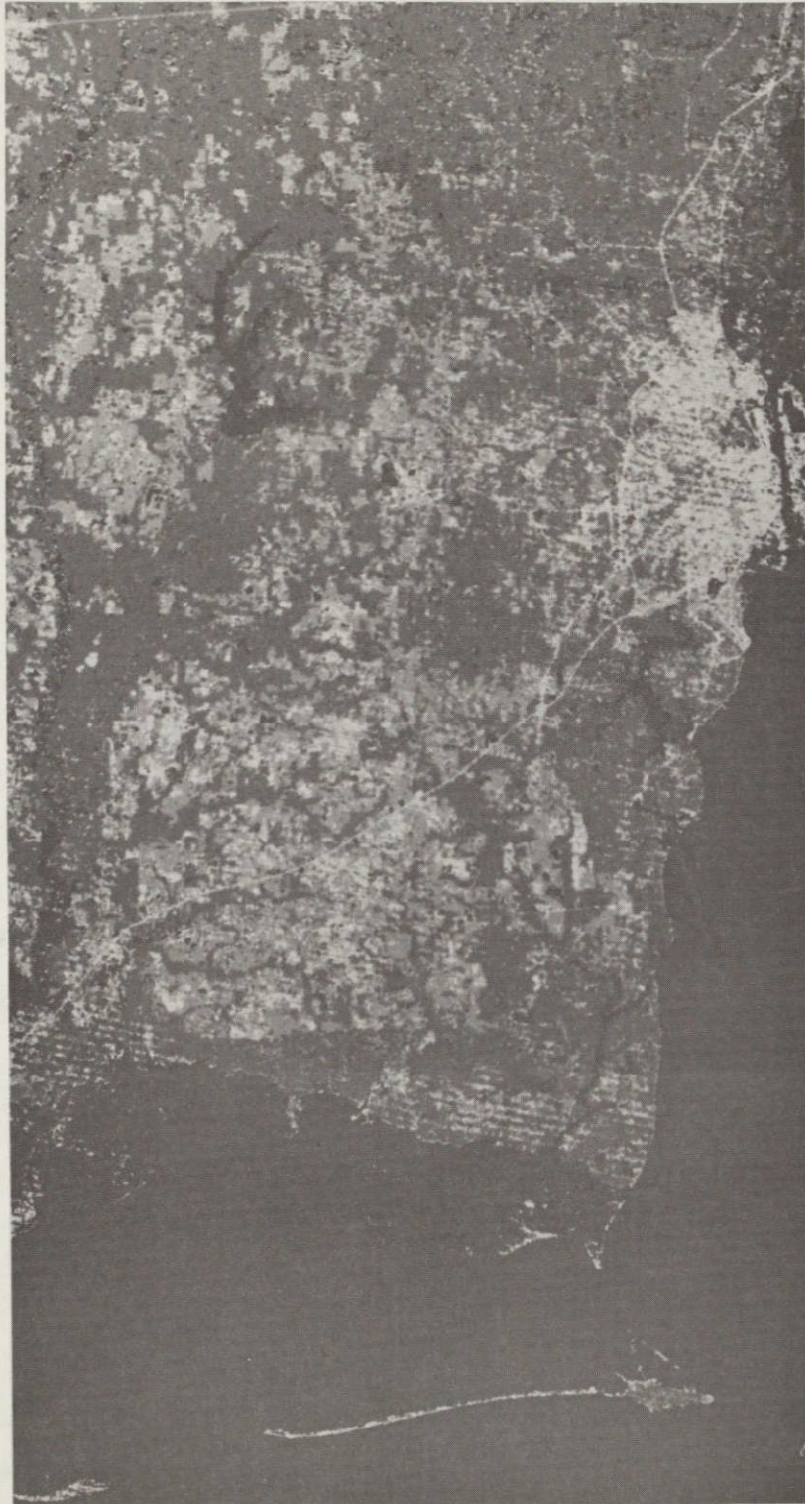


Figure 4.3. Mobile Bay ELLTABCM (Dec. 1973)

Table 4-9. Joint Histogram Between GTM and ELLTAB Classification Map

CLASS NO.	1	2	3	4	5	6	7
1	39610	26065	40534	2111	7151	2582	2982
2	37030	97570	29733	19	6414	185	2839
3	28758	47429	409922	391	32377	190	3311
4	1592	342	915	393754	5394	1309	3796
5	8263	4384	35277	2753	43947	270	1880
6	2456	1596	2818	350	846	1039	1404

GTM V/S ELLTABCM(120573)



Figure 4.4. Mobile Bay LCM (April 1974)

Table 4.10. Joint Histogram Between GTM and Linear Sequential Classification Map

CLASS NO.	1	2	3	4	5	6
1	43249	47676	21108	3129	4462	1411
2	23872	133034	22448	31	3445	960
3	41336	105547	359656	564	14984	291
4	1183	564	1199	388163	5664	1353
5	5991	3239	47075	3809	35515	92
6	1957	3526	2589	521	622	564

GTM V/S LCM(041074)

Table 4.11. Similarity Measures of LCM's w. r. t. GTM

Date	Similarity Measures (%)		
	Normal	Boundaries Ignored	Inventory
Oct. 17, 72	65.77	68.57	85.75
Nov. 17, 73	72.00	74.95	96.99
Dec. 5, 73	73.85	75.81	97.22
Apr. 10, 74	72.15	75.14	91.75
June 21, 74	65.35	67.88	85.99
Jan. 5, 75	71.02	73.90	93.94



Figure 4.5. Mobile Bay PCM (Oct. 1972)

Table 4.12. Joint Histogram Between GTM and Piecewise Linear Classification Map

CLASS NO.	1	2	3	4	5	6
1	52834	40019	16241	3406	6316	2143
2	61697	92098	11507	2682	9699	6107
3	97128	130987	269307	7941	8107	8908
4	2430	1073	869	376362	5878	19554
5	19820	16070	26893	9230	18023	7605
6	2885	3887	1030	548	482	1613

GTM V/S PCM(101772)

Table 4.13. Similarity Measures of PCM's w. r. t. GTM

Date	Similarity Measures (%)		
	Normal	Boundaries Ignored	Inventory
Oct. 17, 72	60.45	63.15	81.30
Nov. 17, 73	70.08	72.65	87.37
Dec. 5, 73	73.42	76.32	97.34
Apr. 10, 74	70.91	73.69	87.55
June 21, 74	68.41	71.00	96.19
Jan. 5, 75	69.76	72.52	90.02



Figure 4.6. Mobile Bay SNEHACM (Nov. 1973)

Table 4.14. Joint Histogram Between GTM and Nearest Neighbor Classification Map

CLASS NO.	1	2	3	4	5	6
1	18113	21173	60411	2030	6416	12832
2	32216	63975	68203	17	4572	14807
3	19220	20367	463351	345	13918	5177
4	1294	1518	1721	387982	11437	2358
5	2104	2270	50547	4110	37109	525
6	1819	1107	4108	500	690	2228

GTM V/S SNEHACM(111773)

Table 4.15. Similarity Measures of SNEHACM's w. r. t. GTM

Date	Similarity Measures (%)		
	Normal	Boundaries Ignored	Inventory
Oct. 17, 72	59.27	61.97	80.78
Nov. 17, 73	72.56	75.51	88.55
Dec. 5, 73	66.22	69.00	87.39
April 10, 74	55.28	57.74	73.36
June 21, 74	58.71	60.65	82.36
Jan. 5, 75	55.18	57.31	78.02

using estimates of means and covariance matrices for the various classes based on training samples. The result of this method is called PRITICM. Figure 4.7 shows a PRITICM of the June 1974 data set, the joint histogram of which appears in Table 4.16. The similarity measures (w.r.t. GTM) of the PRITICM's corresponding to all the six data sets are presented in Table 4.17.

4.5 Comparisons Between Classification Maps

It can be seen from the joint histograms in Section 4.4 that, in many cases, the off-diagonal elements are larger than the corresponding diagonal elements, particularly in the case of classes other than forest (3) and water (4). This is due to the fact that forest and water classes have large homogeneous regions while the other classes have relatively large numbers of boundary pixels, and errors in registration of the GTM and the Landsat images have the most significant effect near the boundary regions. To support the conclusion that the dissimilarities found between the GTM and the CM's are attributable to registration errors, several of the CM's for the December 1973 data set were compared relative to each other. The corresponding similarity measures are shown in Table 4.18. These numbers should be compared with the "normal" columns in the previous tables. It can be seen that these are much larger than the similarity measures relative to the GTM. Also, the joint histograms (not shown here) are all found to have dominant diagonals.

4.6 Chi-Square Tests for a Typical Case

Chi-Square tests were applied to test several hypotheses in the case of the comparison between the GTM and the LCM for the October 1972 data set. Starting with the joint histogram (contingency table) the χ^2 values were computed as defined in [2] for nine different hypotheses listed below:

1. The distribution of the classification inventory agrees with the distribution of the ground truth inventory.
2. The distribution of the correctly classified pixels agrees with the distribution of the ground truth inventory.
3. The distribution of the number of correctly and incorrectly classified pixels is optimum with respect to the given inventory and without regard to class.
4. The correctly classified pixels are randomly distributed.
5. Each ground truth feature is randomly distributed among the classification features according to the ground truth inventory.

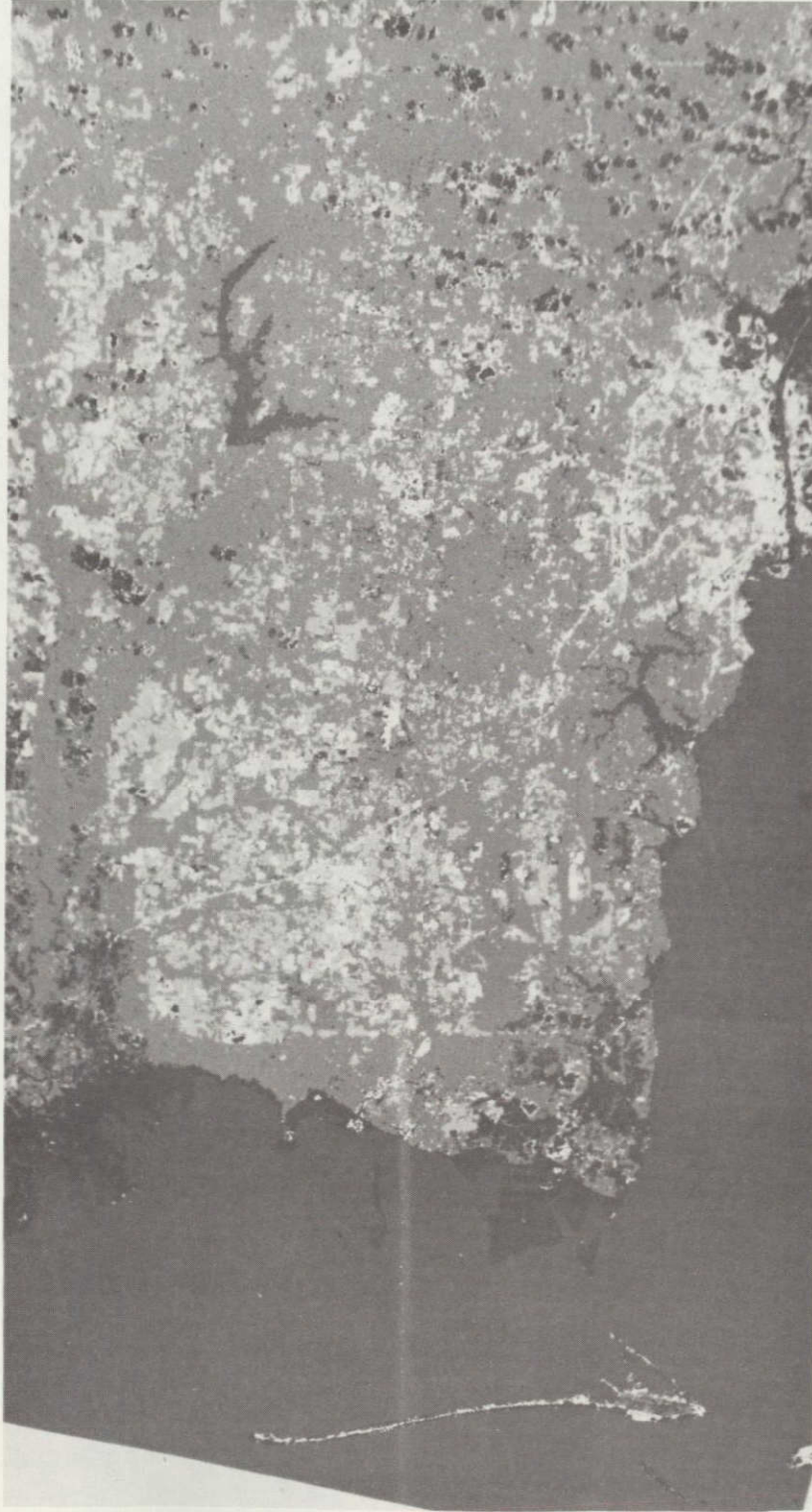


Figure 4.7. Mobile Bay PRITICM (June 1974)

Table 4.16. Joint Histogram Between GTM and Histogram-based Maximum Likelihood Classification Map

CLASS NO.	1	2	3	4	5	6
1	27870	9586	68571	1893	8735	4380
2	27822	63236	84056	82	7368	1226
3	21806	24030	428624	2726	37259	7933
4	1074	73	1651	373863	5604	1019
5	2692	858	47654	4666	39343	522
6	2402	732	3446	446	1591	1179

GTM V/S PRITICM(062174)

Table 4.17. Similarity Measures of PRITICM's w. r. t. GTM

Date	Similarity Measures (%)		
	Normal	Boundaries Ignored	Inventory
Oct. 17, 72	62.98	65.88	88.68
Nov. 17, 73	71.22	73.80	91.49
Dec. 5, 73	73.45	76.31	93.68
April 10, 74	70.75	73.60	90.39
June 21, 74	70.77	73.45	90.44
Jan. 5, 75	69.01	72.00	92.82

Table 4.18. Similarity Measures between CM's for Dec. 73
Data Set

Map 1	Map 2	Similarity Measure (%)
ETCM	LCM	89.07
ETCM	PCM	85.78
ETCM	PRITICM	86.07
SNEHACM	LCM	82.09
SNEHACM	PCM	85.44

6. Each classification feature is randomly distributed among the ground truth features according to the classification inventory.
7. The distribution of the number of correctly and incorrectly classified pixels is random without regard to class.
8. The numbers of correctly and incorrectly classified pixels for a particular class are randomly distributed.
9. The distribution of the classified pixels is independent of the ground truth.

The results of the test are shown in table 4.19. Here, the test numbers correspond to the hypothesis numbers above and D. O. F. is the number of degrees of freedom. It is found that the χ^2 values are all large and the corresponding probabilities are very close to zero indicating that the above hypotheses are false. However, these tests are too stringent [2].

4.7 General Remarks

4.7.1 Qualifiers

The following qualifiers are necessary in drawing the conclusions from the data presented in this chapter.

- The GTM used here was a preliminary draft accompanied by a disclaimer stating that it was meant only for field checking and review. However, it was used in the comparisons here and this step should therefore be considered preliminary.
- The UTM coordinates of the control points needed for the slight correction in orientation of the GTM were obtained from a map of scale 1:125000 with a similar disclaimer.
- The Landsat data were corrected using affine transformations to UTM coordinates, the parameters being obtained based on several ground control points. While the GCP's were chosen sufficiently scattered over the scene and the rms errors were smaller than one pixel at the GCP's, the peak errors were of the order of 2 pixels at the GCP's and could be 3 to 4 pixels elsewhere.

4.7.2 Conclusions

- It was found that the point by point ("normal") similarity measures w. r. t. GTM in the case of the data sets considered here are in the

Table 4.19. Results of the χ^2 Tests

CLASSIFICATION ACCURACY (PERCENT):				
	RANDOM		24.26	
	ACTUAL		65.77	
	OPTIMUM (INVENTORY)		85.75	
	PERCENT OF OPTIMUM ACCURACY = 67.45			
CHI-SQUARE TESTS				
TEST	FEATURE	D.C.F.	CHI-SQUARE	PERCEABILITY
1		5	2.50744188E 05	C.C
2		5	8.50451250E 04	0.0
3		5	4.37546563E 05	C.C
4		5	5.24296641E 04	C.C
5	1	5	9.76624375E 04	0.0
5	2	5	2.49355875E 05	C.C
5	3	5	3.31950063E 05	0.0
5	4	5	8.27560375E 05	C.C
5	5	5	1.41366125E 05	C.C
5	6	5	9.96310547E 03	0.0
6	1	5	1.05604063E 05	C.C
6	2	5	2.74882063E 05	C.C
6	3	5	3.23571375E 05	0.0
6	4	5	8.33603375E 05	C.C
6	5	5	1.11990438E 05	0.0
6	6	5	1.82125508E 04	C.0
7		1	1.24671200E 06	C.C
8	1	1	5.79363072E 04	0.0
8	2	1	2.14709500E 05	C.C
8	3	1	2.49020875E 05	C.C
8	4	1	8.08257500E 05	0.0
8	5	1	1.08216813E 05	C.C
8	6	1	9.51351953E 03	0.0
9		25	1.664785700E 06	C.0

same range as those for the Bald Knob, Tennessee data set [1]. The results for HINDUCM, LCM and ETCM are slightly better than in [1]. These are the only techniques common between the present case and [1] and they support the intuitive conclusion that large homogeneous regions tend to increase classification accuracy.

- The data sets from the November and December 1973 scenes result in higher similarity measures than the others in the case of most of the techniques considered. Also, the October and June scenes seem to result in lower similarity measures in all cases. It is found from the classification maps that these scenes have more cloud cover than the others and, since the cloud pixels are not identified and removed from consideration before finding the similarity measures, they contribute to misclassifications.
- The similarity measures between classification maps of the December data set are significantly higher than those between the CM's and GTM. This could be due to the fact that the CM's are perfectly registered relative to each other while the registration errors contribute to disparities between the CM's and GTM.
- The percentage of boundary pixels is smaller in the present data sets compared to that in [1] and the increase in similarity measures when the boundary pixels are ignored is not as significant. But, when registration is improved, it is expected that there will be greater increases in these similarity measures.
- In [1] it was seen that the similarity measures based on inventories only were very close to those when the boundary errors were ignored. It is found in the present case that the inventory similarities are much higher. This, again, points to registration errors.
- The results of the histogram-based, supervised, table look-up techniques (SNEHACM, PRITICM and PCM) depend on the grid-size chosen in deriving the histogram. Here the same grid-sizes as obtained in the case of HINDUCM's of the corresponding data sets were used for the supervised CM's. Whether higher similarity measures can be obtained with finer grids (i.e. higher radiometric resolution) needs further examination. From the point of view of computation time, however, it is not feasible to use these techniques with grid size unity (i.e. full resolution that is available in the data).

4.7.3 Suggestions for Further Work

It is clear that an improvement in registration of the GTM and the Landsat data sets is needed before further comparisons are made. When the final version of the GTM is received, the changes, if any, should be incorporated and an updated digital version of the GTM produced.

The classification maps in the present work have been at level I. It is difficult to obtain reliable training samples for a level II classification. Therefore, it seems unnecessary to have a GTM at level II. A significant reduction in the manual effort involved in establishing the correspondence between the region numbers and class numbers on the GTM would result if a level I GTM were available. The present digital GTM is at level II and, since much of the manual work has already been done, it seems appropriate to finish further corrections of it at level II. These corrections should include modifying the locations presently having -1's (confusion due to missing boundaries between distinct ground truth classes) and 99's (indicating that labels were unavailable on the GTM supplied).

The GCP (ground control point) coordinates should be determined more accurately and a nonlinear (instead of the affine) transformation be found for each of the data sets. It may not be sufficient to use just the EN term. Better results could be obtained (when the GCP's are accurately located) with all the quadratic terms included in the transformation. It is desirable that the error in registration be less than one pixel throughout the image.

The reasoning behind obtaining similarity measures with the boundary pixels ignored is that the boundary pixels tend to be mixtures of classes and hence errors in classifying them should be "excused". Thus, only the errors in classifying the interior pixels of a homogeneous region are attributable to the deficiencies of a classification technique. This argument can be extended to the case where there are uncertainties in registration. If the maximum error is known to be N pixels then the subset of the image to be considered for comparison should consist only of the points farther than N pixels from the nearest inter-class boundary. If such a subset is large enough to be statistically significant, it will prove to be useful in comparing the classification techniques.

5. TEMPORAL CHANGE DETECTION

5.1 Introduction

The repetitive, eighteen day cycle in the coverage of each Landsat satellite was planned in order to have the capability of examining changes in the same scene as a function of time. Hence it is advisable to examine some concepts and methodology for defining and detecting temporal changes through processing of multiple data sets collected in different Landsat passes over the same scene. It is assumed in this chapter that the data sets have previously been spatially registered.

Temporal change detection as a concept is easily defined as the process of identifying the changes that have taken place in a scene. However, the methodology to be adopted is to be tailored to meet the needs of the user in terms of what precisely is the change the user is looking for. For example, the user could be looking for a change in the shape of certain features in the scene, like a body of water, or the emphasis may be just on the inventory of certain classes, say, forest acreage. Thus, different view points and needs of users have to be accommodated in an automatic information processing system. It is therefore desirable to first establish the identity of each individual pixel in all the data sets uniquely which may then be further processed to delineate the type and extent of change the user is interested in. Further, the correspondence between the clusters or classes of the different data sets has to be established through inventory/spectral matching or manually through supervision.

Once the pixel identity in each image is established and a correspondence between the labels in the two images is derived, the problem is then reduced to overlaying the two images and flagging the pixels appropriately to identify the nature of change occurring at each pixel as well as whether the pixel is in the interior or boundary of a region. The details of the processing involved and the methodology developed to cater to this need are presented in the following sections.

5.2 Overview of the System

The information flow in the system, schematically represented in Figure 5.1, can be briefly described as follows. With the two data sets representing the two temporally separated images of the same scene as input, the preprocessor, consisting of a classification/clustering package (depending on whether the environment is supervised or unsupervised) and a registration and geometric correction package, produces two classification maps GCM1 and 2 which are geometrically corrected and registered with each other. The first image is then processed by the software subsystem FLGBDIES to identify and flag the interior and boundaries of clusters, leading to what will hereafter be referred to as the reference map. Either manually or through automated inventory/ spectral (e.g. MXSMLRTY)

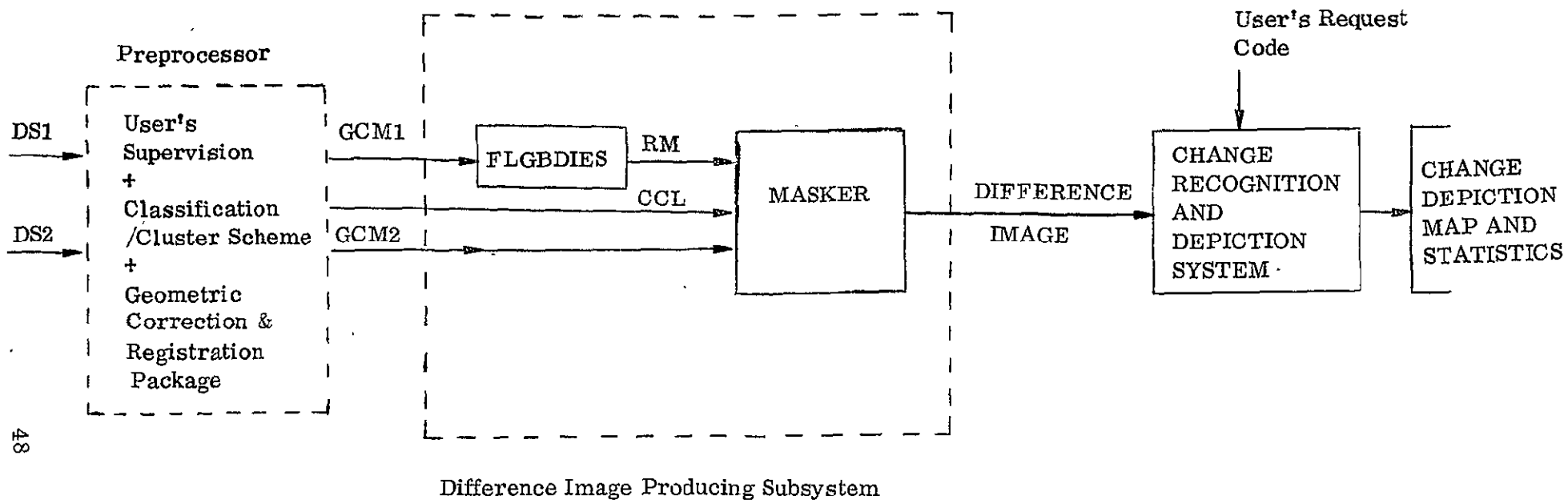


Figure 5.1. Schematic Representation of Temporal Change Detection Methodology

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR.

matching schemes, a cluster or class correspondence list correlating to two classification maps is derived. This list along with the reference map and the second map is then input to the MASKER subsystem. Here, a difference image of the two input images is produced which will have in coded form all the information (concerning each pixel) necessary for recognition of all possible types of changes occurring in the scene. In addition, the coded difference image has also the necessary information to determine whether a given pixel is an interior or boundary pixel in the reference map. With this coded image as input, the CHange Recognition And DEpiction (CHARADE) System is now ready to process the user's input request for detection and depiction of any particular type of change in the scene, as for example, change in the boundary of a certain class, say, the water bodies in the scene, etc.

5.3 Description of the Change Detection System

As described earlier in the overview of the system, and portrayed in Figure 5.1, the change detection system consists of three major components (subsystems)

- A preprocessing subsystem
- A difference image producing subsystem
- A change recognition and depiction subsystem

The preprocessing subsystem in turn consists of several smaller packages. The first significant constituent of the preprocessing subsystem is a classification/clustering scheme (depending, of course, on whether the environment is supervised or unsupervised) which, given the two temporally distinguished data sets (images) produces two classification (cluster) maps. The classification tool could be one of the several evaluated and reported elsewhere. These classification maps are then registered either relative to a common reference such as the UTM coordinates or with respect to each other. This is done by identifying several ground control points and finding appropriate geometric transformations (using the program GEOGREF). The transformations are applied to the two images using the program GEOCOR and the resulting images are shifted relative to each other to ensure proper overlaying.

The registered maps are then compared by using either automated inventory matching scheme such as MXSMLRTY or spectral matching or through the available supervision in the environment to derive a class (cluster) correspondence list correlating the class (cluster) number in the two maps. Thus, the output of this preprocessing system will be the two geometrically corrected maps GCM1 and GCM2 and a class correspondence list LCC.

The difference image producing subsystem consists of two components: a boundary flagging scheme which identifies and flags the boundary pixels (a boundary pixel being defined as a pixel with at least one neighbor different from itself) and a difference image coding scheme which produces a difference image coded so as to label uniquely the type of change occurring at the pixel. In addition, the code also identifies the pixel either as interior or boundary pixel. This code can be written as

$$IPIXEL (\cdot) = (IX (\cdot) - 1) * M + LCC (IY(\cdot))$$

where $IX (\cdot)$ is a value ranging from 1 to M for interior pixels and $M+2$ to $2M+1$ for boundary pixels, depending on the class to which the pixel is assigned in Map 1 with M as the number of classes in Map 1, $IY (\cdot)$ is a value varying from 1 to N depending on the class to which the pixel is assigned in Map 2 with N as the number of classes in Map 2, and $LCC (\cdot)$ is the class correspondence list with numbers ranging from 1 to M designating the equivalence between the classes in Map 2 and Map 1.

Thus the coded difference image represents a completely processed information encoded image which can be employed to determine and depict the type of change of interest to the user.

The CHAnge Recognition And DEpiction (CHARADE) subsystem essentially consists of a process designed to cater to the user's requests for depiction of a particular type of change. The user's input request code is decoded and the coded difference image is scanned to identify and categorize the pixels belonging to the class(es) of interest according to whether they are

- Pixels undergoing the type of change of interest to the user
- Pixels undergoing no change
- Pixels undergoing changes, but not the type specified by the user.

In addition, the rest of the scene of interest is flagged to distinguish it from the areas outside the image. The inventory of the different categories as well as a coded map depicting the pixels belonging to these different categories are produced and recorded. The output images are written out on tape and, in addition, printer plots are also displayed.

Thus, the change detection system, given the two multispectral data sets corresponding to two temporally distinguished images of a single scene, leads to change depiction maps of interest to the user.

5.4 Experimental Results

The change detection system was implemented on IBM 360/65 and tested using the multispectral data sets corresponding to the three passes of Landsat over Mobile Bay area on December 5, 1973; June 21, 1974 and January 5, 1975. The change detection scheme was applied to identify the changes from December to June and December to January. The classification maps were produced by the linear sequential method. Depiction of six different types of change was requested:

- total changes in all classes.
- changes in only the interior regions of all classes.
- changes in only the boundaries of all classes.
- additions (or gross growth) to class 1 (identified manually as Urban).
- deletions (or gross reduction) from the interior region of class 3 (identified manually as Forest)
- changes in the boundaries of class 4 (identified manually as water).

The statistics of these changes are given in Table 5.1

Change depiction maps are shown in Figures 5.2 - 5.6 for the December-January pair of classification maps.

5.5 Discussion of Results and Concluding Remarks

As can be observed in Table 5.1, more of the changes occur along the boundaries than in the interior regions. This is only to be expected as boundary pixels are not only more susceptible to changes but also likely to be mixtures of more than one land use class or category. The figures for inter-seasonal and intra-seasonal changes have to be compared and interpreted in the light of the fact the intra-seasonal case deals with winter season data sets only, although involving a temporal separation of 13 months. On the other hand, the inter-seasonal case while involving a temporal separation of 6 months only, deals with winter and summer data. The statistics reported in Table 5.1 confirm that the inter-seasonal changes are far more significant than other temporal effects. This is especially true in the case of Forest, for example, where the change from December 1973 to June 1974 is significantly higher than the change from December 1973 to January 1975 although the temporal separation is a lot more. Though only a few particular types of changes were looked into in these experiments the system has considerably more flexibility in depicting various other types of changes as discussed earlier.

Table 5.1...Temporal Change Detection Results

No.	Type of Change	DATA SETS CONSIDERED			
		December 5, 1973 Vs. June 21, 1974 INTER-SEASONAL		December 5, 1973 Vs. Jan. 5, 1975 INTRA-SEASONAL	
		No. of pixels changed	Percentage change	No. of pixels changed	Percentage change
1.	Total changes in all classes	442013	31.99	322484	23.24
2.	Changes in interior regions and all classes	158100	18.22	73566	8.42
3.	Changes in the boundaries of all classes	283913	55.24	248918	48.44
4.	Additions to class 1 (URBAN)	80745	5.84	67516	4.87
5.	Deletions from the interior of class 3 (FOREST)	112692	33.49	37738	11.22
6.	Changes in the boundaries of class 4 (WATER)	4500	40.77	4938	44.60



Figure 5.2. Total changes in all classes (represented by dark pixels)



Figure 5.3. Total changes in interior regions (black=change, gray=no change, white=boundary)

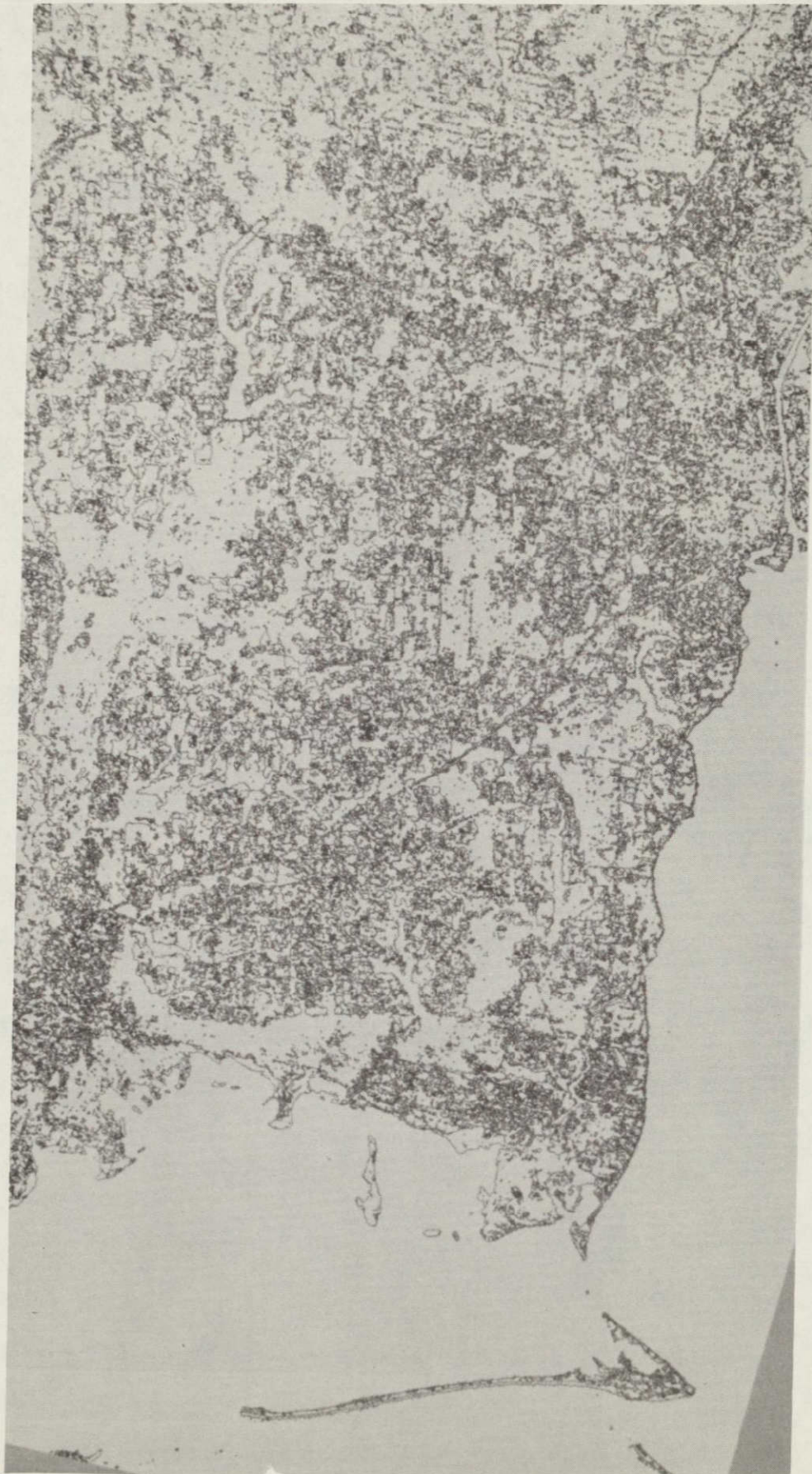


Figure 5.4. Total changes in boundary regions (black=change, gray=no change, white=interior)



REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

Figure 5.5. Additions to the urban class (black=change to urban, gray=no change, white=change to other than urban)

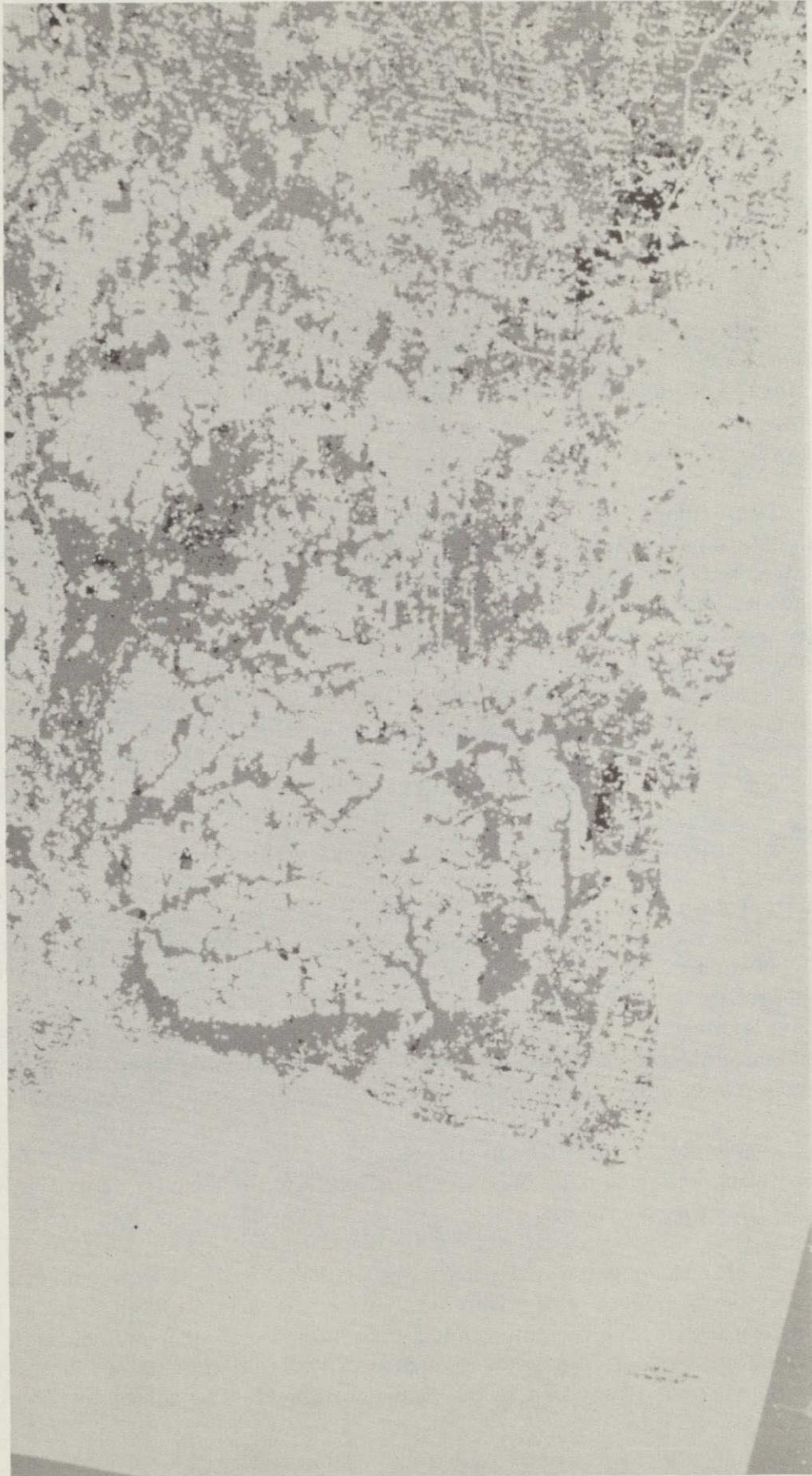


Figure 5.6. Deletions from the interior of the forest class (black=deletion from forest, gray=no change, white=pixels other than interior of forest)

6. DATA COMPRESSION

6.1 Introduction

Remote sensing by means of multispectral scanners such as those employed on the Landsat program and those planned for the Earth Observatory Satellite can result in extremely large quantities of data. The Landsat scanner, for example, could generate approximately 125 reels of 1600 bpi magnetic tape per day. It is apparent that data compression would yield benefits in the recording, transmission and storage of this information. Some of the more obvious benefits are reduced on-board storage, simpler data transmission, reduced ground data recording, and fewer data tapes to archive.

Data compression is accomplished by exploiting the structure or redundancy which exists between data samples. This means that the data does not take on all values with equal probability or that the value at any point is not totally independent of the data at every other point. It is apparent that spatial correlation exists, due to the extension of generally well defined regions on the ground over several neighboring pixels. Hence, the purpose of data compression is to transform the image data in order to reduce the degree of correlation between the samples so that redundancy in transmission is minimized.

6.2 The Karhunen-Loeve Transform

The Karhunen-Loeve [1,2] transform (also known as the eigenvector transformation, the principal components transformation, or the Hotelling [3] transformation) results in uncorrelated transform samples, thus minimizing redundancy. This transformation is optimal, if the criterion used is the mean squared error between the original image and the reconstructed image. The matrix required for implementing the KL transform has as its rows the eigenvectors of the covariance matrix of the data. It then follows that the principal components are uncorrelated (the covariance matrix of the principal components is diagonal), and the variances are the eigenvalues of the original covariance matrix.

Disadvantages are that the correlation matrix of the image must be known, necessitating two passes through the data, and a time-consuming matrix multiplication must be performed.

In order to describe mathematically the transform process, an image is modelled as a sequence of discrete data values with certain statistical properties. Experimental evidence indicates that a large variety of imagery data can be represented by a so-called markov sequence of variables having correlation decreasing exponentially with distance between terms. The autocorrelation matrix R is then given by

$$R_{ij} = p^{|i-j|}, \quad 0 < p < 1.$$

For a sequence of N samples, the elements of R range in value from unity to p^{N-1} . The correlation between neighboring samples is shown in the following definition:

$$x_i = p x_{i-1} + \epsilon_{i-1},$$

where the sequence $\{\epsilon_i\}$ has zero mean and variance $1-p^2$. The KL transform of the sequence $\{x_i\}$ is defined as the transformation that diagonalizes R , with elements the eigenvectors λ_i of R . Thus the transformed sequence $\{\hat{x}_i\}$ has uncorrelated samples with variances given by the eigenvalues.

The KL transform of the sequence $\{x_i\}$ is given by [4]

$$\hat{x}_i = \sum_{j=1}^N \frac{1}{\sqrt{N+\lambda_j^2}} \sin \left[\left(i + \frac{N+1}{2} \right) \omega_j + j \frac{\pi}{2} \right] x_j$$

$$\text{where } \lambda_j^2 = \frac{1-p^2}{1-2p \cos \omega_j + p^2} \quad \text{and}$$

$\{\omega_j\}$ are the positive roots of

$$\tan N\omega = \frac{(p^2-1) \sin \omega}{\cos \omega - 2p + p^2 \cos \omega}.$$

Since the values of ω are so defined (by a transcendental equation), the transformation is defined by nonperiodic sine functions, and no computational simplifications are possible.

6.3 A Fast Karhunen-Loeve Transformation

The KL transformation may be obtained from the Fast Fourier Transform (FFT) if a proper modification of the data is applied. [5] This is the minimum variance or interpolative representation, which is defined as

$$x_i = \bar{x}_i + v_i$$

where \bar{x}_i is a linear combination of the remaining terms in the sequence, i.e., $\{\bar{x}_j, j \neq i\}$. If the variance, $(x_i - \bar{x}_i)^2 = v_i^2$, is minimized, the definition of x_i reduces to

$$\bar{x}_i = \frac{p}{1+p^2} (x_{i+1} + x_{i-1}) + v_i.$$

Since this definition involves terms x_0 and x_{N+1} , the boundary or end conditions must be defined:

$$x_0 = px_1 + v_0$$

$$x_{N+1} = px_N + v_{N+1}$$

The minimum variance sequence does have nearest-neighbor correlation, and hence its correlation matrix ($N \times N$, tridagonal) has the form B^2Q where

$$B^2 = \frac{1-p^2}{1+p^2}$$

$$Q = \begin{bmatrix} 1 & \frac{-p}{1+p^2} & 0 & \\ \frac{-p}{1+p^2} & 1 & \frac{-p}{1+p^2} & 0 \\ 0 & \frac{-p}{1+p^2} & 1 & \frac{-p}{1+p^2} \\ & 0 & \frac{-p}{1+p^2} & 1 \end{bmatrix}$$

The minimum variance representation may be written in the following manner:

$$\frac{-p}{1+p^2} x_{i-1} + x_i + \frac{-p}{1+p^2} x_{i+1} = v_i$$

$$x_1 + \frac{-p}{1+p^2} x_2 = v_1 + \frac{-p}{1+p^2} x_0 = v_1 + b_1$$

$$\frac{-p}{1+p^2} x_{N-1} + x_N = v_N + \frac{-p}{1+p^2} x_{N+1} = v_N + b_N$$

The coefficients are the elements of the rows of matrix Q , and so the equations may be written in matrix form as

$$Q X = V + B$$

where B is an $N \times 1$ vector containing only the information at the end points, b_1 and b_N , with all other elements zero. Thus there is established a correspondence between the original sequence $\{x_i\}$ and the minimum variance sequence $\{v_i\}$. Furthermore, the eigenvectors of Q are [6]

$$W_{ij} = \sqrt{\frac{2}{N+1}} \sin \frac{ij \pi}{N+1}$$

and the eigenvalues are

$$\lambda_j = 1 - \frac{p}{1+p^2} \cos \frac{j \pi}{N+1} .$$

The W form the KL transform matrix of the sequence $\{v_i\}$ (being the eigenvectors of Q , the correlation matrix), and are periodic sine terms, and thus computable by an FFT algorithm.

Applying W to the matrix equation for $\{x_i\}$, we obtain

$$\begin{aligned} WQX &= WV + WB, \\ \text{or } \hat{X} &= \frac{\hat{V}}{\lambda} + \frac{\hat{B}}{\lambda}, \end{aligned}$$

using $WQ = \lambda W$ and denoting the KL transform WX by \hat{X} , etc. For zero mean data, the boundary conditions may be approximated by zero, and the KL transform becomes

$$\hat{x}_i = \frac{\hat{v}_i}{\lambda_i} .$$

Extension to two-dimensional image data is readily obtained. [5] The representation

$$x_i = \frac{p}{1+p^2} (x_{i+1} + x_{i-1}) + v_i,$$

which defines x_i in terms of its two nearest neighbors, is extended to include its eight nearest neighbors, as shown in Figure 6.1.

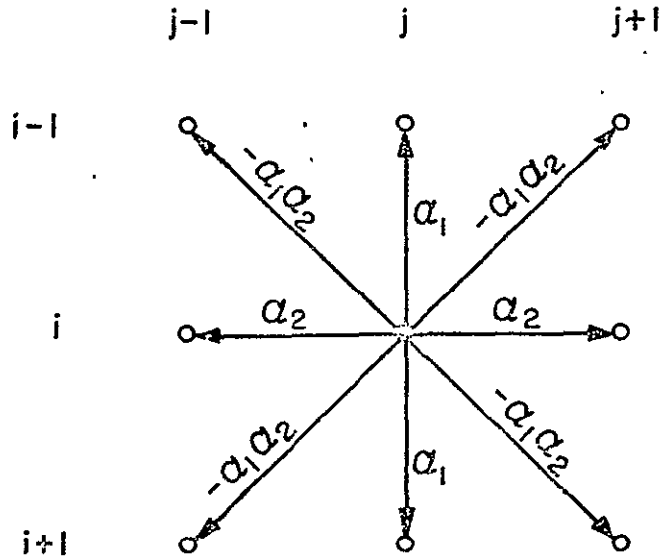


Figure 6.1. Coefficients of eight Nearest Neighbors,
where $\alpha = \frac{p}{1+p^2}$.

6.4 Implementation of the Algorithm

The steps required for coding a data compression program via the KL transform are as follows: [7]

- (i) Calculate the statistics (i.e. mean, variance, horizontal and vertical correlation parameters) of the source image.
- (ii) Create a zero mean image, by subtracting the image mean (or mean of the image block in the case of block by block coding) from each point in the image.
- (iii) Use the equation involving the data sample and its eight nearest neighbors to compute v_{ij} .

- (iv) Take the two dimensional sine transforms of v_{ij} to obtain \hat{v}_{ij} .
- (v) Calculate $\hat{x}_{ij} = \hat{v}_{ij} / \lambda_{ij}$.
- (vi) Quantize \hat{x}_{ij} using n_{ij} bits to obtain the values to be used for transmission. The number of quantization levels required for transmission is proportional to the transmitted value, \hat{x}_{ij} , and hence proportional to $1/\lambda_{ij}$ (\hat{v}_{ij} being relatively small and constant). The number of bits required to transmit $1/\lambda_{ij}$ is:

$$\log_2 (1/\lambda_{ij}) = -\log_2 \lambda_{ij}.$$

Hence, the number of bits to be assigned to the component (ij) varies according to

$$n_{ij} = b_1 - b_2 \log_2 \lambda_{ij}.$$

In terms of the number of bits assigned to the major component, n_{11} , and m , the average number of bits/pixel, n_{ij} is given by:

$$n_{ij} = n_{11} - \frac{N^2 (n_{11} - m) \log \lambda_{ij}/\lambda_{11}}{\sum_{i=1}^N \sum_{j=1}^N \log \lambda_{ij}/\lambda_{11}}$$

Since this expression must be converted to an integer for each n_{ij} , the actual bit rate obtained varies somewhat from the predicted value, m .

6.5 Results

Two computer programs described in reference [7] were implemented on a 255 x 255 segment of Landsat data covering the city of Mobile. The programs were an image analysis program and a Fast KL transform coding program. As supplied, the IBM 360/65 CPU times required were 6 1/2 minutes and 3 1/2 minutes, respectively.

The image analysis program performs the following tasks:

- (i) Compute and print the histogram and statistical parameters of the image
- (ii) Calculate the horizontal and vertical correlation parameters over all image blocks.

- (iii) Compute and list a desired bit rate constant (m) versus actual bit rate table.

The Fast KL transform coding program does the following jobs:

- (i) Create differential image $\{v_{ij}\}$ of a 15 x 15 image block.
- (ii) Apply Fast KL transform to $\{v_{ij}\}$.
- (iii) Calculate bit assignments to different elements in the transform domain.
- (iv) Perform quantization.
- (v) Apply inverse Fast KL transform.
- (vi) Store final result as a 255 x 255 image on magnetic tape.
- (vii) Compute and print histogram and statistics of the encoded image.

The total run time of 10 minutes for a 255 x 255 image is long, but is readily reduced by adjusting the quantization of the desired bit rate table, and by saving the image analysis parameters required in the transform coding program.

The statistical parameters of the original image and the reconstructed image are given in Table 6.1. The histograms of the input and final images and of the pixel-by-pixel differences are given in Figures 6.2, 6.3, and 6.4. The original and reconstructed images are shown in Figure 6.5.

Table 6.1. Statistics of the Input and Output Images

	Input Image	Output Image
Minimum	13.00	10.07
Maximum	82.00	67.01
Range	69.00	56.94
Mean	20.63	20.50
Standard Deviation	4.00	4.03
Mean Squared Error Between Images 2.34		

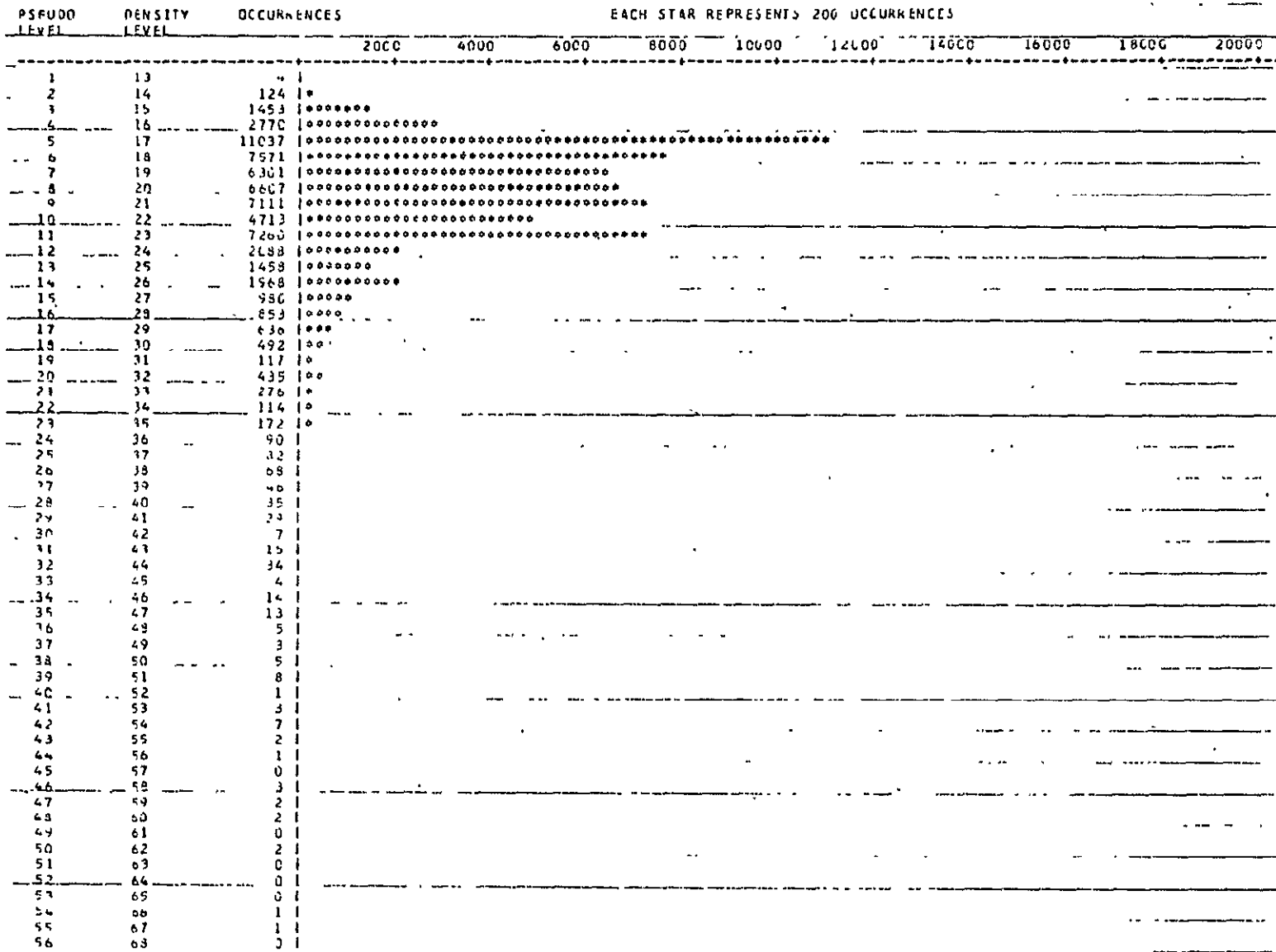


Figure 6.2. Histogram of Input Image.

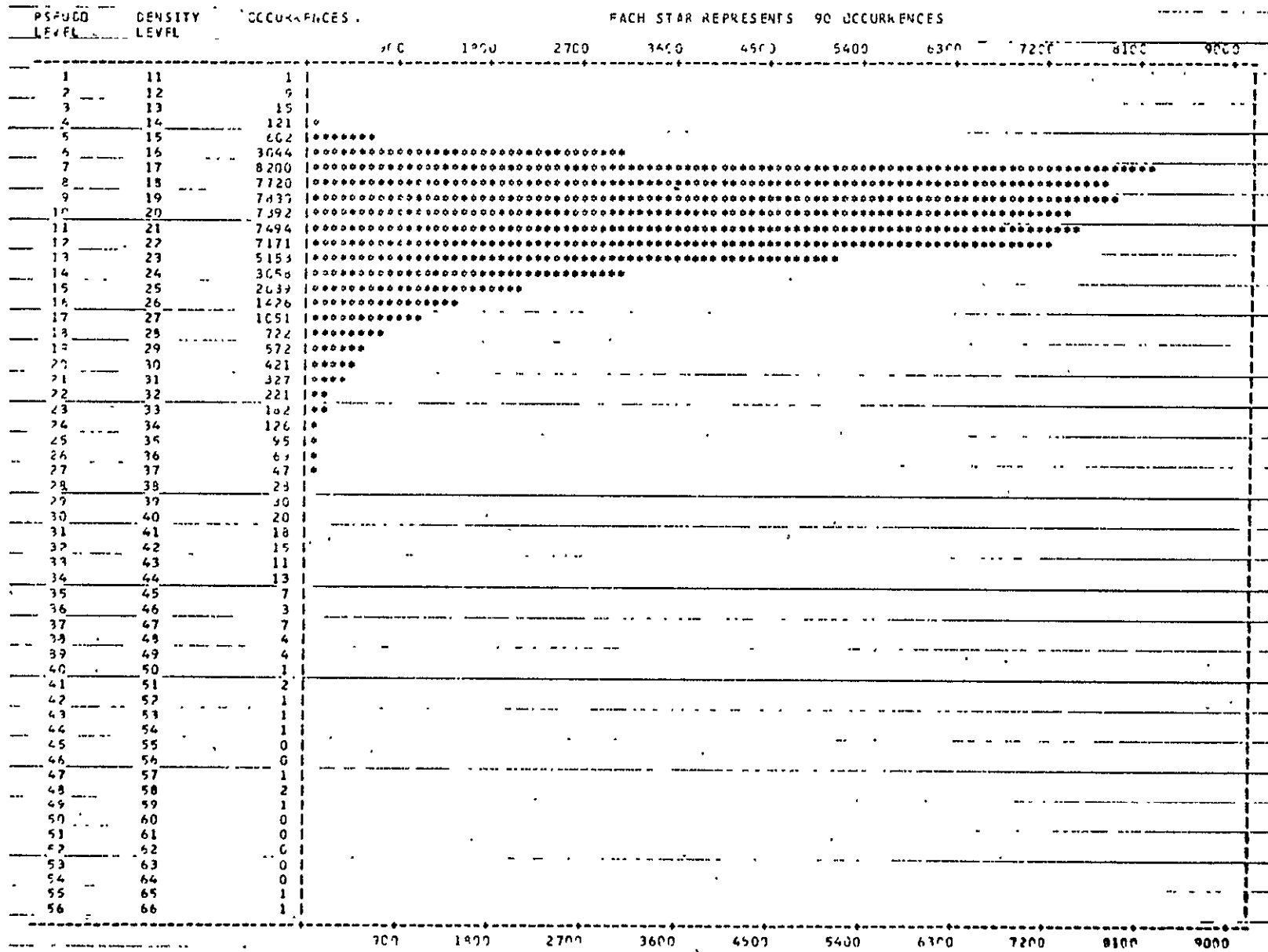
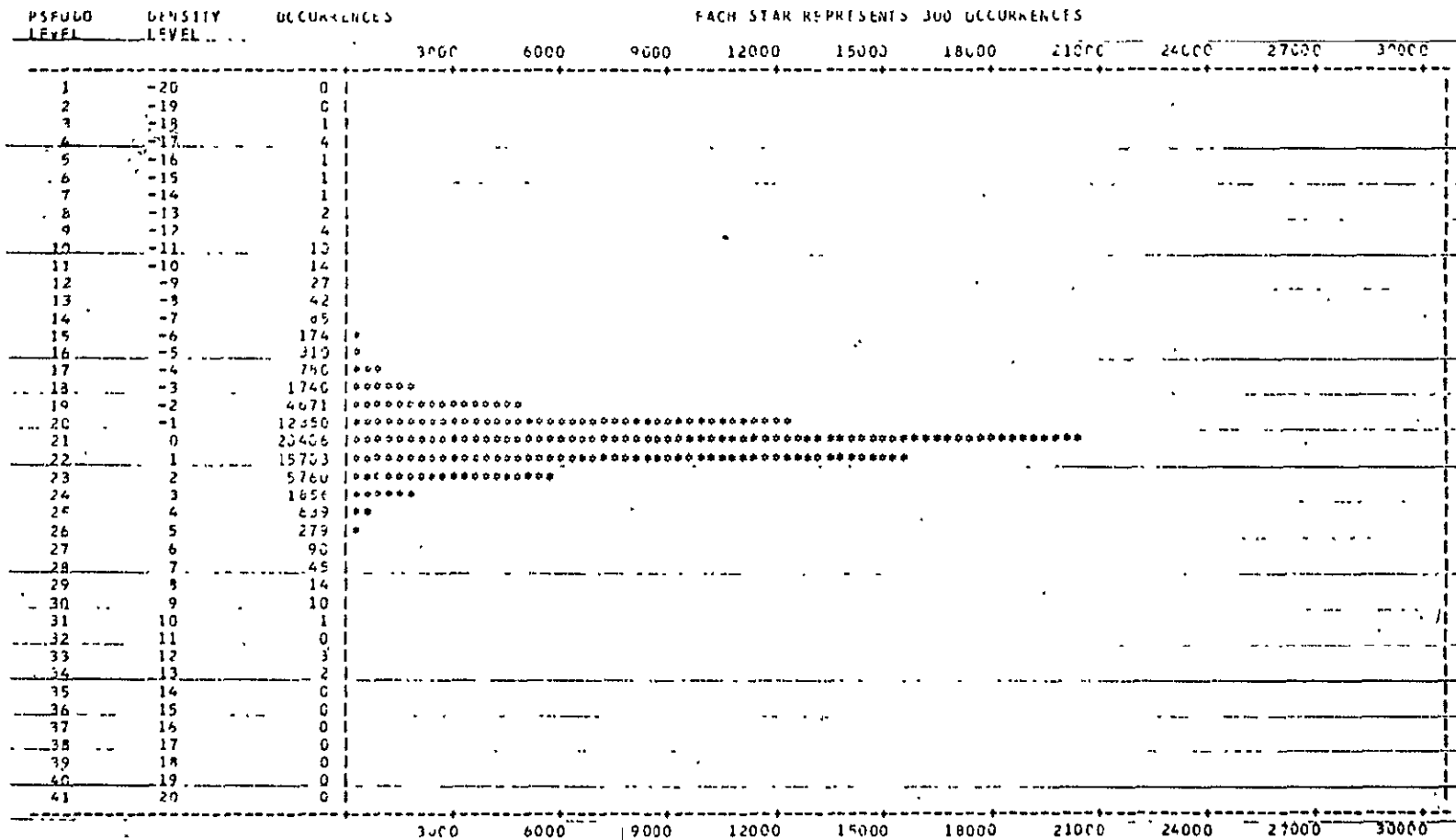


Figure 6.3. Histogram of Output Image.



58

Figure 6.4. Histogram of Output Image Minus Input Image

KARHUNEN-LOEVE TRANSFORM



ORIGINAL IMAGE



1 BIT/PIXEL RECONSTRUCTED IMAGE

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

Figure 6.5. Original and Reconstructed Images of the City of Mobile.

REFERENCES:

Chapter 1

1. R. R. Jayroe, Jr., et al, Classification Software Techniques Assessment, NASA TN D-8240, May 1976.

Chapter 2

1. M. Lybanon, "Geographic Referencing of Remotely Sensed Images Employed General Linear Transformations," CSC Memo No. 5E3030-1-4, January 29, 1975.
2. S. S. Rifman et al, "Experimental Study of Digital Image Processing Techniques for Landsat Data," Final Report, NASA Contract NAS5-20085, Report No. 26232-6004-TU-01, January 31, 1976.
3. H. K. Ramapriyan, "Geometric Correction of Remotely Sensed Images", Computer Sciences Corporation, Memorandum to File, 5E3030-1-1, September 13, 1974.
4. R. R. Jayroe, Jr., et al, Classification Software Technique Assessment, NASA TN D-8240, May 1976.
5. M. Lybanon, "Nonlinear Geometric Distortions in Landsat Multispectral Scanner Data," CSC Memo No. 5E3090-4-1, September 7, 1976.

Chapter 3

1. R. R. Jayroe, Jr., et al, Classification Software Technique Assessment, NASA TN D-8240, May 1976.
2. B. V. Dasarathy, "Histogram Inspired Neighborhood Discerning Unsupervised (HINDU) System of Clustering Multidimensional Data in Distribution-free Environments, CSC Memorandum to File, 5E3080-4-2.
3. B. V. Dasarathy, "SNEHA: Supervised Nearest-Neighbor Establishing Histogram Approach for Pattern Classification of Large Data Sets", CSC Memorandum to File, 5E3090-2-1, March 24, 1976.
4. B. V. Dasarathy, "PRITI: Parametric Recognition Imparting Trained Identification Approach for Processing Large Data Sets in Supervised Environment", CSC Memorandum to File, 5E3090-2-2, April 8, 1976.

REFERENCES (continued)

5. B. V. Dasarathy, "DHARMA: Discriminant Hyperplane Abstracting Residuals Minimization Algorithm for Separating Clusters with Fuzzy Boundaries", Proc. IEEE, Vol. 64, April 1, 1976.

Chapter 4

1. R. R. Jayroe, Jr., et al, Classification Software Technique Assessment, NASA TN D-8240, May 1976.
2. R. R. Jayroe, Jr., "Evaluation Criteria for Software Classification Inventories, Accuracies and Maps", NASA TMX (in publication)

Chapter 5

1. B. V. Dasarathy, "Temporal Change Detection in Scenes: Concepts and Methodology", CSC Memorandum to File, 5E3090-3-1, September 14, 1976.

Chapter 6

1. K. Karhunen, "Ueber Lineare Methoden in der Wahrscheinlichkeitsrechnung," Ann. Acad. Sci. Fenn., Ser. AI: Math. Phys., Vol. 37, 1947.
2. M. Loeve, Probability Theory, Van Nostrand, Princeton, NJ, 1960.
3. H. Hotelling, "Analysis of a Complex of Statistical Variables into Principal Components," J. Educ. Psychology, Vol. 24, p. 471, 1933.
4. W. D. Ray and R. M. Driver, "Further Decomposition of the Karhunen-Loeve Series Representation of a Stationary Random Process," IEEE Trans. Information Theory, Vol. IT-16, p. 663, 1970.
5. A. K. Jain, "A Fast Karhunen-Loeve Transform for Finite Discrete Images," Proc. National Electronics Conference, p. 323, 1974.
6. A. K. Jain, "Image Coding Via a Nearest Neighbors Image Model," IEEE Trans. Communications, Vol. COM-23, p. 318, 1975.
7. A. K. Jain, "Computer Program for Fast Karhunen-Loeve Transform Algorithm", Final Report, NASA Contract No. NAS8-31434, 1976.

PART II

1. INTRODUCTION

This part of the report is a formal documentation of the programs developed for the analysis and evaluation of multivariate decision methods for classification of remotely sensed data and change detection.

There are ten sections in this part, each section documenting one major software element. The programs are not detailed at the subroutine level, but are explained in terms of the inputs and outputs that one needs to know as a user. The subroutines needed for satisfying the external references are tabulated in each case.

The programs and most of the subroutines are in FORTRAN IV and are implemented on an IBM 360 with the H compiler. They are all available as load modules on a users' library. The names of the data sets on which the programs documented here were located at the time this report was prepared are:

SMART.DASARATY.LIBRARY
SMART.RAMPRIYA.D091576.LIBRARY

2. GEOMETRIC CORRECTION

2.1 NAME

GEOCOR8

2.2 PURPOSE

To apply geometric correction using nearest neighbor rule to a large rectangular image. The transformation from the output to input coordinate system can have eight parameters, six of them accounting for rotation, scale change, and shift and two providing a second degree term with the product of the output coordinates.

2.3 CALLING SEQUENCE

This is a main program. It is on a partitioned data set as a load module. The member name is GEOCOR8.

2.4 INPUT-OUTPUT

2.4.1 Input

The following input parameters should be supplied in data cards according to the formats and read statements indicated below.

```
      READ 100, NREC, NEL
      READ 200, A, XO, YO, ALFA, BETA, SX, SY
100   FORMAT (2I6)
200   FORMAT (6F12.3)
```

where

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

A is a 2x2 matrix accounting for rotation, scale change and skew;

XO, YO are the shift parameters;

ALFA, BETA are coefficients of the product term;

SX, SY are scale factors.

The transformation applied would then be

$$\begin{bmatrix} X \\ Y \end{bmatrix} = A \begin{bmatrix} XP \\ YP \end{bmatrix} + \begin{bmatrix} XO \\ YO \end{bmatrix} + \begin{bmatrix} ALFA \\ BETA \end{bmatrix} XP*YP$$

$$\begin{bmatrix} XP \\ YP \end{bmatrix} = \begin{bmatrix} XPP/SX \\ YPP/SY \end{bmatrix}$$

where

XPP, YPP are the coordinates in the output image and
X, Y are the coordinates in the input image.

Note: Generally, this program is used with A, XO, YO, ALFA, BETA found using a mean squared error minimization process with ground control points (e. g. GEOGREF [1]). The units of A are input pixels per km. Then SX and SY will indicate the number of output pixels desired per km in the XP and YP directions respectively.

The input image data should be as unformatted FORTRAN records with NEL words per record and one pixel per word.

2.4.2 Output

The output image data will have NRECO records (unformatted FORTRAN) with NELO words per record and one pixel per word. The values of NRECO and NELO are printed along with the coordinates of the top-left and bottom-right corners of the image. (see Section 2.9, Method). Also, some details about the implementation of the program are printed.

2.4.3 File Storage

This program requires a direct access file of 1500x1500 bytes for intermediate storage (NREC ≤ 1500, NEL ≤ 1500; for larger values of NREC or NEL, the DEFINE FILE statement and the space allocation for unit 90 should be changed).

2.5 EXITS

Not applicable

2.6 USAGE

The program is in FORTRAN IV and implemented on the IBM 360 using the H compiler. The program is in the users' library as a load module.

2.7 EXTERNAL INTERFACES

This program calls several routines. The linkage is indicated in the following table.

Calling Program	Programs Called
GEOCOR8	SARN DAWN GEOMM
GEOMM	GEOM1 DARN VNATS GEOM2 GEOM3 SAWN
GEOM1	VMOV GEOM1B XCHNGE
GEOM3	GEOM4
GEOM1B	GEOM1A
GEOM4	DARN

2.8 Performance Specifications

2.8.1 Storage

This program is 192680 bytes long, mainly due to an array IX dimensioned 48000 words. This array can be reduced in size, but the cost is an increase in direct access reading. Including the external references and buffers this program needs 256 K bytes of storage.

2.8.2 Execution Time

The time required depends largely on the output image size which in turn depends on the input image size and the transformation parameters. The time needed to correct a 1200x 1200 Landsat image to UTM coordinates to produce 20 pixels per km, thus generating approximately 2150x1850 pixels of output, is about 14 minutes on an IBM 360/65.

2.8.3 I/O Load

None except as specified by Section 2.4.

2.8.4 Restrictions

$NREC \leq 1500$; $NEL \leq 1500$ (See Section 2.4.3). The numbers on the input image should be between 0 and 255.

2.9 Method

The details of the method are presented in Section 2.5 of part I. These steps are implemented by the routine GEOMM. The main program GEOCOR8 first reads the input data from a sequential data set (unit 10), converts them into bytes and copies to the direct access data set (unit 90). The processed image will appear as a sequential data set on unit 8. The main feature of GEOMM is that it requires only one work array IX which it allocates for various buffers depending on the computed value of the output record length. The details of the subroutines follow the description in section 2.5 and are also apparent in the comments in the attached listing.

2.10 Comments

This program is designed to handle resampling with the nearest neighbor rule but can be modified easily to perform bilinear or bicubic interpolation. The present method of data handling involves considerably more I/O than that described in [3] which used segmentation, but was designed because computations needed for the nonlinear transformation would be complex under that approach.

2.11 Listings

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

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

2.12 Tests

The program has been checked out using a test pattern, applying a 45° rotation to it and printing the results. Also, a transformation with a small nonlinear term added to the 45° rotation has been tried. The program has been used to correct a classification map of the Landsat data of the Mobile Bay, Ala. test site to UTM coordinates using a nonlinear transformation.

COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=0C0CK,

SOURCE, EBCDIC, NOLIST, NODECK, LOAD, MAP, NOEDIT, ID, NOXREF

ISN 0002 COMMON/GEOM/A,XD,YD,ALFA,BETA,DET1,DET2,ILO,IHI,JLO,JHI,B,NREAD
 ISN 0003 DIMENSION A(2,2),B(2,2)
 ISN 0004 DIMENSION IX(48000)
 ISN 0005 EQUIVALENCE(IX(4001),LX(1))
 ISN 0006 LOGICAL*1 LX(1500)
 ISN 0007 DATA MAXC/48000/
 ISN 0008 DEFINE FILE 90(1500,1500,L,IAV)

C

READ INPUT IMAGE SIZE.

ISN 0009 READ 100,NREC,NEL

C

COPY INPUT IMAGE TO DISK.

ISN 0010 NEL4=NEL*4

ISN 0011 DO 10 I=1,NREC

ISN 0012 CALL SARN(10,IX,NEL4)

ISN 0013 DO 20 J=1,NEL

ISN 0014 LX(J)=IX(J)

ISN 0015 10 CALL DAWN(90,I,LX,NEL)

C

READ TRANSFORMATION PARAMETERS.

THE EIGHT PARAMETERS A,XD,YD,ALFA,BETA ARE THOSE FOUND BY GEOGREF.

SX AND SY ARE SCALE FACTORS IN THE E AND N DIRECTIONS IN PIXELS/KM.

ISN 0016 READ 200,A,XD,YD,ALFA,BETA,SX,SY

C

MODIFY THE TRANSFORMATION ACCOUNTING FOR SCALE FACTORS.

ISN 0017 A(1,1)=A(1,1)/SX

ISN 0018 A(2,1)=A(2,1)/SX

ISN 0019 A(1,2)=A(1,2)/SY

ISN 0020 A(2,2)=A(2,2)/SY

ISN 0021 ALFA=ALFA/SX/SY

ISN 0022 BETA=BETA/SX/SY

C

APPLY THE TRANSFORMATION.

ISN 0023 CALL GEOMM(IX,MAXC,NREC,NEL,8)

ISN 0024 STOP

ISN 0025 100 FORMAT(2I6)

ISN 0026 200 FORMAT(6F12.3)

ISN 0027 END

REPRODUCIBILITY OF THE
 ORIGINAL PAGE IS POOR
 ORIGINAL PAGE IS POOR

COMPILER OPTIONS - NAME= MAIN,OPT=C2,LINECNT=56,SIZE=0000K,

SOURCE, EBCDIC, NOLIST, NODECK, LOAD, MAP, NOEDIT, ID, NOXREF

ISN 0002 SUBROUTINE GEOM1(IX,MAXC,NREC,NEL,NTAPU)
 ISN 0003 DIMENSION IX(MAXC)
 ISN 0004 COMMON/GEOM/A,X0,Y0,ALFA,BETA,DET1,DET2,ILO,IHI,JLO,JHI,B,NRECO
 ISN 0005 DIMENSION A(2,2),B(2,2)

C

C PURPOSE: TO APPLY GEOMETRIC TRANSFORMATION TO A LARGE IMAGE STORED
 C ON A DIRECT ACCESS DATA SET(UNIT 90). THE OUTPUT WILL APPEAR ON
 C SEQUENTIAL ACCESS DEVICE NTAPD.
 C ONLY ONE ARRAY, IX, OF MAXC WORDS IS SUPPLIED AS A WORK AREA AND THE
 C ALLOCATIONS FOR VARIOUS SUBARRAYS IS HANDLED INTERNALLY.

C

C INPUTS: A, X0, Y0, ALFA, BETA ARE EIGHT PARAMETERS DEFINING A
 C TRANSFORMATION OF THE FORM:

C $(X \ Y) = (XP \ YP)A' + (X0 \ Y0) + (ALFA \ BETA)(XP*YP)$

C WHERE THE ' DENOTES TRANSPOSITION, X,Y ARE THE COORDINATES IN THE

C INPUT IMAGE, XP,YP ARE THOSE IN THE OUTPUT IMAGE.

C THE IMAGE ON UNIT 90 SHOULD BE IN BYTES(1 BYTE/PIXEL).

C

C OUTPUTS: A, X0, Y0, ALFA, BETA ARE MODIFIED BY THE ROUTINE GEOM1
 C FOR CONVENIENCE OF IMPLEMENTATION. BUT, FIRST, A IS TRANSFERRED
 C TO B SO THAT THE GIVEN TRANSFORMATION CAN BE RECOVERED IF NEEDED.

C ILO, IHI, JLO, JHI ARE THE COORDINATES(IN THE TRANSFORMED
 C IMAGE) OF THE TOP AND BOTTOM ROWS AND LEFT AND RIGHT COLUMNS OF
 C THE OUTPUT IMAGE.

C THE OUTPUT IMAGE WILL HAVE IHI-ILO+1 RECORDS WITH JHI-JLO+1

C WORDS PER RECORD(ONE WORD PER PIXEL).

C

C DEFINE FILE 90(NREC,NEL,L,IAV)

C

C FIRST, COMPUTE THE OUTPUT IMAGE SIZE AND MODIFY THE SUPPLIED TRANS-
 C FORMATION FOR CONVENIENCE OF IMPLEMENTATION.

ISN 0006 CALL GEOM1(NREC,NEL,NRECO,NEL0)

ISN 0007 WRITE(6,100)NREC,NEL,NRECO,NEL0

C

C IN COMPUTING AN OUTPUT RECORD, WE NEED TO ALLOW FOR 2 * NEL0 WORDS
 C FOR COMPUTING COORDINATES IN THE INPUT IMAGE CORRESPONDING TO EACH
 C POINT OF OUTPUT, NEL0 WORDS FOR STORING THE OUTPUT VALUES AND NEL0
 C WORDS FOR A "CODE ARRAY". THUS, MAXC-4*NEL0 WORDS ARE AVAILABLE
 C FOR A CIRCULAR BUFFER FOR THE INPUT RECORDS AND A BUFFER POINTER
 C ARRAY.

C

ISN 0008 MAXCP=MAXC-NEL0*4

ISN 0009 NR=MAXCP*4/(NEL+4)

ISN 0010 WRITE(6,200)MAXC,MAXCP,NR

C

C FIND STARTING ADDRESSES FOR WORK AREAS.

C

C X COORDINATES

ISN 0011 IAD1=MAXCP+1

C

C Y COORDINATES.

ISN 0012 IAD2=IAD1+NEL0

C

```

      C
      C   CODES INDICATING WHICH PARTS OF AN OUTPUT RECORD ARE TO BE COMPUTED
ISN 0013      C   IAD3=IAD2+NELO
      C
      C   OUTPUT RECORD.
ISN 0014      C   IAD4=IAD3+NELO
      C
      C   BUFFER POINTERS
ISN 0015      C   IAD5=IAD1-NR
      C
      C   INITIALIZE CIRCULAR BUFFER AND BUFFER POINTER ARRAYS.
ISN 0016      C   IRI=1
ISN 0017      C   IRF=1
ISN 0018      C   CALL DARN(90,1,IX,NEL)
ISN 0019      C   NREAD=1
ISN 0020      C   CALL VNATS(IX(IAD5),NR)
      C
      C   COMPUTE OUTPUT RECORDS.
ISN 0021      C   NELO4=NELO*4
ISN 0022      C   DO 10 I=1,NRECO
      C
      C   FIND X, Y COORDINATES IN THE INPUT IMAGE CORRESPONDING TO EACH OF
      C   THE OUTPUT POINTS IN THE I' TH RECORD.
ISN 0023      C   CALL GEOM2(IX(IAD1),IX(IAD2),I,NELC,NREC,NEL)
      C
      C   COMPUTE I' TH OUTPUT RECORD.
ISN 0024      C   CALL GEOM3(IX,IX(IAD1),IX(IAD2),IX(IAD3),IX(IAD4),IX(IAD5),NR,NREC
      C   ,NFL,NELO,IRI,IRF)
08 ISN 0025      C   IE(MOD(I,100),EQ,C)WRITE(6,400)I,NREAD
ISN 0027      10   CALL SAWN(NTAPO,IX(IAD4),NELO4)
ISN 0028      C   RREAD=FLOAT(NREAD)/FLOAT(NREC)
ISN 0029      C   WRITE(6,300)NREAD,RREAD
ISN 0030      C   RETURN
ISN 0031      100  FORMAT(' INPUT IMAGE SIZE = '15,' BY'15/
      C   ' OUTPUT IMAGE SIZE = '15,' BY'15)
ISN 0032      200  FORMAT(' BUFFER SIZE='16/ ' NUMBER OF WORDS AVAILABLE FOR INPUT RE
      C   'CORDS='16/ ' NUMBER OF INPUT RECORDS THAT CAN BE HELD IN CORE AT A
      C   ' TIME ='14,'.')
ISN 0033      300  FORMAT(' NUMBER OF CALLS TO DIRECT ACCESS READ ROUTINE='15/
      C   ' AVERAGE NUMBER OF TIMES EACH INPUT RECORD WAS READ='F9.2)
ISN 0034      400  FORMAT(' FINISHED COMPUTING'16,' RECORDS OF OUTPUT. NREAD='17)
ISN 0035      C   END

```


COMPILER OPTIONS - NAME= MAIN,OPT=C2,LINECNI=56,SIZE=C000K,

SOURCE,EPDCIC,NOLIST,NODECK,LOAD,MAP,NOFDIT,ID,NDXRFF

```

ISN 0002 SUBROUTINE GEOM1(NREC,NEL,NRECC,NELD)
ISN 0003 COMMON/GEOM/A,XD,YD,ALFA,BETA,DET1,DET2,ILO,IHI,JLO,JHI,B,NREAD
ISN 0004 DIMENSION A(2,2),B(2,2)
C THIS ROUTINE MODIFIES THE GIVEN TRANSFORMATION FOR CONVENIENCE OF
C COMPUTATION.
C IT IS ARRANGED SUCH THAT THE NUMBER OF INPUT RECORDS TO BE HELD IN
C CORE FOR COMPUTING ONE OUTPUT RECORD IS MINIMIZED AND X, Y INCREASE
C WITH RESPECT TO XP, YP RESPECTIVELY.
C
C PRINT THE INPUT PARAMETERS.
C
ISN 0005 WRITE(6,100)
ISN 0006 WRITE(6,200)((A(I,J),J=1,2),I=1,2)
ISN 0007 WRITE(6,300)XD,YD,ALFA,BETA
ISN 0008 CALL VMOV(A,4,B)
ISN 0009 DET1=A(1,1)*A(2,2)-A(1,2)*A(2,1)
ISN 0010 DET2=A(1,1)*BETA-A(2,1)*ALFA
ISN 0011 CALL GEOM1B(NREC,NEL)
C
C FIND MAXIMUM NUMBER OF INPUT RECORDS NEEDED TO PRODUCE A RECORD OF
C OUTPUT (R1 IF THE GIVEN TRANSFORMATION IS USED; R2 IF THE ROLES OF
C XP AND YP ARE INTERCHANGED).
ISN 0012 R1=ABS((A(1,2)+ALFA*ILO)*(JHI-JLO))
ISN 0013 R1=AMAX1(R1,ABS((A(1,2)+ALFA*IHI)*(JHI-JLO)))
ISN 0014 R2=ABS((A(1,1)+ALFA*JLO)*(IHI-ILO))
ISN 0015 R2=AMAX1(R2,ABS((A(1,1)+ALFA*JHI)*(IHI-ILO)))
ISN 0016 IF(R1.LE.R2)GO TO 10
C
C MODIFY THE TRANSFORMATION TO INTERCHANGE THE ROLES OF XP AND YP.
ISN 0018 CALL XCHNGE(A(1,1),A(1,2))
ISN 0019 CALL XCHNGE(A(2,1),A(2,2))
ISN 0020 CALL XCHNGE(ILO,JLO)
ISN 0021 CALL XCHNGE(IHI,JHI)
ISN 0022 CONTINUE
C
C MODIFY THE TRANSFORMATION, IF NECESSARY, TO ENSURE THAT X INCREASES
C WITH XP.
ISN 0023 IF(A(1,1).GE.0.)GO TO 20
ISN 0025 A(1,1)=-A(1,1)
ISN 0026 A(2,1)=-A(2,1)
ISN 0027 ALFA=-ALFA
ISN 0028 BETA=-BETA
ISN 0029 CALL XCHNGE(ILO,IHI)
ISN 0030 ILO=-ILO
ISN 0031 IHI=-IHI
ISN 0032 20 CONTINUE
C
C MODIFY THE TRANSFORMATION, IF NECESSARY, TO ENSURE THAT Y INCREASES
C WITH YP.
ISN 0033 IF(A(2,2).GE.0.)GO TO 30
ISN 0035 A(1,2)=-A(1,2)
ISN 0036 A(2,2)=-A(2,2)

```

18

```

ISN 0037      ALFA=-ALEA
ISN 0138      BETA=-BETA
ISN 0039      CALL XCHNGE(JLO,JHI)
ISN 0140      JLO=-JLO
ISN 0141      JHI=-JHI
ISN 0142      30  CONTINUE
ISN 0043      NRECO=IHI-ILO+1
ISN 0144      NELO =JHI-JLO+1

      C
      C PRINT MODIFIED TRANSFORMATION
ISN 0145      WRITE(6,400)
ISN 0046      WRITE(6,200)((A(I,J),J=1,2),I=1,2)
ISN 0047      WRITE(6,300)X0,Y0,ALFA,BETA
ISN 0148      WRITE(6,500)ILO,JLO,IHI,JHI
ISN 0049      RETURN
ISN 0151      100  FORMAT(///' GIVEN TRANSFORMATION PARAMETERS:')
ISN 0151      200  FORMAT(///' _MATRIX'/(1X2E15.7))
ISN 0152      300  FORMAT(//' SHIFT VECTOR:'2E15.7/
      ' COEFFICIENTS OF PRODUCT TERM:'2E15.7)
ISN 0153      400  FORMAT(///' MODIFIED TRANSFORMATION PARAMETERS:')
ISN 0054      500  FORMAT(///' TOP LEFT CORNER=(,'16,'','16,'') . . . BOTTOM RIGHT CORNER=(,'
      '16,'','16,'') .')
ISN 0055      END

```

COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=0000K,
SOURCE, EBCDIC, NOLIST, NODECK, LOAD, MAP, NOEDIT, ID, NOXREF

```
ISN 0002 SUBROUTINE GEOM1A(X,Y,XP,YP,ISIGN)
ISN 0003 COMMON/GFOM/A,XD,YD,ALFA,BETA,DET1,DET2,ILO,IHI,JLO,JHI,B,NREAD
ISN 0004 DIMENSION A(2,2),B(2,2)
      C
      C FIND XP, YP GIVEN X, Y.
ISN 0005 XXD=X-XD
ISN 0006 YYD=Y-YD
ISN 0007 BB=DET1+ALFA*YYD-BETA*XXD
ISN 0008 CC=YYD*A(1,2)-XXD*A(2,2)
ISN 0009 IF(ABS(DET2).GT.1.E-8)XP=(-BB+ISIGN*SQRT(BB*BB-4*DET2*CC))/
      (2*DET2)
ISN 0011 IF(ABS(DET2).LE.1.E-9)XP=-CC/BB
ISN 0013 YP=(YYD-A(2,1)*XP)/(A(2,2)+BETA*XP)
ISN 0014 RETURN
ISN 0015 END
```

COMPILER OPTIONS - NAME= MAIN,OPT=??,LINECNT=56,SIZE=C007K,

SOURCE,FBGDIC,NPLIST,NODECK,LOAD,MAP,NOEDIT,IO,NOXREF

```

ISN 0002  SUBROUTINE GEOM19(NREC,NEL)
ISN 0003  COMMON/GEOM/A,X0,Y0,ALFA,BETA,DET1,DET2,ILO,IHI,JLO,JHI,B,NREAD
ISN 0004  DIMENSION A(2,2),B(2,2)
      C
      C  E.IND MIN AND MAX VALUES OF XP, YP FOR X, Y RANGING FROM 1 TO NREC
      C  AND 1 TO NEL RESPECTIVELY.
ISN 0005  CALL GEOM1A(1.,1.,XP,YP,1)
ISN 0006  CALL GEOM1A(1.,1.,XPM,YPM,-1)
ISN 0007  ISIGN=1
ISN 0008  IF(ABS(XPM).GE.ABS(XP))GOTO 10
ISN 0009  ISIGN=-1
ISN 0010  XP=XPM
ISN 0011  YP=YPM
ISN 0012  XMIN=XP
ISN 0013  XMAX=XP
ISN 0014  YMIN=YP
ISN 0015  YMAX=YP
ISN 0016  NREC1=MAX0(NREC-1,1)
ISN 0017  NEL1=MAX0(NEL-1,1)
ISN 0018  DO 20 K=1,2
ISN 0019  IREC=1
ISN 0020  IEL=NEL1
ISN 0021  IF(K.EQ.1)GO TO 30
ISN 0022  IREC=NREC1
ISN 0023  IEL=1
ISN 0024  CONTINUE
ISN 0025  DO 20 I=1,NREC,IREC
ISN 0026  DO 20 J=1,NEL,IEL
ISN 0027  X=I
ISN 0028  Y=J
ISN 0029  CALL GEOM1A(X,Y,XP,YP,ISIGN)
ISN 0030  XMAX=AMAX1(XMAX,XP)
ISN 0031  XMIN=AMIN1(XMIN,XP)
ISN 0032  YMAX=AMAX1(YMAX,YP)
ISN 0033  YMIN=AMIN1(YMIN,YP)
ISN 0034  CONTINUE
ISN 0035  ILO=XMIN-1.5
ISN 0036  IHI=XMAX+1.5
ISN 0037  JLO=YMIN-1.5
ISN 0038  JHI=YMAX+1.5
ISN 0039  RETURN
ISN 0040  END

```

COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=0000K,

SOURCE, EBCDIC, NOLIST, NODFCK, LOAD, MAP, NOEDIT, ID, NOXREF

```
ISN 0002 SUBROUTINE GEOM2(X,Y,I,NELC,NREC,NEL)
ISN 0003 COMMON/GEOM/A,X0,Y0,ALFA,BETA,DET1,DET2,ILO,IHI,JLO,JHI,B,NREAD
ISN 0004 DIMENSION A(2,2),B(2,2)
```

C
C

EIND ARRAYS DE X AND Y COORDINATES FOR THE I*TH OUTPUT RECORD.

```
ISN 0005 DIMENSION X(NELO),Y(NELO)
ISN 0006 II=I+ILO-1
ISN 0007 DO 10 J=1,NELO
ISN 0008 JJ=J+JLO-1
ISN 0009 X(J)=A(1,1)*II+A(1,2)*JJ+XC+ALFA*II*JJ
ISN 0010 Y(J)=C
ISN 0011 IF(1.GT.X(J).OR.X(J).GT.NREC)GO TO 10
ISN 0013 Y(J)=A(2,1)*II+A(2,2)*JJ+Y0+BETA*II*JJ
ISN 0014 IF(1.GT.Y(J).OR.Y(J).GT.NEL)Y(J)=C
ISN 0016 10 CONTINUE
ISN 0017 RETURN
ISN 0018 END
```


COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=0000K,

SOURCE,PRCDIC,NOLIST,NODFCK,LOAD,MAP,NOCBIT,IO,NOXREF

```

ISN 0002      SUBROUTINE GEOM4(IN,X,Y,IREMC,IC,ICB,NR,NREC,NEL,NFLO,IRI,IRF)
C
C   COMPUTE ONE RECORD OF RESAMPLED OUTPUT BY READING THE NECESSARY
C   INPUT RECORDS FROM THE DIRECT ACCESS DEVICE (UNIT 90).
C   THIS PROGRAM IS DESIGNED TO HANDLE NEAREST NEIGHBOR AND BILINEAR
C   INTERPOLATION FOR RESAMPLING. THE COMPUTATIONS OF JRI, JRF SHOULD
C   BE CHANGED IN THE CASE OF BICUBIC INTERPOLATION.
ISN 0003      COMMON/GEOM4/A,X0,Y0,ALFA,BETA,DET1,DFT2,ILO,IHI,JLO,JHI,B,NREAD
ISN 0004      DIMENSION A(2,2),B(2,2)
ISN 0005      LOGICAL*1 IN
ISN 0006      DIMENSION ICB(NR),IREMC(NFLO),X(NELO),Y(NELO),IN(NEL,NR),IO(NELO)
C
C   DEFINE FILE 90(NREC,NEL,L,IAV)
C   INITIALIZE IREMC.
ISN 0007      DO 10 I=1,NELO
ISN 0008      IREMC(I)=1
ISN 0009      IO(I)=0
ISN 0010      10 IF(Y(I).EQ.C.)IREMC(I)=2
C
C   FIND MAX AND MIN OF RECORD NUMBERS NEEDED TO COMPUTE THE REMAINING
C   PART OF THE OUTPUT RECORD.
ISN 0012      IPASS=0
ISN 0013      JRI=1000000
ISN 0014      JRF=0
ISN 0015      IPASS=IPASS+1
ISN 0016      DO 20 I=1,NELO
ISN 0017      IF(IREMC(I).NE.1)GO TO 20
C
C   IREMC(I).EQ.1 INDICATES THAT IO(I) HAS NOT BEEN COMPUTED YET.
ISN 0019      IX1=X(I)
ISN 0020      JRI=MIN0(JRI,IX1)
ISN 0021      JRF=MAX0(JRF,IX1)
ISN 0022      20 CONTINUE
ISN 0023      JRF=MIN0(JRF+1,NREC)
ISN 0024      IF(JRI.GT.JRF)RETURN
C
C   NOW, JRI, JRF ARE THE MIN AND MAX RECORD NUMBERS NEEDED FOR
C   COMPUTING THE REMAINING PART OF THE OUTPUT RECORD.
C   IRI THROUGH IRF ARE THE RECORDS IN CORE. EXCEPT IN THE CASE
C   IPASS=1, AS MUCH AS POSSIBLE OF THE OUTPUT RECORD WILL HAVE BEEN
C   COMPUTED USING IRI THROUGH IRF BEFORE EACH CALL TO GEOM4.
C
C   IF(IRI.LE.JRI.AND.IRF.GE.JRF) THE ENTIRE OUTPUT RECORD CAN BE
C   COMPUTED.
ISN 0026      IDIR=0
C
C   IF IRF.LT.JRF READ FORWARD AND COMPUTE PART OF THE OUTPUT RECORD.
ISN 0027      IF(IRF.LT.JRF)IDIR=1
C
C   IF IRF.GE.JRF.AND.IRI.GT.JRI READ BACKWARD AND COMPUTE PART OF
C   THE OUTPUT RECORD.
ISN 0029      IF(IRF.GE.JRF.AND.IRI.GT.JRI)IDIR=-1
C

```

08

```
C   IF IPASS.EQ.1 THEN IT IS THE FIRST CALL TO GEOM4 CORRESPONDING TO
C   THE PRESENT RECORD. THEREFORE, COMPUTE AS MUCH AS POSSIBLE WITH-
C   OUT READING ANY NEW RECORDS.
ISN 0031   IF(IPASS.EQ.1)IDIR=C
C
ISN 0033   CALL GEOM4(IDIR,NLEFT,IRI,IRF,NR,JRI,JRF,IN,NEL,ICB,IEMC,NFLD,
            IN,X,Y)
ISN 0034   IF(NLEFT.GT.0)GO TO 3C
ISN 0036   RETURN
ISN 0037   END
```

COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=7000K,

SOURCE,F8CDIC,NOLIST,NODECK,LOAD,PAP,NOPRIT,IO,NOXREF

```

ISN 0002 SUBROUTINE GEOY4( IDIR, NLEFT, IRI, IRF, NR, JRI, JRF, IN, NEL, ICB, IREMC,
      NELO, IC, X, Y)
ISN 0003 LOGICAL*1 IN
ISN 0004 DIMENSION IN(NEL,NRI), ICB(NRI), IREMC(NFLO), X(NELO), Y(NELC), IN(NELO)
ISN 0005 DIMENSION A(2,2), B(2,2)
ISN 0006 COMMON/GEOM/A, X0, Y0, ALFA, BETA, DET1, DET2, ILO, IHI, JLO, JHI, B, NREAD

      C
      C COMPUTE A PART OF OF THE OUTPUT RECORD BY READING AS MANY RECORDS
      C AS POSSIBLE OF THE INPUT IMAGE IMMEDIATELY AHEAD OF OR BEHIND
      C THOSE ALREADY IN CORE.
      C THIS PROGRAM IS DESIGNED FOR NEAREST NEIGHBOR INTERPOLATION. THE
      C
ISN 0007 NR21=NR*2-1
ISN 0008 IRI0=IRI
ISN 0009 IF( IDIR) 10, 20, 30

      C
      C IDIR.LT.0. FIND IRI, IRF TO READ BACKWARD.
ISN 0010 10 IRI=MAX0( IRI-NR+1, JRI)
ISN 0011 IRF=MIN0( IRI+NR-1, IRF)
ISN 0012 IRI=IRI
ISN 0013 IR2=IRI0-1
ISN 0014 GO TO 40

      C
      C IDIR.GT.0. FIND IRI, IRF TO READ FORWARD.
ISN 0015 30 IRF0=IRF
ISN 0016 IRF=MIN0( IRF+NR-1, JRF)
ISN 0017 IRI=MAX0( IRF-NR+1, IRI)
ISN 0018 IRI=IRF0+1
ISN 0019 IR2=IRF

      C
      C MODIFY ICB SUCH THAT ICB(1) GIVES ADDRESS OF IRI'ITH INPUT RECORD.
ISN 0020 40 INC=IRI-IRI0 +NR21
ISN 0021 DO 50 I=1, NR
ISN 0022 50 ICB( I)=MOD( ICB( I)+INC, NR)+1

      C
      C READ THE NECESSARY INPUT RECORDS.
ISN 0023 DO 60 I=IR1, IR2
ISN 0024 NREAD=NREAD+1
ISN 0025 60 CALL DARN( 90, I, IN( I, ICB( I-IRI+1)), NEL )

      C
      C COMPUTE PARTS OF THE OUTPUT ARRAY THAT CAN BE COMPUTED.
ISN 0026 20 NLEFT=0
ISN 0027 DO 70 I=1, NELO
ISN 0028 IF( IREMC( I), NE.1) GO TO 70
ISN 0029 IR=X( I)+.5
ISN 0030 IF( IR.LT. IRI.OR. IR.GT. IRF) GO TO 75
ISN 0031 IR=ICB( IR-IRI+1)
ISN 0032 IC=Y( I)+.5
ISN 0033 IQ( I)=IN( IC, IR)
ISN 0034 IREMC( I)=2
ISN 0035 GO TO 70
ISN 0036 75 NLEFT=NLEFT+1

```

REPRODUCIBILITY OF THE
 ORIGINAL PAGE IS POOR

ISN 0039 70 CONTINUE
ISN 0040 RETURN
ISN 0041 END

COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=0000K,
SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOFOIT,IO,NIXREF

```
ISN 0002 SUBROUTINE VNATS(IX,N)  
ISN 0003 DIMENSION IX(N)  
ISN 0004 DO 10 I=1,N  
ISN 0005 10 IX(I)=1  
ISN 0006 RETURN  
ISN 0007 END
```

06

COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=0000K,
SOURCE,SPCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,LD,NOXREF

ISN 0002 SUBROUTINE XCHNGE(X,Y)
ISN 0003 W=X
ISN 0004 X=Y
ISN 0005 Y=W
ISN 0006 RETURN
ISN 0007 END

3. THINNING OF BOUNDARY IMAGES

3.1 NAME

PEELS

3.2 PURPOSE

Starting with the output of a microdensitometer digitizing a boundary image, to apply a given threshold of density and reduce the thickness of the boundary lines by "peeling" their outer layers while preserving the distinctness of regions separated by them.

3.3 CALLING SEQUENCE

CALL PEELS (NTAPI, NTAPO, NREC, NEL, IT, MPASS, MDEV,
NDEV, 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 (see Section 3.9, Method);

MDEV, NDEV are logical unit numbers of two direct access scratch data sets defined as indicated in the listing of PEELS;

LX, LY, IBDY are scratch arrays with LX, LY dimensioned as indicated in the listing and IBDY dimensioned NEL.

3.4 INPUT-OUTPUT

3.4.1 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.

3.4.2 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.

3.4.3 File Storage

This program requires two direct access scratch data sets to handle the intermediate iterations of the boundary data. The sizes of these data sets are indicated in the listings attached.

3.5 EXITS

No nonstandard exits.

3.6 USAGE

The program is in FORTRAN IV and implemented on the IBM 360 with the H compiler. The program is in the user's library as a load module.

3.7 EXTERNAL INTERFACES

This subroutine calls several subroutines and the linkage is shown in the following table.

3.8 PERFORMANCE SPECIFICATIONS

3.8.1 Storage

The subroutine PEELS is 1458 bytes long. However, including a driver (whose size depends largely on the dimensions of LX, LY, IBDY which are functions of NEL), the required subroutines and the buffers the program needs approximately 70K for handling NEL=2100.

3.8.2 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 Mobile Bay GTM (a 4000x2100 level II map with boundaries 3 and 4 pixels thick) the initial thresholding and reformatting took about 10 minutes and the subsequent iterations about 6 minutes each, with a final reformatting and copying step taking about 7 minutes. Thus, with MPASS=4, it takes about 40 minutes of CPU time to process the image.

Calling Program	Program Called
PEELS	PET SARN* VLTHR CMPRES DAWN* PEELER DARN EXPBDY
CMPRES	ISTORE ⁺
PEELER	SVSCI PEELR1 PEELRO DAWN*
EXPBDY	ILOAD ⁺
PEELR1	DARN BLSFTV BRSFTV [●]
PEELRO	IOR ⁺ ICOMP1 ⁺ IAND ⁺ BLSFTV
BLSFTV	ILOAD ⁺ ISTORE ⁺
BRSFTV	ILOAD ⁺ ISTORE ⁺

- * Entry under DARN
- + Logical function available in the user's library under a main member name LOGFUNC
- Entry under BLSFTV

3.8.3 Restrictions

None

3.9 METHOD

The program has three major steps:

- (i) Thresholding, compressing and writing on a direct access unit.
- (ii) Iterating to "peel" boundaries.
- (iii) Changing to SLIC format and writing on output sequential data set.

3.9.1 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)LY(I) = LX(I).GE.IT
IT (IT.LT.0)LX(I) = LX(I).LE.IABS(IT)
```

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.

The compressed boundary information is then written on the direct access unit MDEV using the routine DAWN.

3.9.2 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), we examine a 3x3 neighborhood centered around the point. Consider the array

```
a  b  c
d  e  f
g  h  i
```

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 follow 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 $\bar{b}h=1$. (Here, \bar{b} denotes the complement of b). Then, we need only check whether e is a nonessential boundary point, that is, whether two 0's in the 3x3 array which are disconnected will stay disconnected where e is made 0. Connectivity, in this context, is defined as the existence of a path not including 1's and consisting only of horizontal and vertical segments.

Now, it is easy to see that e is essential if and only if $a\bar{d}=1$ or $\bar{c}f=1$. Therefore, the condition for a top peel is that

$$\bar{b}h(\bar{a}+d)(\bar{c}+f)=1.$$

Equivalently, to perform a top peel we set

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

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 ($J(1) \rightarrow$ previous, $J(2) \rightarrow$ current, $J(3) \rightarrow$ 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), K+ 1.

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, of 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.

3.9.3 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.

3.10 COMMENTS

On large images this program takes a long time to execute. To avoid loss of data on long runs it is suggested that the direct access data sets be saved so that, with slight modifications, the routine PEELS can continue where the last run stopped due to insufficient CPU time.

3.11 LISTINGS

The listings of PEELS and most of the associated routines are attached at the end of this section. The routines not included are: PET, a routine used for printing time elapsed between sections of a program; SVSCI, a routine which sets all elements of an array to a given constant; DARN and the associated entry points for array read/write and the logical functions under member name LOGFUNC.

3.12 TESTS

The program was tested on a small portion of a boundary image, the image printed before and after peeling and was found to work satisfactorily.

COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=0000K,

SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NODEDIT,LD,NJXREF

```

ISN 0002      SUBROUTINE PEELS(NTI,NTD,NREC,NEL,IT,MPASS,MDEV,NDEV,LX,LY,IBDY)
ISN 0003      DIMENSION LX(1),LY(1),IBDY(1)

C
C      DIMENSION LX(36*((NEL-1)/32+1)),LY((NEL-1)/4+1)
C      DEFINE FILE MDEV(NREC,(NEL-1)/32+1,U,IAV1)
C      DEFINE FILE NDEV(NREC,(NEL-1)/32+1,U,IAV2)
C

ISN 0004      N=(NEL-1)/32+1
ISN 0005      CALL PET(2)
ISN 0006      DO 10 I=1,NREC
ISN 0007      CALL SARN(NTI,LX,NEL)
ISN 0008      CALL VLTHR(LX,NEL,IT,LY)
ISN 0009      LX(N)=0
ISN 0010      CALL CMPRES(LY,NEL,LX)
ISN 0011      10  CALL DAWN(MDEV,I,LX,N*4)
ISN 0012      CALL PET(2)
ISN 0013      DO 20 IPASS=1,MPASS
ISN 0014      IF(MOD(IPASS,2).EQ.1)CALL PEELER(MDEV,NDEV,NREC,N,LX,LX(12*N+1),
      LX(24*N+1),LY,NP)
ISN 0016      IF(MOD(IPASS,2).EQ.0)CALL PEELER(NDEV,MDEV,NREC,N,LX,LX(12*N+1),
      LX(24*N+1),LY,NP)

ISN 0018      PRINT 100,IPASS,NP
ISN 0019      CALL PET(2)
ISN 0020      IF(NP.EQ.0)GO TO 30
ISN 0022      20  CONTINUE
ISN 0023      IPASS=MPASS
ISN 0024      30  JDEV=NDEV
ISN 0025      IF(MOD(IPASS,2).EQ.0)JDEV=MDEV
ISN 0027      DO 40 I=1,NREC
ISN 0028      CALL DARN(JDEV,I,LX,N*4)
ISN 0029      CALL EXPBDY(LX,N,NEL,IBDY,J)
ISN 0030      40  WRITE(NTD)J,(IBDY(L),L=1,J)
ISN 0031      CALL PET(2)
ISN 0032      RETURN
ISN 0033      100 FORMAT(5X'DURING PASS NUMBER'13,' THROUGH PEELER'16,' WORDS OF CON
      -PRESSED BOUNDARY INFORMATION WERE CHANGED.')
ISN 0034      END

```

COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=0000K,
 SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NODEIT,IO,NOXREF

```

ISN 0002      SUBROUTINE BLSFTV(IX,N,IY)
ISN 0003      DIMENSION IX(N),IY(N)
ISN 0004      N1=N-1
ISN 0005      DO 10 I=1,N1
ISN 0006      IY(I)=ILOAD(IX(I+1),32,1)
ISN 0007      10  IY(I)=ISTORE(IX(I),IY(I),32,31)
ISN 0008      IY(N)=0
ISN 0009      IY(N)=ISTORE(IX(N),IY(N),32,31)
ISN 0010      RETURN
ISN 0011      ENTRY BRSFV(IX,N,IY)
ISN 0012      IY(1)=ILOAD(IX(1),32,31)
ISN 0013      DO 20 I=2,N
ISN 0014      IY(I)=ILOAD(IX(I),32,31)
ISN 0015      20  IY(I)=ISTORE(IX(I-1),IY(I),32,1)
ISN 0016      RETURN
ISN 0017      END
    
```

COMPILER OPTIONS - NAME= MAIN,DPT=02,LINECNT=56,SIZE=6000K,
 SOURCE,EBCDIC,NULIST,NODECK,LJAD,MAP,NOEDIT,LD,NJXREF

```

ISN 0002      SUBROUTINE BPRINT (LX,N,NEL,LY,LO,L1)
ISN 0003      DIMENSION LX(N)
ISN 0004      LOGICAL ILOAD
ISN 0005      LOGICAL*1 LY(NEL),LO,L1
ISN 0006      JWRD=1
ISN 0007      JBIT=33
ISN 0008      DO 10 I=1,NEL
ISN 0009      LY(I)=LO
ISN 0010      JBIT=JBIT-1
ISN 0011      IF(JBIT.NE.0)GO TO 20
ISN 0013      JBIT=32
ISN 0014      JWRD=JWRD+1
ISN 0015      20  IF(ILOAD(LX(JWRD),JBIT,1))LY(I)=L1
ISN 0017      10  CONTINUE
ISN 0018      PRINT 100,LY
ISN 0019      100  FORMAT(1X4(32A1,1X))
ISN 0020      RETURN
ISN 0021      END
    
```

COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=0000K,
SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,NOJREF

```
ISN 0002      SUBROUTINE CMPRES(LX,NEL,LY)
ISN 0003      LOGICAL*1 LX(NEL)
ISN 0004      DIMENSION LY(1)
ISN 0005      JWRD=1
ISN 0006      JBIT=33
ISN 0007      DO 10 I=1,NEL
ISN 0008      JBIT=JBIT-1
ISN 0009      IF(JBIT.NE.0)GO TO 20
ISN 0011      JBIT=32
ISN 0012      JWRD=JWRD+1
ISN 0013      20  IX=LX(I)
ISN 0014      LY(JWRD)=ISTORE(IX,LY(JWRD),JBIT,1)
ISN 0015      10  CONTINUE
ISN 0016      RETURN
ISN 0017      END
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=CCOOK,
SOURCE,EBLDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,NJKREF

```
ISN 0002      SUBROUTINE EXPBDY(LX,N,NEL,IBDY,J)
ISN 0003      DIMENSION LX(N),IBDY(1)
ISN 0004      LOGICAL ILOAD
ISN 0005      JWRD=1
ISN 0006      JBIT=33
ISN 0007      J=0
ISN 0008      DO 10 I=1,NEL
ISN 0009      JBIT=JBIT-1
ISN 0010      IF(JBIT.NE.0)GO TO 20
ISN 0012      JBIT=32
ISN 0013      JWRD=JWRD+1
ISN 0014      20  IF(.NOT.ILOAD(LX(JWRD),JBIT,1))GO TO 10
ISN 0016      J=J+1
ISN 0017      IBDY(J)=I
ISN 0018      10  CONTINUE
ISN 0019      RETURN
ISN 0020      END
```


COMPILER OPTIONS - NAME* MAIN,OPT=02,LINECNT=56,SIZE=9000K,
SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NODEIT,LD,NJXREF

```

1SN 0002      SUBROUTINE PEELR(MDEV,NDEV,NREC,N,LX,LXR,LXL,LY,NP)
1SN 0003      DIMENSION LX(N,3,4),LXR(N,3,4),LXL(N,3,4),LY(N),J(3)
1SN 0004      NREC1=NREC+1
1SN 0005      NREC2=NREC+2
1SN 0006      NREC3=NREC+3
1SN 0007      NREC4=NREC+4
1SN 0008      J(1)=1
1SN 0009      J(2)=2
1SN 0010      J(3)=3
1SN 0011      CALL SVSCI(LX,N*12,0)
1SN 0012      CALL SVSCI(LXR,12*N,0)
1SN 0013      CALL SVSCI(LXL,12*N,0)
1SN 0014      NP=0
1SN 0015      DO 10 I=1,NREC4
1SN 0016      DO 20 K=1,4
1SN 0017      IF(I.LE.NREC+K)GO TO 20
1SN 0019      CALL SVSCI(LX(1,J(3),K),N,0)
1SN 0020      CALL SVSCI(LXR(1,J(3),K),N,0)
1SN 0021      CALL SVSCI(LXL(1,J(3),K),N,0)
1SN 0022      20 CONTINUE
1SN 0023      IF(I.LE.NREC)CALL PEELR(MDEV,I,LX,J,N,LXR,LXL)
1SN 0025      IF(I.GT.1.AND.I.LE.NREC1)
      . CALL PEELR(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)
1SN 0027      IF(I.GT.2.AND.I.LE.NREC2)
      . CALL PEELR(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)
1SN 0029      IF(I.GT.3.AND.I.LE.NREC3)
      . CALL PEELR(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)
1SN 0031      IF(I.GT.4)
      . CALL PEELR(LX(1,1,4),LXR(1,1,4),LXL(1,1,4),J,N,4,
      . LY,0,0,NP)
1SN 0033      IF(I.GT.4)CALL DAWN(NDEV,I-4,LY,4*N)
1SN 0035      DO 30 K=1,3
1SN 0036      30 J(K)=MOD(J(K),3)+1
1SN 0037      10 CONTINUE
1SN 0038      RETURN
1SN 0039      END

```

108

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
COMPILER OPTIONS - NAME= MAIN,DPT=02,LINECNT=56,SIZE=0000K,  
SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,1D,NJXREF  
ISN 0002 SUBROUTINE PEELR1(NDEV,I,LX,J,N,LXR,LXL)  
ISN 0003 DIMENSION LX(N,3),LXR(N,3),LXL(N,3),J(3)  
ISN 0004 CALL DARN(NDEV,I,LX(1,J(3)),N*4)  
ISN 0005 CALL BLSFTV(LX(1,J(3)),N,LXL(1,J(3)))  
ISN 0006 CALL BRSFTV(LX(1,J(3)),N,LXR(1,J(3)))  
ISN 0007 RETURN  
ISN 0008 END
```

COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=0000K,
SOURCE,EBCDIC,NULIST,NUDECK,LJAD,MAP,NDEDIT,LD,NUXREF

```

ISN 0002      SUBROUTINE PEELR(LX,LXR,LXL,J,N,ISIDE,LY,LYR,LYL,NP)
ISN 0003      DIMENSION LX(N,3),LXR(N,3),LXL(N,3),LY(N),J(3),IW(3),LYR(N),LYL(N)
ISN 0004      DO 60 I=1,N
ISN 0005          LY(I)=LY(I,J(2))
ISN 0006          IF(LY(I).EQ.0)GO TO 60
ISN 0008          GO TO (10,20,30,40),ISIDE
ISN 0009      10      IW(1)=IDR(LX(I,J(1)),ICOMP1(LX(I,J(3)),32,32))
ISN 0010          IW(2)=IAND(LXR(I,J(1)),ICOMP1(LXR(I,J(2)),32,32))
ISN 0011          IW(3)=IAND(LXL(I,J(1)),ICOMP1(LXL(I,J(2)),32,32))
ISN 0012          GO TO 50
ISN 0013      20      IW(1)=IDR(LXR(I,J(2)),ICOMP1(LXL(I,J(2)),32,32))
ISN 0014          IW(2)=IAND(LXR(I,J(1)),ICOMP1(LX(I,J(1)),32,32))
ISN 0015          IW(3)=IAND(LXR(I,J(3)),ICOMP1(LX(I,J(3)),32,32))
ISN 0016          GO TO 50
ISN 0017      30      IW(1)=IDR(LX(I,J(3)),ICOMP1(LX(I,J(1)),32,32))
ISN 0018          IW(2)=IAND(LXR(I,J(3)),ICOMP1(LXR(I,J(2)),32,32))
ISN 0019          IW(3)=IAND(LXL(I,J(3)),ICOMP1(LXL(I,J(2)),32,32))
ISN 0020          GO TO 50
ISN 0021      40      IW(1)=IDR(LXL(I,J(2)),ICOMP1(LXR(I,J(2)),32,32))
ISN 0022          IW(2)=IAND(LXL(I,J(1)),ICOMP1(LX(I,J(1)),32,32))
ISN 0023          IW(3)=IAND(LXL(I,J(3)),ICOMP1(LX(I,J(3)),32,32))
ISN 0024      50      IW(1)=IDR(IW(1),IW(2))
ISN 0025          IW(1)=IDR(IW(1),IW(3))
ISN 0026          LY(I)=IAND(LY(I),IW(1))
ISN 0027          IF(LX(I,J(2)).NE.LY(I))NP=NP+1
ISN 0029      60      CONTINUE
ISN 0030          IF(ISIDE.EQ.4)RETURN
ISN 0032          CALL BLSFTV(LY,N,LYL)
ISN 0033          CALL BRSFTV(LY,N,LYR)
ISN 0034          RETURN
ISN 0035          END

```

COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=0000K,
 SPURCF,EBCDIC,NULIST,NUDECK,LOAD,MAP,NUEDIT,IO,N)XREF

```

ISN 0002      SUBROUTINE VLTHR(LX,N,IT,LY)
ISN 0003      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.GE.0, LX(I).GE.IT IMPLIES LY(I)=T. IF IT<0 LX(I).LE.IABS(IT)
C      IMPLIES LY(I)=T.
C
ISN 0004      ITT=IABS(IT)
ISN 0005      IF(IT.LT.0)GO TO 10
ISN 0007      DO 20 I=1,N
ISN 0008      LY(I)=F
ISN 0009      20 IF(LX(I).GE.ITT)LY(I)=T
ISN 0011      RETURN
ISN 0012      10 DO 30 I=1,N
ISN 0013      LY(I)=F
ISN 0014      30 IF(LX(I).LE.ITT)LY(I)=T
ISN 0016      RETURN
ISN 0017      END
    
```

4.1 FINDING DISCONTINUITIES IN BOUNDARY DATA

4.1 NAME

BOUDIM

4.2 PURPOSE

To find the discontinuities in digital curves stored in SLIC format.

4.3 CALLING SEQUENCE

CALL BOUDIM (IBDY, NTAPI, NREC, NEL, IRC, ND, NDIS)

where

IBDY is a scratch array to be dimensioned NEL*3 where NEL is the maximum number of boundary points in a given line;

NTAPI is the logical unit number on which the input boundary data are stored;

NREC is the number of lines (records) in the input data set;

IRC is the output array of coordinates of the discontinuities;

ND is the maximum number of discontinuities expected [IRC is dimensioned (ND, 2)];

NDIS is the output integer giving the actual number of discontinuities found.

NTAPI, NREC, NEL, ND are inputs to this routine IRC, NDIS are outputs.

4.4 INPUT-OUTPUT

4.4.1 Input

The input data should be on logical unit NTAPI as a sequential data set consisting of NREC records. Each record should consist of the coordinates of the intersection of the corresponding scan line with the boundary image written as

$$J, (X(I), I = 1, J)$$

where J is the number of such intersections and IX(I) are the coordinates.

4.4.2 Output

The output of this program is only through the calling sequence.

4.4.3 File Storage

None

4.5 EXITS

No nonstandard exits.

4.6 USAGE

This program is written in FORTRAN IV and is implemented on the IBM 360 with the H compiler. It is available on the users' library as a load module.

4.7 EXTERNAL INTERFACES

The linkage with other subroutines needed with this routine is indicated in the following table.

Calling Program	Program(s) Called
BOUDIM BOUDIS	BOUDIS JCOUNT

4.8 PERFORMANCE SPECIFICATIONS

4.8.1 Storage

This program is 834 bytes long. Including the external references listed above, the storage needed will be 2578 bytes (excluding the calling program which should provide storage for the arrays IRC and IBDY).

4.8.2 Execution Time

TBD

4.8.3 I/O Load

None

4.8.4 Restrictions

None

4.9 METHOD

The routine BOUDIM simply handles the I/O needed for finding the discontinuities. Connectivity, in this context, is defined in terms of the eight nearest neighbors of the point under consideration. Therefore, while examining the i th record of data, it is necessary to have the $(i-1)$ st and $(i+1)$ st records in core. The movement of data in core is avoided by using a circular buffer IBDY dimensioned (NEL, 3) and indexed by the pointer array IND dimensioned 3. Initially, IND is set to {1, 2, 3}. Always, IND(2) points to the current row. The numbers of boundary points in the three rows stored in core are in (NB(IND(J)), J=1, 3). The routine BOUDIM starts by reading the first record into IBDY (*, 2). Then, for I=1, NREC the $(I+1)$ st record is read into IBDY (*, IND(3)). The (NREC+1)th record is undefined. Therefore, in that case, NB(IND(3)) is simply set to 0. The routine BOUDIS is called to determine the coordinates of the discontinuities on the I 'th record. Then the pointer array IND is updated.

The functioning of BOUDIS is as follows. Each of the boundary points in the current record is treated as the point e in the following array.

a	b	c
d	e	f
g	h	i

The number of boundary points in this array excepting e is called the connectivity count of e . The connectivity count is calculated by examining the arrays IBDY (*, IND(2)), IBDY (*, IND(1)) and IBDY (*, IND(3)), stopping the calculations when the count equals 2. If the count is smaller than 2, then the point e is a discontinuity. The row and column coordinates of e and the continuity count are then stored in (IRC(NDIS,K), K=1, 3).

4.10 COMMENTS

None

4.11 LISTINGS

The listings of this routine, with BOUDIS and JCOUNT are attached at the end.

4.12 TESTS

This program has been tested in conjunction with SMOB2, a smoothing routine documented in the next section.

COMPILER OPTIONS - NAME= MAIN,DPT=02,LINECNT=56,SIZE=0000K,

SOURCE,EBCDIC,NDLIST,NODECK,LOAD,MAP,NOEDIT,LD,NJXREF

```
ISN 0002      SUBROUTINE BOUDIM( IBDY,NTAPI,NREC,NEL,IRC,ND,NDIS)
              C
              C   FIND DISCONTINUITIES ON BOUNDARIES GIVEN THE INFO. ON NTAPI IN SLIC
              C   FORMAT.
              C
ISN 0003      DIMENSION IBDY(NEL,3),IND(3),NB(3)
ISN 0004      DIMENSION IRC(ND,2)
ISN 0005      DO 10 I=1,3
ISN 0006      IND(I)=I
ISN 0007      10  NB(I)=0
ISN 0008      READ(NTAPI)NB2,(IHDY(I,2),J=1,NB2)
ISN 0009      NB(2)=NB2
ISN 0010      NDIS=0
ISN 0011      DO 20 I=1,NREC
ISN 0012      IF(I.LT.NREC)READ(NTAPI)NB3,(IBDY(J,IND(3)),J=1,NB3)
ISN 0014      IF(I.EQ.NREC)NB3=0
ISN 0016      NB(IND(3))=NB3
ISN 0017      CALL BOUDIS( IBDY,IND,NB,NEL,I,NDIS,IRC,ND)
ISN 0018      DO 30 J=1,3
ISN 0019      30  IND(J)=MOD(IND(J),3)+1
ISN 0020      20  CONTINUE
ISN 0021      RETURN
ISN 0022      END
```

111

REPRODUCIBILITY OF THIS
ORIGINAL PAGE IS FOR

COMPILE OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=0000K,

SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,IO,NJXREF

ISN 0002 SUBROUTINE BOUNDIS(IBDY,IND,NB,NEL,IR,NDIS,IRC,ND)

ISN 0003 DIMENSION IBDY(NEL,3),IND(3),NB(3)

ISN 0004 DIMENSION IRC(IND,3)

C

C IBDY(J,INC(1),J=1,NB(IND(1))) ARE THE BOUNDARY COORDINATES IN THE

C PREVIOUS, CURRENT AND NEXT LINES FOR I=1,2,3 RESPECTIVELY.

C FIND THE DISCONTINUITIES AT THE CURRENT LINE. A DISCONTINUITY

C IS DEFINED AS A BOUNDARY POINT NOT CONNECTED TO AT LEAST TWO OTHER

C BOUNDARY POINTS.

C IT IS ASSUMED THAT THE BOUNDARY POINTS IN EACH ROW ARE IN ASCEND-

C ING ORDER.

C

ISN 0005 NB1=NB(IND(1))

ISN 0006 NB2=NB(IND(2))

ISN 0007 NB3=NB(IND(3))

ISN 0008 IF(NB2, EQ, 0) RETURN

ISN 0010 DO 10 J=1, NB2

ISN 0011 ICOUNT=0

ISN 0012 IF(J.GT.1.AND.IBDY(J,IND(2))-IBDY(J-1,IND(2)).EQ.1)ICOUNT=

ICOUNT+1

ISN 0014 IF(J.LT.NB2.AND.IBDY(J+1,IND(2))-IBDY(J,IND(2)).EQ.1)ICOUNT=

ICOUNT+1

ISN 0016 IF(ICOUNT.GE.2)GO TO 10

ISN 0018 IF(NB1, NE, 0) ICOUNT=

ICOUNT+JCOUNT(IBDY, (IND(2)-1)*NEL+J, (IND(1)-1)*NEL+1, (IND(1)-1)

*NEL+NB1)

ISN 0020 IF(ICOUNT.GE.2)GO TO 10

ISN 0022 IF(NB3, NE, 0) ICOUNT=ICOUNT+

JCOUNT(IBDY, (IND(2)-1)*NEL+J, (IND(3)-1)*NEL+1, (IND(3)-1)*NEL+NB3

)

ISN 0024 IF(ICOUNT.GE.2)GO TO 10

ISN 0026 NDIS=NDIS+1

ISN 0027 WRITE(6,100)NDIS,IR,IBDY(J,IND(2)),ICOUNT

ISN 0028 IRC(NDIS,1)=IR

ISN 0029 IRC(NDIS,2)=IBDY(J,IND(2))

ISN 0030 IRC(NDIS,3)=ICOUNT

ISN 0031 10 CONTINUE

ISN 0032 RETURN

ISN 0033 100 FORMAT(' DISCONTINUITY NO, '15,' AT ('14,' '14,'), ICOUNT='12,'')

ISN 0034 END

COMPILER OPTIMS NAME= MAIN,OPT=02,LINECNT=50,SIZE=000CK,
SJRCE,BCD,NOLIST,NODECK,LOAD,MAP,NOEDIT,LD,NOXREF

```

1SN 0012 ..... FUNCTION JCUNT(IX,J,J1,J2) ..... 00000000
1SN 0013 ..... DIMENSION IX(1) ..... 00000100
C ..... JCUNT= NO. OF VALUES OF JJ SUCH THAT J1.LE.JJ.LE.J2 ..... 00000300
C ..... AND ABS(IX(J)-IX(JJ)).LE.1 ..... 00000400
1SN 0014 ..... JCUNT=1 ..... 00000500
1SN 0015 ..... IF(J1.GT.J2)RETURN ..... 00000600
1SN 0016 ..... K=0 ..... 00000700
1SN 0017 ..... DO 10 JJ=J1,J2 ..... 00000800
1SN 0018 ..... IF(IX(JJ)-IX(J).LT.-1)GO TO 10 ..... 00000900
1SN 0019 ..... IF(IX(JJ)-IX(J).GT.1)GO TO 20 ..... 00001000
1SN 0020 ..... K=K+1 ..... 00001100
1SN 0021 ..... 10 CONTINUE ..... 00001200
1SN 0022 ..... 20 JCUNT=K ..... 00001300
1SN 0023 ..... RETURN ..... 00001400
1SN 0024 ..... END ..... 00001500

```

5. SMOOTHING BOUNDARY DATA

5.1 NAME

SMOB2

5.2 PURPOSE

To patch discontinuities in a digital curve.

5.3 CALLING SEQUENCE

CALL SMOB2(IRC, MDIS, IDIS, NDIS, NDEV, IBDY, IW1, IW2, NREC, K)

where

IRC is an input array dimensioned (MDIS, 3) with IRC(I, 1), IRC(I, 2)) giving the row and column coordinates of the I'th discontinuity and IRC(I, 3) giving its connectivity count for I=1 through NDIS;

IDIS is the discontinuity number at which the patching should be started (only the discontinuities corresponding to I=IDIS through NDIS will be patched);

NDEV is the logical unit number of a direct access device on which the input boundary data set is located; the output after smoothing is written back on NDEV.

IBDY, IW1, IW2 are work arrays to be dimensioned as indicated in the listing attached;

NREC is the number of records in the boundary image;

K is maximum coordinate difference over which the nearest boundary points are checked for patching a discontinuity. (See 5.9, Method).

All parameters except the work arrays are inputs.

5.4 INPUT-OUTPUT

5.4.1 Input

The input data should be on the direct access unit NDEV, consisting of NREC records, the I'th record readable by

READ(NDEV'I)N, (IBDY(J, 1)), J=1, N).

5.4.2 Output

The output data will be on NDEV in the same format as the input.

5.4.3 File Storage

None.

5.5 EXITS

No nonstandard exits.

5.6 USAGE

The program is in FORTRAN IV and is presently implemented on IBM 360 using the H compiler. It is available on the user's library in the form of a load module.

5.7 EXTERNAL REFERENCES

The linkage is indicated in the following table:

Calling Program	Programs Called
SMOB2	PATCH3
PATCH3	SVSCI PATCH1 SORT ELIRPT
PATCH1	CONTEL PATCH4 PATCH2 PRTVEC
SORT	MVMRMR
ELIRPT	VMOV

5.8 PERFORMANCE SPECIFICATIONS

5.8.1 Storage

The size of SMOB2 is 1068 bytes. Including a main program to supply the arrays required to handle a maximum of 2100 boundary points per record with $K=20$ and the buffers, this program needs approximately 114K bytes for execution.

5.8.2 Execution Time

Highly dependent on the image size, complexity and the number of discontinuities to be patched. In the case of the Mobile Bay, Alabama level II GTM which had 4000 records with 728 discontinuities of which about 530 required patches to be generated, the execution time on IBM 360/65 was about 9 minutes. Since there is a considerable amount of I/O involved on the direct access unit NDEV, a significant improvement in execution time can be achieved by using the array read/write routines DARN and DAWN wherever implied DO loops have been used in the subroutine PATCH3.

5.8.3 I/O Load

None

5.8.4 Restrictions

None

5.9 METHOD

The routine SMOB2 simply consists of a DO loop which calls PATCH3 to generate the patch points needed for the L 'th discontinuity and prints the details of the patches produced, with L ranging from IDIS through NDIS.

Consider the routine PATCH3. Suppose (I, J) is the address of the discontinuity at which a patch is to be produced. Then, the records $I-K$ through $I+K$ (bounded, of course, by 1 and NREC) are read from NDEV. While each record is read one row of a $2K+1$ by $2K+1$ binary matrix $IW1$ is defined. The elements of the row are initially set to 0 and whenever the $(J-K+L)$ 'th column in the present row of the input image has a boundary point, the $(L+1)$ st element is set to 1.

After defining $IW1$, the routine PATCH1 is used to check the array $IW1$, eliminate the 1's contiguous with the $(K+1, K+1)$ th element, find the nearest 1 among the remaining and join it to that element by a straight line and store the row and column coordinates of the points so produced in an array $IW2$. Further, if the contiguity count of the point of interest is 0, then the 1's contiguous with

the point joined to the point of interest are also eliminated and a straight line patch is produced to the nearest remaining 1.

The addresses in IW2 are then merged with the data on the input direct access data set by reading the corresponding records of input, sorting the column coordinates in each record using SORT, eliminating repetitions of column coordinates using ELIRPT and writing back on NDEV.

5.10 COMMENTS

The routine SMOB2 can be used in conjunction with BOUDIM or independently. If used independently, the coordinates of discontinuities may be supplied by reading a sequential data set produced by a separate run of BOUDIM. If the program terminates due to lack of time, the execution can be continued by a subsequent run with an updated value of IDIS provided the output data set on NDEV is kept.

5.11 LISTINGS

Listings of SMOB2 and the important routines called by it are shown at the end of this section.

5.12 TESTS

This routine has been tested by using the coordinates of the discontinuities produced by BOUDIM on the Mobile Bay GTM. The first 40 discontinuities were examined in detail by printing the arrays IW1. The performance of the routine was found to be satisfactory.

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR
FOUR

COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=0000K,

SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,LD,NQXREF

```
ISN 0002      SUBROUTINE SMOB2(IRC,MDIS,IDIS,NDIS,NDEV,IBDY,IW1,IW2,NREC,K)
ISN 0003      COMMON/PTCHAD/I1,J1,I2,J2,NP
ISN 0004      DIMENSION IRC(MDIS,2), IBDY(1),IW1(1),IW2(1)
              C   D*N IBDY(MAX. EXPECTED NO. OF BOUNDARY POINTS IN A LINE AFTER SMOOTHING)
              C
              C   D*N IW1(K21**2),IW2(K21**2) WHERE K21=2*K+1.
              C
ISN 0005      DO 20 I=IDIS,NDIS
ISN 0006      CALL PATCH3(IBDY,IRC(I,1),IRC(I,2),IRC(I,3),K,IW1,IW2,NREC,NDEV)
ISN 0007      IF(I2.NE.0)PRINT 100,I,IRC(I,1),IRC(I,2),I1,J1,I2,J2
ISN 0009      IF(I1.NE.0.AND.I2.EQ.0)PRINT 101,I,IRC(I,1),IRC(I,2),I1,J1
ISN 0011      IF(I1.EQ.0)PRINT 102,I,IRC(I,1),IRC(I,2)
ISN 0013      20  CONTINUE
ISN 0014      RETURN
ISN 0015      100  FORMAT(2X15,' : ('15','15,') JOINED TO ('15','15,') AND ('15','15,')
              .    15,')')
ISN 0016      101  FORMAT(2X15,' : ('15','15,') JOINED TO ('15','15,')')
ISN 0017      102  FORMAT(2X15,' : NO PATCH POINTS PRODUCED AT ('15','15,')')
ISN 0018      END
```

COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=0000K,

SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NODEIT,LD,NOXREF

ISN 0002

SUBROUTINE CONTEL(IW1,IW2,M,N)

ISN 0003

DIMENSION IW1(M,N),IW2(2,1)

C

C

C

C

C

C

THIS PROGRAM ELIMINATES ALL 1'S IN IW1 CONNECTED TO THE 1 AT
(IW1(1,1),IW2(2,1)). IW2 SHD BE DIMENSIONED (2,K) WHERE K IS TWICE
THE NUMBER OF NODES IN THE PIECE OF DIGITAL CURVE CONNECTED TO THE
POINT OF INTEREST.

ISN 0004

K=1

ISN 0005

L=1

ISN 0006

40

DO 20 KK=1,K

ISN 0007

I=IW2(1,KK)

ISN 0008

J=IW2(2,KK)

ISN 0009

IW1(I,J)=0

ISN 0010

I1=MAX0(I-1,1)

ISN 0011

I2=MIND(I+1,M)

ISN 0012

J1=MAX0(J-1,1)

ISN 0013

J2=MIND(J+1,N)

ISN 0014

DO 10 II=I1,I2

ISN 0015

DO 10 JJ=J1,J2

ISN 0016

IF(IW1(II,JJ).EQ.0)GO TO 10

ISN 0018

L=L+1

ISN 0019

IW2(1,I)=II

ISN 0020

IW2(2,L)=JJ

ISN 0021

IW1(II,JJ)=0

119

ISN 0022

10

CONTINUE

ISN 0023

20

CONTINUE

ISN 0024

IF(L.EQ.K)RETURN

ISN 0026

K1=K+1

ISN 0027

LL=0

ISN 0028

DO 30 KK=K1,K

ISN 0029

LL=LL+1

ISN 0030

IW2(1,LL)=IW2(1,KK)

ISN 0031

30

IW2(2,LL)=IW2(2,KK)

ISN 0032

K=LL

ISN 0033

L=LL

ISN 0034

GO TO 40

ISN 0035

END

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=0000K,

SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,LD,NOXREF

```

: ISN_0002      SUBROUTINE_PATCH1(IW1,IW2,M,N,I,J,ICOUNT)
ISN 0003      COMMON/PTCHAD/I1,J1,I2,J2,NP
ISN 0004      DATA IPASS/0/
ISN 0005      IPASS=IPASS+1
ISN 0006      IF(IPASS.LE.40)CALL PRVVEC(IW1,M,N,I)
C
C      GENERATE PATCH POINTS IN IW1 STARTING FROM (I,J) TO THE NEAREST
C      (2-ICOUNT) NONCONTIGUOUS NEIGHBORS.
C
C      SEE CONTEL FOR DIMENSIONING INFO FOR IW2.
C
ISN 0008      DIMENSION IW1(M,N),IW2(2,1)
ISN 0009      IW2(1,1)=I
ISN 0010      IW2(2,1)=J
C
C      ELIMINATE POINTS CONTIGUOUS WITH (I,J).
C
ISN 0011      CALL CONTEL(IW1,IW2,M,N)
C
C      FIND NEAREST NEIGHBOR OF (I,J) WHICH IS SET.
C
ISN 0012      I2=0
ISN 0013      CALL PATCH4(IW1,M,N,I,J,I1,J1)
C      NOW (I1,J1) IS THE NEAREST NEIGHBOR.
C
120 ISN 0014      IF(ICOUNT.NE.0.OR.I1.EQ.0)GO TO 10
C
C      ELIMINATE POINTS CONTIGUOUS WITH (I1,J1)
C
ISN 0016      IW2(1,1)=I1
ISN 0017      IW2(2,1)=J1
ISN 0018      CALL CONTEL(IW1,IW2,M,N)
ISN 0019      CALL PATCH4(IW1,M,N,I,J,I2,J2)
C      NOW (I2,J2) IS THE NEXT NEAREST NEIGHBOR.
ISN 0020      10 CONTINUE
ISN 0021      MP=0
ISN 0022      NP=0
ISN 0023      IF(I1.EQ.0)RETURN
C
C      PRODUCE PATCH ADDRESSES IN IW2.
C
ISN 0025      CALL PATCH2(IW2,NP,I1,J1,I,J)
ISN 0026      IF(I2.NE.0)CALL PATCH2(IW2(1,NP+1),MP,I2,J2,I,J)
ISN 0028      NP=NP+MP
ISN 0029      IF(IPASS.LE.40)CALL PRVVEC(IW2,2*NP,2)
ISN 0031      RETURN
ISN 0032      END

```


COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=0000K,

SCURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,NJXREF

```

ISN 0002      SUBROUTINE PATCH2(IW,NP,I1,J1,I2,J2)
ISN 0003      DIMENSION IW(2,1)
C
C      TO GENERATE COORDINATES OF LINE JOINING (I1,J1) AND (I2,J2).
C      DIMENSION IW(2,MP) WHERE MP= MAX. NO. OF POINTS EXPECTED TO BE
C      PRODUCED. NP= NO. OF POINTS ACTUALLY PRODUCED BY THIS ROUTINE.
C
ISN 0004      IMA=MINO(I1,I2)+1
ISN 0005      IMX=MAXO(I1,I2)-1
ISN 0006      JMN=MINO(J1,J2)+1
ISN 0007      JMX=MAXO(J1,J2)-1
ISN 0008      I12=I1-I2
ISN 0009      J12=J1-J2
ISN 0010      RI12=I12
ISN 0011      RJ12=J12
ISN 0012      NP=0
ISN 0013      IF(IMX-IMN.GT.JMX-JMN)GO TO 10
ISN 0015      IF(JMN.GT.JMX)RETURN
ISN 0017      DO 20 J=JMN,JMX
ISN 0018      I=I1+(J-J1)*I12/RJ12+.5
ISN 0019      NP=NP+1
ISN 0020      IW(1,NP)=I
ISN 0021      20  IW(2,NP)=J
ISN 0022      RETURN
C
ISN 0023      10  IF(IMN.GT.IMX)RETURN
ISN 0025      DO 30 I=IMN,IMX
ISN 0026      NP=NP+1
ISN 0027      J=J1+(I-I1)*J12/RI12+.5
ISN 0028      IW(1,NP)=I
ISN 0029      30  IW(2,NP)=J
ISN 0030      RETURN
ISN 0031      END

```

COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=0000K,

SOURCE,EBCDIC,NOLIST,NODECK,LOAD,PAP,NOEDIT,ID,NOXREF

```

ISN 0002      SUBROUTINE PATCH3(IBDY,I,J,ICOUNT,K,IW1,IW2,NREC,NDEV)
ISN 0003      COMMON/PTCHAD/I1,J1,I2,J2,NP
              C DIMENSION IW1(K21**2),IW2(2,MP), WHERE MP=MAX(ND, OF PATCH POINTS
              C EXPECTED TO BE GENERATED BY PATCH2, DIMENSION REQUIRED BY CONTEL)
              C PATCH DISCONTINUITY AT I,J.
ISN 0004      DIMENSION IW1(1),IW2(2,1),IBDY(1)
ISN 0005      K21=K*2+1
ISN 0006      K1=MAX0(I-K,1)
ISN 0007      K2=MIN0(I+K,NREC)
ISN 0008      KK21=K2-K1+1
ISN 0009      CALL SVSC1(IW1,KK21*K21,0)
ISN 0010      ICLMMK=J-K
ISN 0011      ICLMPK=J+K
ISN 0012      DO 10 KK=K1,K2
ISN 0013      READ(NDEV,KK)N,(IBDY(L),L=1,N)
ISN 0014      IF(N.EQ.0)GO TO 10
ISN 0016      DO 20 L=1,N
ISN 0017      IF(IBDY(L).LT.ICLMMK)GO TO 20
ISN 0019      IF(IBDY(L).GT.ICLMPK)GO TO 10
              C
              C TREAT IW1 AS A 2-D ARRAY DIMENSIONED KK21*K21 WITH THE GIVEN POINT
              C AT (I-K1+1,K+1)TH LOCATION.
              C
ISN 0021      IW1((IBDY(L)-ICLMMK)*KK21+KK-K1+1)=I
ISN 0022      20 CONTINUE
122 ISN 0023      10 CONTINUE
              C
              C GENERATE PATCH ADDRESSES IN IW2.
              C
ISN 0024      CALL PATCH1(IW1,IW2,KK21,K21,I-K1+1,K+1,ICOUNT)
              C
              C MERGE ADDRESSES FOUND IN IW2 WITH THE BOUNDARY ADDRESSES ON DISC.
              C
ISN 0025      IF(NP.EQ.0)RETURN
ISN 0027      IP=1
ISN 0028      IP1=1
ISN 0029      30 IP=IP+1
              C
              C FIND NEXT CHANGE IN IW2(I,*)
              C
ISN 0030      IF(IP.GT.NP)GO TO 40
ISN 0032      IF(IW2(1,IP).EQ.IW2(1,IP-1))GO TO 30
ISN 0034      40 IP2=IP-1
ISN 0035      KK=IW2(1,IP2)+K1-1
ISN 0036      READ(NDEV,KK)N,(IBDY(L),L=1,N)
ISN 0037      DO 50 JP=IP1,IP2
ISN 0038      N=N+1
ISN 0039      50 IBDY(N)=IW2(2,JP)+ICLMMK-1
ISN 0040      CALL SORT(IBDY,1,N,N,1,T,TT)
ISN 0041      CALL ELIRPT(N,IBDY)
ISN 0042      WRITE(NDEV,KK)N,(IBDY(L),L=1,N)
ISN 0043      IP1=IP

```

ISN 0044 IF(IP,LE,NP)GO TO 30
ISN 0046 RETURN
ISN 0047 END

123

COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=0000K,

SOURCE, EBCDIC, NOLIST, NODCHECK, LOAD, MAP, NOEDIT, ID, NOXREF

```
ISN 0002      SUBROUTINE PATCH4(IW1,M,N,L,J,IO,JO)
ISN 0003      DIMENSION IW1(M,N)
ISN 0004      IO=0
ISN 0005      JO=0
ISN 0006      IF(L.EQ.0)RETURN
ISN 0008      IDMIN=M**2+N**2+1
ISN 0009      DO 10 II=1,M
ISN 0010      DO 10 JJ=1,N
ISN 0011      IF(IW1(II,JJ).EQ.0)GO TO 10
ISN 0013      ID=(II-I)**2+(JJ-J)**2
ISN 0014      IF(ID.GE.IDMIN)GO TO 10
ISN 0016      IO=II
ISN 0017      JO=JJ
ISN 0018      IDMIN=ID
ISN 0019      10 CONTINUE
ISN 0020      RETURN
ISN 0021      END
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

COMPILER OPTIONS - NAME= MAIN,DPT=02,LINECNT=56,SIZE=0000K,
SOURCE,EBCDIC,NOLIST,NJOB=CK,LOAD,MAP,NOEDIT,IO,NJXREF

```
ISN 0002      SUBROUTINE PRTEC(IX,N,IFMT)
ISN 0003      DIMENSION IX(N)
ISN 0004      IF(IFMT.EQ.1)PRINT 100,IX
ISN 0006      IF(IFMT.EQ.2)PRINT 200,IX
ISN 0008      RETURN
ISN 0009      100  FORMAT(10X4I11)
ISN 0010      200  FORMAT(1X40I3)
ISN 0011      END
```

6. IDENTIFICATION OF CONNECTED REGIONS

6.1 NAME

REGIONS

6.2 PURPOSE

To identify all distinct connected regions in an image given the boundary data in SLIC format and produce 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.

6.3 CALLING SEQUENCE

This is a main program. In its present version the image size is supplied through DATA statements.

6.4 INPUT-OUTPUT

6.4.1 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.

6.4.2 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.

6.4.3 File Storage

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

6.5 EXITS

Not applicable

6.6 USAGE

This program is in FORTRAN IV and is implemented on IBM 360 with the H compiler. The associated subroutines are available as load modules on the user's library. The deck for the main program is available with the authors and needs only slight modifications in the DEFINE FILE and DATA statements for use on any data set.

6.7 EXTERNAL INTERFACES

This program uses several subroutines as indicated by the linkage table below:

Calling Program	Programs Called
REGIONS	PET VMAXI4 VMINI4 RIDER SVSCI DARN SEQLS SAWN
RIDER	SVSCI2 SVSCL1 SORT RIDER1 RIDER4 DAWN VMOV2 RIDER2 PRTVE2 DARN VMAXI2
SEQLS	SORT FLIPV
SORT	MVMRMR
RIDER1	SVSCI2
RIDER4	RIDER5 SVSCI2 PRTVE2 RIDER7 RIDER6 SVSCL1

Calling Program	Programs Called
RIDER2	PRTVE2
RIDER7	VMAXI2 VMINI2

6.8 PERFORMANCE SPECIFICATIONS

6.8.1 Storage

The present version of the main program is 134,436 bytes long. The external references required and the buffers increase this to 192K bytes. However, the size is dependent on the data set to be handled and the dimension statements should be changed to satisfy specific requirements.

```

DIMENSION IX(2NR+2, N), IRES(MSEG+1)
INTEGER*2 IW1(NEL), IW2(NEL), ITABL(MR*MSEG), IS(MR)
INTEGER*2 LW(MR)
LOGICAL*1 IDENT(MR, MR)

```

where

```

NR    = Maximum number of regions expected;
N     = Maximum number of boundary points expected in a record;
MSEG  = Maximum number of "segments" required to handle the
        image (see 6.9, Method);
MR    = Maximum number of regions identifiers permissible in a
        segment;
NEL   = Number of pixels per line in the output map.

```

6.8.2 Execution Time

The time is highly dependent on the size and complexity of the image. The Mobile Bay GTM (level II) resulting in a region identification map with 400x2100 pixels and consisting of 1742 regions had to be handled in 15 sections and took 19.5 minutes of CPU time on IBM 360/65.

6.8.3 Restrictions

None

6.9 METHOD

This program has five major sections.

- (i) Determination of the bounds on the column coordinates of boundaries on the input data set;
- (ii) Finding a preliminary set of region identifiers;
- (iii) Finding the areas of each of the regions;
- (iv) Generating a mapping such that the region numbers are used in the order of decreasing areas;
- (v) Modifying the region numbers by table look-up.

6.9.1 Determination of Bounds

The maximum and minimum values of the column coordinates of the boundary points are determined. If the minimum is greater than 1, it is set to 1. If the maximum is less than the value of NEL supplied, it is set to NEL. The value of NEL is then changed to Max-Min+1. The output image size will then be NREC by NEL.

6.9.2 Finding Preliminary Region Identifiers

This is the most important step in the program. The subroutine RIDER is used for this purpose. Its function is similar to the routine with the same name described in [4]. The routine in [4] was designed to print an error message and return with NR=0 when the number of distinct regions exceeded MR. But the present version can handle up to MR*MSEG distinct regions while still using a "region identity matrix" of size MR by MR (rather than MR*MSEG by MR*MSEG).

This routine uses the arrays IW1 and IW2 as the previous and current records of region identifiers. By convention, region numbers 1 and 0 indicate the "exterior" of the image and boundary points. The MR by MR array IDENT is used to store information about identity of regions, IDENT(I,J) = .TRUE. meaning that region numbers I and J refer to the same connected region.

Initially, 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.

The boundary coordinates in the input record are arranged in ascending order. The routine RIDER1 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 there is no non-zero element in that segment of IW1, a new region number is started and assigned to the interval in IW2. If there is a nonzero element, that number is filled into 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 RIDER1 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 RIDER.

Now, if RIDER1 returns a positive NR, the array IW2 is written as the I'th record on the direct access data set (unit number IDEVO in RIDER, same as 90 in the main program) and IW2 is moved into IW1 (so that it becomes the "previous" record while handling the next record).

If RIDER1 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 look-up 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 1 to consecutive numbers starting with 2; 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 look-up 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 record 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 RIDER prints an error message, sets NR=0 and exits.

Otherwise, RIDER1 is called again, IW2 is found and written on IDEVO 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 look-up table for the final segment. A call to RIDER2 changes the look-up tables for all the segments such that consecutive region numbers are used.

Finally, each record from IDEVO is read, the appropriate look-up table is used to modify it and the record is written back on IDEVO. Also, NR is set to the maximum region number used after table look-up.

6.9.3 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.

6.9.4 Finding the Final Look-up 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 SEQLS). 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 look-up table. The actual coding follows these principles but is slightly different in detail to preserve the identities of regions 0 and 1 which have special significance.

6.9.5 Deriving the Final Region Identification Map

The look-up table generated above is used to modify the region identifiers on IDEVO, record by record, and write out the final sequential data set on unit 12.

6.10 COMMENTS

An approach suggested in [5] can 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.

6.11 LISTINGS

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

6.12 TESTS

This program has been tested on the Mobile Bay GTM both before and after smoothing and found to work satisfactorily. Also, the results have been found to be identical (on a smaller data set) with those obtained by the earlier version of this program.

COMPILER OPTIONS - NAME= MAIN,UPT=00,LINECNT=56,SIZE=0000K,

SOURCE,EBCDIC,LIST,NUDECK,LOAD,MAP,NUEDIT,ID,NJXREF

```

ISN 0002 DIMENSION IX(4000),IRES(21)
ISN 0003 INTEGER*2 IW1(2100),IW2(2100),ITABL(8000),IS(400)
ISN 0004 INTEGER*2 LW(400)
ISN 0005 LOGICAL*1 IDENT(300,300)
ISN 0006 DEFINE FILE 9C(4000,4200,L,IAV)

C
C D.N.IX(MAX(2NR+2,N) WHERE NR=MAX. NO. OF REGIONS EXPECTED AND
C N= MAX NO. OF BOUNDARY POINTS EXPECTED IN ANY RECORD)
C

ISN 0007 DATA NREC,MR,MSEG/4000,300,20/
ISN 0008 DATA NEL/2100/
ISN 0009 MAXX=-1000000
ISN 0010 MINX= 1000000
ISN 0011 CALL STRTMR
ISN 0012 CALL PET(0)
ISN 0013 DO 10 I=1,NREC
ISN 0014 READ(8)N,(IX(J),J=1,N)
ISN 0015 IF(N.EQ.0)GO TO 10
ISN 0017 CALL VMAXI4(IX,N,MAXX)
ISN 0018 CALL VMINI4(IX,N,MINX)
ISN 0019 10 CONTINUE
ISN 0020 REWIND 8
C IDENTIFY CONNECTED REGIONS.
ISN 0021 PRINT 600,MINX,MAXX,NEL
ISN 0022 MINX=MIN0(MINX,1)
ISN 0023 PRINT 600,MINX,MAXX,NEL
ISN 0024 MAXX=MAX0(MAXX,NEL)
ISN 0025 PRINT 600,MINX,MAXX,NEL
ISN 0026 NEL=MAXX-MINX+1
ISN 0027 PRINT 600,MINX,MAXX,NEL
ISN 0028 600 FORMAT(' MINX,MAXX,NEL='3I8)
ISN 0029 PRINT 100,NREC,NEL
ISN 0030 100 FORMAT(//' IMAGE SIZE=( '15,' '15,' )' )
ISN 0031 NDUM=1
ISN 0032 PRINT 1000,NDUM
ISN 0033 1000 FORMAT(' NDUM='15)
ISN 0034 CALL PET(2)
ISN 0035 CALL RIDER(8,NREC,NEL,9C,MINX,IX,IW1,IW2,ITABL,IDENT,MR ,NR,LW,
MSEG,IRES,IS)
ISN 0036 CALL PET(2)
C
C FIND AND PRINT HISTOGRAM OF REGION IDENTIFICATION MAP.
C
ISN 0037 PRINT 200
ISN 0038 200 FORMAT(//10X'REGION NO. '10X'NO. OF PIXELS')
ISN 0039 CALL SVSCI(IX,NR+1,0)
ISN 0040 DO 30 I=1,NREC
ISN 0041 CALL DARN(90,I,IW1,NEL*2)
ISN 0042 DO 30 IEL=1,NEL
ISN 0043 J=IW1(IEI)+1
ISN 0044 30 IX(J)=IX(J)+1
ISN 0045 NR1=NR+1

```

REPRODUCIBILITY OF THE
 ORIGINAL PAGE IS POOR


```

ISN 0046      DU 40 I=1,NR1
ISN 0047      J=1-1
ISN 0048      IF(IX(I).NE.0)PRINT 300,J,IX(I)
ISN 0050      40  CONTINUE
ISN 0051      300  FORMAT(11X16,16X19)
ISN 0052      CALL PET(2)
C
C  REARRANGE NUMBERS IN DESCENDING ORDER OF POPULATIONS.
C  LEAVE 0 AND 1 UNCHANGED SINCE THEY CORRESPOND TO EXTERIOR AND
C  BOUNDARY POINTS RESPECTIVELY.
C
ISN 0053      CALL SEQLS(IX(3),IX(NR1+1),NR-1,NR-1)
ISN 0054      PRINT 400
ISN 0055      400  FORMAT('1 REGIONS AFTER REASSIGNMENTS:')
ISN 0056      PRINT 200
ISN 0057      DO 50 I=1,NR1
ISN 0058      J=1-1
ISN 0059      IF(I.LE.2)PRINT 300,J,IX(I)
ISN 0061      IF(I.GT.2)PRINT 350,J,IX(I),IX(I+NR-1)
ISN 0063      50  CONTINUE
ISN 0064      350  FORMAT(11X16,16X19,17X16)
ISN 0065      IX(1)=0
ISN 0066      IX(2)=1
ISN 0067      DO 60 I=3,NR1
ISN 0068      60  IX(IX(I+NR-1)+2)=I-1
ISN 0069      CALL PET(2)
C
C  MODIFY REGION NUMBERS ACCORDING TO NEW ASSIGNMENTS FOUND IN IX.
C
ISN 0070      DO 70 I=1,NREC
ISN 0071      CALL DARN(90,I,IW1,NEL*2)
ISN 0072      DO 80 IEL=1,NEL
ISN 0073      J=IW1(IEL)+1
ISN 0074      80  IW1(IEL)=IX(J)
ISN 0075      70  CALL SAWN(12,IW1,NEL*2)
ISN 0076      CALL PET(2)
ISN 0077      STOP
ISN 0078      END

```

SUBROUTINE-DANN(IDEV,IREC,X,N)

LOGICAL*1 X(N)

READ(IDEV,IREC)X

RETURN

ENTRY DANN(IDEV,IREC,X,N)

WRITE(IDEV,IREC)X

RETURN

ENTRY DANN(NTAPE,X,N)

READ(NTAPE)X

RETURN

ENTRY DANN(NTAPE,X,N)

WRITE(NTAPE)X

RETURN

END

```
SUBROUTINE FLIPV(X,N,Y)  
DIMENSION X(N),Y(N)
```

```
C
```

```
C
```

```
C
```

```
      EQ'CE(X,Y)
```

```
      N2=N/2
```

```
      N1=N+1
```

```
      DO 10 I=1,N2
```

```
      W=X(I)
```

```
      I1=N1-I
```

```
      Y(I)=X(I1)
```

```
      Y(I1)=W
```

```
10
```

```
      CONTINUE
```

```
      RETURN
```

```
      END
```

LEVEL 21.1 (JAN-73)

05/360 FORTRAN-H

DATE 74.288/11.57.54

COMPILER-OPTIONS NAME=MAIN,OPT=02,LINECNT=56,SIZE=GEORG,
SOURCE,DCU,NOLIST,NODECK,LOAD,HAP,NOEDIT,IO,NOXREF

15N 0002 SUBROUTINE MVNKKK(A,MA,N,B,MB,IA,IB)
15N 0003 DIMENSION A(MA,N),L(MB,N)
15N 0004 DO 10 J=1,N
15N 0005 10 B(IA,J)=A(IA,J)
15N 0006 RETURN
15N 0007 END

137

```
      SUBROUTINE PET(I)
      IF(I.NE.0)GO TO 10
      CALL TIMER(ITIME1)
      TTIME=0.
      WRITE(6,200)
200  FORMAT(10X,'BEGINNING TIMING*** TIME NOW IS 0')
      RETURN
10  CALL TIMER(ITIME2)
      TIME=(ITIME2-ITIME1)/100.
      TTIME=TTIME+TIME
      ITIME1=ITIME2
      WRITE(6,100)TIME,TTIME
100  FORMAT(10X'TIME ELAPSED SINCE LAST PRINTING OF TIME='E12.3,
      'SEC., TOTAL TIME ELAPSED='E12.3,'SEC.')
```

```

SUBROUTINE PRIVE2(IX,N)
  INTEGER*2 IX(N)
  PRINT 100,IX
100  FORMAT(1X25I5)
  RETURN
  END
```

```

SUBROUTINE RIDER(NTAPB,NREC,NEL,IDEVO,ICMN,IBDY,IW1,IW2,ITABL,
IDENT,MR,NR,LW,MSEG,IRES,IS)

```

```

C
C TO IDENTIFY ALL DISTINCT CONNECTED REGIONS IN A PICTURE SEPARATED
C BY BOUNDARY LINES. THE BOUNDARY DATA ARE GIVEN AS NREC RECORDS
C SEQUENTIAL FILE NTAPB, EACH RECORD BEING WRITTEN AS
C N,(IBDY(I),I=1,N)
C THE OUTPUT OF THE PROGRAM IS AN NREC*NEL DIRECT ACCESS FILE ON
C IDEVO CONSISTING OF 0'S FOR BOUNDARY POINTS AND DISTINCT REGION
C NUMBERS FOR EACH OF THE CONNECTED REGIONS. ICMN= MINIMUM COLUMN
C NUMBER WHICH, ON THE OUTPUT FILE, WILL CORRESPOND TO THE FIRST
C COLUMN. IT IS NECESSARY THAT
C ICMN.LE.MIN(IBDY).LE.MAX(IBDY).LE.ICMN+NEL-1.
C DEFINE FILE IDEVO(NREC,NEL*2,L,IAV)
C

```

```

DIMENSION IBDY(1)
LOGICAL*1 IDENT(MR,MR)
LOGICAL * 1 LW(MR,MR)
INTEGER*2 IW1(NEL),IW2(NEL),ITABL(MR,MSEG),IS(1)
DIMENSION IRES(MSEG)

```

```

C
C INITIALIZE A WORK ARRAY IW1 WITH 1'S AND IDENT WITH .FALSE.
C

```

```

CALL SVSCI2(IW1,NEL,1)
CALL SVSCL1(IDENT,MR*MR,.FALSE.)
ISEG=1
IRES(1)=1
NR=1

```

```

C
C LOOP ON RECORDS
C

```

```

DO 10 IREC=1,NREC

```

```

C
C READ ONE RECORD OF BOUNDARY INFO.
C

```

```

READ(NTAPB)N,(IBDY(I),I=1,N)
IF(N.NE.0)CALL SORT(IBDY,1,N,N,1,T,TT)

```

```

C
C USE IW1 AND IBDY TO SET ARRAY IW2 AND MATRIX IDENT
C

```

```

30 CONTINUE

```

```

CALL RIDER1(IW1,IBDY,ICMN,NEL,N,IW2,IDENT,MR,NR)

```

```

IF(NR.GT.0)GO TO 20

```

```

PRINT 200, IREC,NR

```

```

200 FORMAT(' (IREC, NR) = '2I6)

```

```

IF(IREC.EQ.IRES(ISEG))GO TO 40

```

```

NR=-NR

```

```

CALL RIDER4(IDENT,LW,MR,NR,ITABL,ISEG,IW1,NEL,IS,.FALSE.)

```

```

ISEG=ISEG+1

```

```

IF(ISEG.GT.MSEG)GO TO 40

```

```

IRES(ISEG)=IREC

```



```

      IF(NR.LE.MR)GO TO 30
40  PRINT 100,IREC
      NR=0
100  FORMAT(' ERROR CONDITION IN RIDER. SUPPLIED MR OR MSEG WAS EXCEED
      .ED AT RECORD NUMBER 'I6, '. RETURNING WITH NR=0')
      RETURN
      20  CALL DAWN(IDEVO,IREC,IW2,NEL*2)
      CALL VMOV2(IW2,NEL,IW1)
      10  CONTINUE
      CALL RIDER4(IDENT,LW,MR,NR,ITABL,ISEG,IW1,NEL,IS, .TRUE.)
      CALL RIDER2(ITABL,MR*ISEG)
      IRES(ISEG+1)=NREC+1
      PRINT 300
300  FORMAT('///' FINAL TABLES FOR MODIFYING REGION NUMBERS')
      DO 60 JSEG=1,ISEG
      PRINT 400,JSEG
      400  FORMAT('0 SEGMENT NUMBER 'I3)
      60  CALL PRTVE2(ITABL(I,JSEG),MR)
      JSEG=1
      DO 70 IREC=1,NREC
      IF(IREC.LT.IRES(JSEG+1))GO TO 50
      JSEG=JSEG+1
      50  CONTINUE
      CALL DAWN(IDEVO,IREC,IW1,NEL*2)
      DO 80 IEL=1,NEL
      I=IW1(IEL)
      IF(I.NE.0)IW1(IEL)=ITABL(I,JSEG)
      80  CONTINUE
      70  CALL DAWN(IDEVO,IREC,IW1,NEL*2)
      NR=0
      CALL VMAXI2(ITABL,MR*ISEG,NR)
      RETURN
      END

```

COMPILER OPTIONS NAME= MAIN,OPT=CZ,LINELN1=56,SIZE=800,K,
SOURCE,BCD,NOLIST,HODECK,LEAD,MAP,NCEDIT,IO,NOXREF

```

1SN 0002      SUBROUTINE RTDR1 (IX,IBDY,ICMN,NEL,N,IY,IUENI,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
1SN 0003      DIMENSION IBDY(N)
1SN 0004      INTEGER*2 IX(NEL),IY(NEL)
1SN 0005      LOGICAL*1 IDENT(MR,MR)
1SN 0006      IF (N.EQ.0) GO TO 10
1SN 0008      CALL SVSCI2(IY,NEL,IY)
1SN 0009      GO TO 20
1SN 0010      10  CONTINUE
1SN 0011      MR=NR
      C
      C   ALL POINTS TO THE LEFT OF IBDY(1) ARE 'EXTERIOR' POINTS (REGION 1)
      C
1SN 0012      I=IBDY(1)-ICMN
1SN 0013      IF (I.GT.0) CALL SVSCI2(IY,I,I)
      C
      C   ALL POINTS TO THE RIGHT OF IBDY(N) ARE 'EXTERIOR' POINTS.
      C
1SN 0015      I=IBDY(N)-ICMN+2
1SN 0016      IF (NEL.GE.1) CALL SVSCI2(IY(I),NEL-I+1,I)
      C
      C   DESIGNATE ALL BOUNDARY POINTS AS 'REGION 0'.
      C
1SN 0018      DO 30 I=1,N
1SN 0019      J=IBDY(I)-ICMN+1
1SN 0020      30  IY(J)=0
      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.
      C
1SN 0021      IF (N.EQ.1) GO TO 20
1SN 0023      N1=N-1
1SN 0024      DO 40 I=1,N1
1SN 0025      L1=IBDY(I)-ICMN+2
1SN 0026      L2=IBDY(I+1)-ICMN
1SN 0027      IF (L1.GT.L2) GO TO 40
1SN 0029      DO 50 L=L1,L2
1SN 0030      IF (IX(L).EQ.0) GO TO 50
1SN 0032      LL=L
1SN 0033      GO TO 60
1SN 0034      50  CONTINUE
1SN 0035      NR=NR+1
1SN 0036      IF (MR.LE.MR1) GO TO 70
1SN 0038      MR=NR
1SN 0039      RETURN

```

```

ISN 0040 70 CALL SVSC12(IY(L1),L2=L1+1,NR)
ISN 0041 IDENT(NR,NR)=.TRUE.
ISN 0042 GO TO 4F
ISN 0043 60 LL=IX(ILL)
ISN 0044 CALL SVSC12(IY(L1),L2=L1+1,LL)
ISN 0045 40 CONTINUE
ISN 0046 20 CONTINUE
C
C SET IDENT MATRIX TO INDICATE REGION NUMBERS CORRESPONDING TO
C IDENTICAL REGIONS.
C
ISN 0047 DO 80 IEL=1,NEL
ISN 0048 I=IX(IEL)
ISN 0049 IF(I.EQ.0)GO TO 80
ISN 0051 J=IY(IEL)
ISN 0052 IF(J.EQ.0)GO TO 80
ISN 0054 IDENT(I,J)=.TRUE.
ISN 0055 80 CONTINUE
ISN 0056 RETURN
ISN 0057 END

```

```

SUBROUTINE RIDER2(ITABL,NR)
C
C CHANGE REGION NUMBERS SUCH THAT CONSECUTIVE NUMBERS ARE USED.
C
C INTEGER*2 ITABL(NR,2)
C
C FIND THE SET OF NUMBERS IN ITABL(*,1).
C
  DO 5 I=1,NR
5    ITABL(I,2)=0
    DO 10 J=1,NR
    J=ITABL(I,1)
10   IF(J.NE.0)ITABL(J,2)=ITABL(J,2)+1
    PRINT 1C0
    CALL PRV2(ITABL(1,2),NR)
C
C CHANGE ITABL(*,2) TO GET A LOOKUP TABLE FOR ITABL(*,1).
C
  J=0
  DO 20 I=1,NR
  IF(ITABL(I,2).EQ.0)GO TO 20
  J=J+1
  ITABL(I,2)=J
20  CONTINUE
  PRINT 2C0
  CALL PRV2(ITABL(1,2),NR)
C
C CHANGE ITABL(*,1).
C
  DO 30 I=1,NR
30  ITABL(I,1)=ITABL(ITABL(I,1),2)
  RETURN
100 FORMAT('0NUMBERS OF OCCURENCES OF REGION NUMBERS IN TH ABOVE TABLE
. (S)')
200 FORMAT('0LOOK-UP TABLE TO CHANGE NUMBERS IN THE ABOVE TABLE(S)')
END

```

```

SUBROUTINE RIDER4(IDENT,LW,MR,NR,ITABL,ISEG,IW1,NEL,IS, LAST)
LOGICAL LAST
LOGICAL*1 IDENT(MR,MR)
INTEGER*2 ITABL(MR,1),LW(MR),IS(1),IW1(NEL)

```

```

C
C   D'N ITABL(MR,ISEG),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.
C
C   THIS ROUTINE IS CALLED FROM RIDER WHEN ALL RECORDS ARE PROCESSED
C   (LAST=.TRUE.) OR WHEN THE NUMBER OF REGION NUMBERS FOUND WHILE TEST-
C   ING (IREC+1)'TH RECORD EXCEEDS MR. (LAST=.FALSE.).
C   THEN,
C       1. THE REGION CONNECTIVITY MATRIX IDENT IS REDUCED TO GET A LOOK
C       UP 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 OCCURING IN THE IREC'TH RECORD
C       ARE FOUND; A CORRESPONDENCE ARRAY IS BETWEEN CURRENT AND NEXT SEGMENT
C       SET UP. THE LAST RECORD(IW1) IS MODIFIED TO MATCH THE NUMBERING
C       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.

```

SECTION 1.

```

C
C   DO 50 I=1,NR
C   DO 50 J=1,NR
50  IDENT(I,J)=IDENT(I,J).OR.IDENT(J,I)
C   CALL RIDER5(IDENT,MR,NR,ITABL(1,ISEG),LW)
C   ISEGT=(ISEG-1)*MR
C   DO 10 I=1,NR
10  IF(ITABL(I,ISEG).GT.1)ITABL(I,ISEG)=ITABL(I,ISEG)+ISEGT
C   IF(MR.GT.NR)CALL SVSCI2(ITABL(NR+1,ISEG),MR-NR,0)
C   PRINT 100,ISEG
C   CALL PRTVE2(ITABL(1,ISEG),MR)

```

SECTION 2.

```

C
C   IF(ISEG.EQ.1)GO TO 60
C   ISEG1=ISEG-1
C   CALL RIDER7(ITABL(1,ISEG),IS,NCR,ITABL,MR*ISEG1)
C   DO 40 JSEG=1,ISEG1
C   KSEG=ISEG-JSEG
C   PRINT 200,KSEG
40  CALL PRTVE2(ITABL(1,KSEG),MR)

```

SECTION 3.

```

C
C   60  IF(LAST)RETURN
C   CALL RIDER6(IW1,NEL,IS,NR)

```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```

NCR=NR
PRINT 300, NR
CALL PRV2 (IS, NR)
DO 70 I=1, NR
70 IS(I)=ITABL (IS(I), ISEG)
PRINT 400
CALL PRV2 (IS, NR)

C
C          SECTION 4.
C CONNECTIVITIES BETWEEN NEW REGIONS I, J IN THE LAST RECORD ARE FOUND
C BY TESTING WHETHER IS(I).EQ.IS(J)
C
CALL SVSCL1 (IDENT, MR*MR, .FALSE.)
DO 20 I=1, NR
IDENT(I, I)=.TRUE.
IF (I.EQ.NR) GO TO 20
I1=I+1
DO 30 J=I1, NR
30 IDENT(I, J)=IS(I).EQ.IS(J)
20 CONTINUE
RETURN
100 FORMAT (/// ' THE FOLLOWING IS A PRELIMINARY TABLE FOR MODIFYING SEG
. MENT NUMBER 'I3)
200 FORMAT (' ) THE FOLLOWING IS AN UPDATED TABLE FOR MODIFYING SEGMENT
. UMBER 'I3)
300 FORMAT (' THE DISTINCT REGION NUMBERS PRESENT IN THE LAST RECORD OF
. THE CURRENT SEGMENT (TO BE REASSIGNED NOS. 1 THROUGH 'I4/' IN THE
. EXT SEGMENT): ')
400 FORMAT (' ASSIGNMENTS FOR THE ABOVE REGIONS. FROM THE PRELIMINARY L
. BK-UP TABLE: ')
END

```

```

SUBROUTINE RIDERS5(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 TERE EXISTS A SEQUENCE (K(ID),ID=1,...,L) WITH K(1)=I,
C K(L)=J AND IDENT(K(ID),K(ID+1))=.TRUE.
      INTEGER*2 IT(N),M(1)
      LOGICAL*1 IDENT(MD,N)
      DO 100 I=1,N
100   IT(I)=I
      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
      C
      C
      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

```


~~SLBRoutine RIDER6(IX,M,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(M),IS(1)
N=1
IS(1)=1
DO 10 I=1,M
IF(IX(I).LE.1)GO TO 10
IF(N.EQ.0)GO TO 20
DO 30 J=2,N
30 IF((IS(J).EQ.IX(I))GO TO 40
20 N=N+1
IS(N)=IX(I)
IX(I)=N
GO TO 10
40 IX(I)=J
10 CONTINUE
RETURN
END
```

COMPUTER SCIENCES CORPORATION, MAR. 12, 1976.

SUBROUTINE RIDER7(IX,IS,N,IY,M)
INTEGER*2 IX(N),IS(N),IY(M)

```
C  
C   MODIFY RELEVANT ENTRIES IN IY ACCORDING TO CONNECTIVITIES FOUND I  
C   IN IS.  
   IF(N.EQ.1)RETURN  
   MAX=IS(2)  
   MIN=IS(2)  
   CALL VMAXI2(IS(2),N-1,MAX)  
   CALL VMINI2(IS(2),N-1,MIN)  
   DO 10 J=1,M  
   IF(IY(J).LT.MIN.OR.IY(J).GT.MAX)GO TO 10  
   DO 20 I=2,N  
   IF(IY(J).NE.IS(I))GO TO 20  
   IY(J)=IX(I)  
   GO TO 10  
20  CONTINUE  
10  CONTINUE  
   RETURN  
   END
```

```
SUBROUTINE SEQSL(X,ISEQ,N,ND)
DIMENSION X(ND,2),ISEQ(ND),T(2),TT(2)
```

```
C
C MUST EQUIVALENCE (X(1,2),ISEQ(1))
C
```

```
IFLAG=0
GO TO 10
ENTRY SEQLS(X,ISEQ,N,ND)
IFLAG=1
10 DO 20 I=1,N
20 ISEQ(I)=1
CALL SORT(X,1,N,ND,2,T,TT)
IF (IFLAG.EQ.0) RETURN
CALL FLIPV(X,N,X)
CALL FLIPV(ISEQ,N,ISEQ)
RETURN
END
```

COMPILER OPTIONS NAME MAIN,OPT=02,LINECNT=56,SIZE=0000K,
SOURCE,BCD,NOLIST,NODECK,LOAD,MAP,NOEDIT,LD,NOXREF

```

151
ISN 0002 SUBROUTINE SORT(A,II,JJ,MM,NN,T,TT)
ISN 0003 DIMENSION A(MM,NN),T(NN),TT(NN),IU(16),IL(16)
ISN 0004 INTEGER A,T,TT
ISN 0005 M=1
ISN 0006 I=II
ISN 0007 J=JJ
ISN 0008 -5 IF(I.GE.J)GO TO 70
ISN 0010 10 K=I
ISN 0011 IJ=(I+J)/2
ISN 0012 CALL MVHRMR(A,MM,NN,T,I,IJ,I)
ISN 0013 IF(A(I,I).LE.T(I))GO TO 20
ISN 0015 CALL MVHRMR(A,MM,NN,A,MM,I,IJ)
ISN 0016 CALL MVHRMR(T,I,NN,A,MM,I,I)
ISN 0017 CALL MVHRMR(A,MM,NN,T,I,IJ,I)
ISN 0018 -20 L=J
ISN 0019 IF(A(J,I).GE.T(I))GO TO 40
ISN 0021 CALL MVHRMR(A,MM,NN,A,MM,J,IJ)
ISN 0022 CALL MVHRMR(T,I,NN,A,MM,I,J)
ISN 0023 CALL MVHRMR(A,MM,NN,T,I,IJ,I)
ISN 0024 IF(A(I,I).LE.T(I))GO TO 40
ISN 0026 CALL MVHRMR(A,MM,NN,A,MM,I,IJ)
ISN 0027 CALL MVHRMR(T,I,NN,A,MM,I,I)
ISN 0028 CALL MVHRMR(A,MM,NN,T,I,IJ,I)
ISN 0029 GO TO 40
ISN 0030 -30 CALL MVHRMR(A,MM,NN,A,MM,K,L)
ISN 0031 CALL MVHRMR(TT,I,NN,A,MM,I,K)
ISN 0032 -40 L=L-1
ISN 0033 IF(A(L,I).GT.T(I))GO TO 40
ISN 0035 CALL MVHRMR(A,MM,NN,TT,I,L,I)
ISN 0036 50 K=K+1
ISN 0037 IF(A(K,I).LT.T(I))GO TO 50
ISN 0039 IF(K.LE.L)GO TO 30
ISN 0041 IF(L-1.LE.J-K)GO TO 60
ISN 0043 IL(M)=I
ISN 0044 IU(M)=L
ISN 0045 I=K
ISN 0046 M=M+1
ISN 0047 GO TO 80
ISN 0048 -60 IL(M)=K
ISN 0049 IU(M)=J
ISN 0050 J=L
ISN 0051 M=M+1
ISN 0052 GO TO 80
ISN 0053 70 M=M-1
ISN 0054 IF(M.EQ.0)RETURN
ISN 0056 I=IL(M)
ISN 0057 J=IU(M)
ISN 0058 80 IF(J-1.GE.I)GO TO 10
ISN 0060 IF(I.EQ.II)GO TO 5
ISN 0062 I=I-1
ISN 0063 -90 I=I+1
ISN 0064 IF(I.EQ.J)GO TO 70

```

```
ISN-0066 CALL MVHRMR(A,MM,NN,T,1,I+1,1)
ISN 0067 IF(A(I,1).LE.T(1))GO TO 90
ISN 0069 K=I
ISN 0070 100 CALL MVHRMR(A,MM,NN,A,MM,K,K+1)
ISN 0071 K=K-1
ISN 0072 IF(T(1).LT.A(K,1))GO TO 100
ISN 0074 CALL MVHRMR(T,1,NN,A,MM,1,K+1)
ISN 0075 GO TO 90
ISN 0076 END
```

152

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

COMPILER-OPTIONS NAME=MAIN,OPT=02,LINECNT=56,SIZE=0000K,
SOURCE,BCD,NOLIST,NODECK,LCAD,MAP,NOEDIT,LD,NOXREF

```

SN 0002 SUBROUTINE SVSCI(IX,N,IS)
SN 0003 DIMENSION IX(N)
SN 0004 DO 10 I=1,N
SN 0005 10 IX(I)=IS
SN 0006 RETURN
SN 0007 END

```

158

COMPILER OPTIONS - NAME= MAIN,OPT=C2,LINECNT=56,SIZE=CHCK,
SOURCE,BCD,NOLIST,NODECK,LLAU,MAP,NLEDT,LD,NJXREF

```

15N 0002  SUBROUTINE SVSCI2(IX,N,IS)
15N 0003  INTEGER*2 IX(N)
15N 0004  DO 10 I=1,N
15N 0005  IX(I)=IS
15N 0006  RETURN
15N 0007  END

```


COMPILER OPTIONS NAME MAIN,OPT=02,LINECNT=56,SIZE=0000K,
SOURCE,BCD,NOLIST,NODECK,LOAD,MAP,NOEDIT,IO,NOXREF

ISN 0002 SUBROUTINE SVSGL1(IX,N,L)
ISN 0003 LOGICAL * 1 IX(N),L
ISN 0004 DO 10 I=1,N
ISN 0005 10 IX(I)=L
ISN 0006 RETURN
ISN 0007 END

COMPILER OPTIONS - NAME=MAIN,OPT=C2,LINECNT=56,SIZE=000K,
SOURCE,DCU,NOLIST,NOCHECK,LLAD,MAP,WLDIT,LD,NOXREF

```

15N 0002  SUBROUTINE VMAX12(IX12,MYMAX12)
15N 0003  INTEGER*2 IX12(N)
15N 0004  DO 10 I=1,N
15N 0005  10  IF(IX12(I).GT.MYMAX12)MYMAX12=IX12(I)
15N 0007  RETURN
15N 0008  ENTRY VMIN12(IX12,M,MIN12)
15N 0009  DO 20 I=1,N
15N 0010  20  IF(IX12(I).LT.MIN12)MIN12=IX12(I)
15N 0012  RETURN
15N 0013  END

```

156

COMPILER OPTIONS = NAME= MAIN,OPT=02,LINECNT=56,SIZE=C000K,
SOURCE,DCD,NOLIST,NODECK,LOAD,MAP,NODEIT,LD,NUXREF

ISN 0002 --- SUBROUTINE VMAX14(IX14,N,MAX14)

ISN 0003 --- DIMENSION IX14(N)

ISN 0004 --- DO 10 I=1,N

ISN 0005 10 MAX14=MAX0(IX14(I),MAX14)

ISN 0006 --- RETURN

C

ISN 0007 --- ENTRY VMIN14(IX14,N,MIN14)

ISN 0008 --- DO 20 I=1,N

ISN 0009 20 MIN14=MIN0(IX14(I),MIN14)

ISN 0010 --- RETURN

ISN 0011 --- END

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

157

COMPILER OPTIONS = NAME= MAIN,OPT=027,LINECNT=56,SIZE=001,K,
SOURCE,BLO,NOLIST,NODECK,LEAD,MAP,HELD11,IO,NOXREF

```

15N 0002  SUBROUTINE VMDV2(IX,N,IY)
15N 0003  I=INTEGER*2 IX(N),IY(N)
15N 0004  DO 10 I=1,N
15N 0005  10 IY(I)=IX(I)
15N 0006  RETURN
15N 0007  END

```

158

7. DELETION OF BOUNDARY POINTS

7.1 NAME

DBOUND

7.2 PURPOSE

To modify each of the "0" pixels in an image to the most frequently occurring number in its 3 by 3 neighborhood. (This is useful, for example, in generating a level I GTM from a level II map and/or suppressing all the boundary points in a GTM and replacing them with reasonable class labels).

7.3 CALLING SEQUENCE

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

where

NREC = Number of records in the input image;

NEL = 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 as indicated in the listings.

All the calling arguments except IX and IY are inputs.

7.4 INPUT-OUTPUT

Both the input and output sequential data sets have the same format. The number of records is NREC. The number of pixels per record is NEL and the number of bytes per pixel is 4. The records are in unformatted FORTRAN.

7.5 EXITS

No nonstandard exits

7.6 USAGE

The program is in FORTRAN IV and implemented on the IBM 360 using the H compiler. The program is in the users' library as a load module.

7.7 EXTERNAL INTERFACES

The subprograms required by this routine are:

SARN, a sequential access array read routine;

VMOV, a routine to move a vector in core;

MAJOR, a function giving the most frequently occurring number in a 3 by 3 neighborhood.

7.8 PERFORMANCE SPECIFICATIONS

7.8.1 Storage

This subroutine is 1036 bytes long. With the main program needed to call it for an image with NEL=866, the external references and buffers, the storage required is 40K bytes.

7.8.2 Execution Time

Depends on image size. For a test case of 1624 by 866 pixels it took approximately 100 seconds.

7.8.3 I/O Load

None

7.8.4 Restrictions

None

7.9 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, I3 are set at 1, 2 and 3 respectively. After each record is processed, the pointers I1, I2, 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 most frequent number occurring in the set of eight (If such a number is not unique, the first encountered number is taken).

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.

7.10 COMMENTS

None

7.11 LISTINGS

The listings of DBOUND and MAJOR are attached at the end of this section.

7.12 TESTS

This program was used in removing the extraneous boundary points after conversion of the level II GTM of the Mobile Bay region to a level I map. Line-printer plots of the maps before and after the application of DBOUND indicate satisfactory operation of this program.

COMPILER OPTIONS - NAME= MAIN,OPT=02,LINESCT=56,SIZE=0002K,

SOURCE,FCNTIC,NJLIST,NODECK,LOAD,MAP,NODEIT,IO,NQXOFF

```

ISN 0002 SUBROUTINE DBOUND(NREC,NEL,NFL2,NTAPI,NTAPD,IX,IY)
ISN 0003 DIMENSION IX(NFL2,3),IY(NEL)
ISN 0004 NFL4=NEL*4
      C INITIALIZE ARRAY IX.
      C THE PURPOSE OF THIS PROGRAM IS TO MODIFY POINTS IN THE IMAGE ON NTAPI
      C WITH NUMBER 0 TO THE NUMBER OCCURING MOST FREQUENTLY IN A 3 BY 3
      C NEIGHBORHOOD.
ISN 0005 I1=1
ISN 0006 I2=2
ISN 0007 I3=3
ISN 0008 CALL SARV(NTAPI,IX(2,I1),NFL4)
ISN 0009 IX(1,I1)=IX(2,I1)
ISN 0010 IX(NEL2,I1)=IX(NEL+1,I1)
ISN 0011 CALL VMOV(IX(1,I1),NFL2,IX(2,I2))
ISN 0012 DO 10 I=1,NREC
      C
      C IF(I.LT.NREC) READ (I+1)ST RECORD INTO IX(*,I3).
ISN 0013 IF(I.LT.NREC)CALL SARV(NTAPI,IX(2,I3),NFL4)
ISN 0015 IF(I.EQ.NREC)CALL VMOV(IX(2,I2),NFL,IX(2,I3))
ISN 0017 IX(1,I3)=IX(2,I3)
ISN 0018 IX(NEL2,I3)=IX(NEL+1,I3)
      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.
ISN 0019 DO 20 J=1,NFL
ISN 0020 IY(J)=IX(J+1,I2)
ISN 0021 20 IF(IY(J).EQ.0)IY(J)=MAJOR(IX,NEL2,I1,I2,I3,J+1)
ISN 0023 WRITE(NTAPD,IY)
      C
      C MODIFY I1,I2,I3 IN PREPARATION FOR THE NEXT RECORD.
ISN 0024 I1=I1
ISN 0025 I1=I2
ISN 0026 I2=I3
ISN 0027 I3=I1
ISN 0028 10 CONTINUE
ISN 0029 RETURN
ISN 0030 END

```


COMPILER OPTIONS - NAME= MAIN,OPT=02,LINFCNT=56,STZF=0000K,
SOURCE,FBC,IC,NOLIST,NOFCK,LPAD,MAP,NOEDIT, ID,NOXREF

```

168
ISN 0002      FUNCTION MAJOR(IX,NEL,I1,I2,I3,J)
ISN 0003      DIMENSION IX(NEL,3),LABEL(9),NUMBER(9)
              C
              C  FIND THE MOST FREQUENTLY OCCURRING NUMBER AMONG THE EIGHT NEIGHBORS
              C  OF IX(J,I2).  NOTE THAT 1.LT.J.LT.NEL.
ISN 0004      LABEL(1)=IX(J-1,I1)
ISN 0005      NUMBER(1)=1
ISN 0006      N=1
ISN 0007      J2=J-2
ISN 0008      DO 30 I=1,3
ISN 0009      IF(I.EQ.1)II=I1
ISN 0011      IF(I.EQ.2)II=I2
ISN 0013      IF(I.EQ.3)II=I3
ISN 0015      KM=1
ISN 0016      IF(I.EQ.1)KM=2
ISN 0018      INC=1
ISN 0019      IF(I.EQ.2)INC=2
ISN 0021      DO 10 K=KM,3,INC
ISN 0022      DO 20 L=1,N
ISN 0023      20  IF(IX(J2+K,II).EQ.LABEL(L))GO TO 40
ISN 0025      N=N+1
ISN 0026      LABEL(N)=IX(J2+K,II)
ISN 0027      NUMBER(N)=1
ISN 0028      GO TO 10
ISN 0029      40  NUMBER(L)=NUMBER(L)+1
ISN 0030      10  CONTINUE
ISN 0031      30  CONTINUE
ISN 0032      MAX=0
ISN 0033      DO 50 I=1,N
ISN 0034      IF(NUMBER(I).LE.MAX)GO TO 50
ISN 0036      MAJOR=LABEL(I)
ISN 0037      MAX=NUMBER(I)
ISN 0038      50  CONTINUE
ISN 0039      RETURN
ISN 0040      END

```

8. COMPUTATION OF CONTINGENCY MATRICES

8.1 NAME

CONTMATS

8.2 PURPOSE

To obtain and print "contingency matrices"[6], 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).

8.3 CALLING SEQUENCE

This is a main program which takes card inputs as indicated below:

```
          READ 100, TITLE
          READ 200, NREC, NEL, M, N
100      FORMAT (80A1)
200      FORMAT (4I6)
```

where

TITLE is a title of up to 80 characters to be printed on the top of each page of output.

NREC, NEL are number of records and number of pixels per record, respectively.

M, N are the numbers of classes in the two maps.

8.4 INPUT-OUTPUT

8.4.1 Input

The input maps should be sequential data sets on units 8 and 9 with NREC records and NEL pixels per record on each of them. The number of bytes per pixel should be 4 and the records should be unformatted and FORTRAN readable.

8.4.3 File Storage

None

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

8.5 EXITS

Not Applicable

8.6 USAGE

The program is in FORTRAN IV and implemented on the IBM 360 using the H compiler. The program and the external references needed are in the users' library as load modules.

8.7 EXTERNAL INTERFACES

The subroutine linkage is indicated in the following table:

Calling Program	Programs Called
CONTMATS	CONMAT CONMXP
CONMAT	SARN VMOV SVSCI INTLOG
CONMXP	SVSCI

8.8 PERFORMANCE SPECIFICATIONS

8.8.1 Storage

The present version of the program is designed for $NEL \leq 2000$, $M \leq 15$, $N \leq 15$. The main program size is 62340 bytes. Including the buffers and the external references, the program requires 110K bytes.

8.8.2 Execution Time

Depends largely on image size and number of classes. For the case $NREC=1624$, $NEL=866$, $M=N=6$, it took approximately 10 minutes.

8.8.3 I/O Load

None

8.8.4 Restrictions

None

8.9 METHOD

The definitions of contingency matrices used here have been discussed in detail in [6] 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

- (i) M*N matrices (size 4 by 8) showing counts of agreements and disagreements for each type of transition for each pair of classes;
- (ii) 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 (i) and dividing by 3;
- (iii) A 4 by 4 matrix showing totals of each type of transition obtained by adding all the matrices in (ii);
- (iv) 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 4x4 matrix corresponding to classes (I, J) defined in (ii);
- (v) 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.

8.10 COMMENTS

None

8.11 LISTINGS

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

8.12 TESTS

Portions of the printout from a test run on the Mobile Bay GTM v/s LCM (December 5, 1973) are shown at the end of this section.

COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=0000K,
SOURCE,ERLOTIC,NULIST,NODECK,LOAD,MAP,NCE01T,TD,NOXREF

```

C
C      D*N IX(NEL*2),IY(NEL*2),IH(4,8,M,N),IH1(4,4),IH2(M,N)
C      DIMENSION INVI(M),INV2(N)
ISN 0002      DIMENSION IX(4000),IY(4000),IH(32,225),IH2(225),INVI(15),INV2(15)
ISN 0003      LOGICAL*1 TITLE(80)
C
C      MAIN PROGRAM TO FIND AND PRINT CONTIGENCY MATRICES AND RELATED
C      RESULTS.
C
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 THE NUMBER OF
C              RECORDS, NUMBER OF PIXELS PER RECORD, NUMBER OF CLASSES IN
C              MAP 1 AND NUMBER OF CLASSES IN MAP 2, RESPECTIVELY.
C      THE INPUT MAPS SHOULD BE IN UNFORMATTED FORTRAN READABLE RECORDS
C      ON UNITS 8 AND 9, ONE PIXEL/WORD.
ISN 0004      READ(5,100)TITLE
ISN 0005      READ(5,200)NREC,NEL,M,N
C
C      FIND AND PRINT MATRICES SHOWING NUMBERS OF AGREEMENTS AND DIS-
C      AGREEMENTS OF TRANSITIONS FOR EACH PAIR OF CLASSES.
ISN 0006      CALL CONMAT(8,9,NREC,NEL,M,N,IX,IY,IH)
ISN 0007      CALL CONHXP(TITLE,IH,IH2,INVI,INV2,NREC,NEL,M,N)
ISN 0008      STOP
ISN 0009      100  FORMAT(80A1)
ISN 0010      200  FORMAT(416)
ISN 0011      END

```

167

COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=000CK,

ISN 0002 SOURCE,EOC0IC,NOEDIT,NOBICK,LOAD,MAP,NOEDIT,IO,NOXREF
 ISN 0003 SUBROUTINE,CCOMAT(NTAP1,NTAP2,NREC,NEL,M,N,IX,IY,IH)
 DIMENSION IX(NEL,2),IY(NEL,2),IH(4,8,M,N)

C

C THIS PROGRAM FINDS CONTINGENCY PATRICS INDICATING AGREEMENTS
 C BETWEEN TWO MAPS IN TERMS OF CLASS LABELS AND THE BOUNDARY TYPES.
 C IH(*,*,I,J) REFERS TO LOCATIONS WITH CLASS I IN MAP 1 AND CLASS J
 C IN MAP 2. LEFT HALF OF IH GIVES A COUNT OF AGREEMENTS AND THE
 C RIGHT HALF, DISAGREEMENTS.
 C ROWS OF IH(*,*,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

ISN 0004 I1=1
 ISN 0005 I2=2
 ISN 0006 NEL4=NEL*4

C

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

C INITIALIZE THE ARRAYS IX AND IY. THE "PREVIOUS" ROW TO ROW 1 IS
 C CONSIDERED IDENTICAL TO ROW 1.

C

168 ISN 0007 CALL SARN(NTAP1,IX,NEL4)
 ISN 0008 CALL SARN(NTAP2,IY,NEL4)
 ISN 0009 CALL VMOV(IX,NEL,IX(1,2))
 ISN 0010 CALL VMOV(IY,NEL,IY(1,2))
 ISN 0011 CALL SVSCI(IH,32*M*N,C)

C

ISN 0012 LOOP ON RECORDS.
 DO 10 I=1,NREC

C

ISN 0013 LOOP ON PIXELS.
 ISN 0014 DO 20 J=1,NEL
 JP=MAX(1,J-1)

C

ISN 0015 NONPOSITIVE VALUES OF MAP LABELS ARE NOT OF INTEREST.
 IF(IX(J,I2).LE.0.OR.IY(J,I2).LE.0)GO TO 20

C

C FIND ROW AND COLUMN NUMBERS IN IH TO BE INCREMENTED.

C

C CHECK THE NATURE OF THE BOUNDARIES IN BOTH THE MAPS.

ISN 0017 K=IX(J,I2)
 ISN 0018 L=IY(J,I2)
 ISN 0019 II=1+INTLOG(IX(J,I2).NE.IX(J,I1))+2*INTLOG(IX(J,I2).NE.IX(JP,I2))
 ISN 0020 JJ=1+INTLOG(IY(J,I2).NE.IY(J,I1))+2*INTLOG(IY(J,I2).NE.IY(JP,I2))

C

C FIND THE INCREMENTS.

ISN 0021 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))

REPRODUCIBILITY OF THE
 ORIGINAL PAGE IS POOR

```
ISN 0022      INC3=3-INC
ISN 0023      IH(I1,JJ,K,L)=IH(I1,JJ,K,L) + INC
ISN 0024      IH(I1,JJ+4,K,L)=IH(I1,JJ+4,K,L)+INC3
ISN 0025      20  CONTINUE
              C
              C  EXCHANGE I1 AND I2
ISN 0026      IH=I1
ISN 0027      I1=I2
ISN 0028      I2=I1
              C
              C  READ NEXT RECORDS INTO IX(*,I2),IY(*,I2)
ISN 0029      IF(I.EQ.NREC)GO TO 10
ISN 0030      CALL SARN(NTAP1,IX(1,I2),NEL4)
ISN 0031      CALL SARN(NTAP2,IY(1,I2),NEL4)
ISN 0032      10  CCNTINUE
ISN 0033      RETURN
ISN 0034      END
ISN 0035
```

COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=C000K,

SOURCEF, EPCDIC, NCLIST, NLOECK, LOAD, MAP, NQEDIT, ID, NQXREF

```

ISN 0002 SUBROUTINE COMXP(TITLE, IH, IH2, INV1, INV2, NREC, NEL, M, N)
ISN 0003 DIMENSION IP(4,3,P,N), IH1(4,4), IH2(M,N), INV1(M), INV2(N)
ISN 0004 LOGICAL*1 TITLE(80)

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 THE NUMBER OF
C RECORDS, NUMBER OF PIXELS PER RECCRD, NUMBER OF CLASSES IN
C MAP 1 AND NUMBER OF CLASSES IN MAP 2, RESPECTIVELY.

ISN 0005 L=0
ISN 0006 DO 10 I=1,M
ISN 0007 DO 10 J=1,N
ISN 0008 IF(MOD(L,4).NE.C)GO TO 15
ISN 0009 WRITE(6,110)TITLE
ISN 0010 WRITE(6,400)NREC,NEL
ISN 0011 CONTINUE
ISN 0012 15 L=L+1
ISN 0013 WRITE(6,500)I,J
ISN 0014 DO 20 K=1,4
ISN 0015 WRITE(6,300)((IH(K,KK,I,J),KK=1,8)
ISN 0016 20 CONTINUE
ISN 0017 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 OCCURENCES
C OF CLASSES (I,J) IN MAPS 1,2, THE NUMBERS OF JOINT OCCURENCES
C OF EACH TYPE OF TRANSITION IN THE TWO MAPS.

170 ISN 0018 DO 30 I=1,M
ISN 0019 DO 30 J=1,N
ISN 0020 DO 30 K=1,4
ISN 0021 DO 30 L=1,4
ISN 0022 30 IF(K,L,I,J)=(IH(K,L,I,J)+ IH(K,L+4,I,J))/2
ISN 0023 L=L+1
ISN 0024 DO 40 I=1,M
ISN 0025 DO 40 J=1,N
ISN 0026 IF(MOD(L,4).NE.C)GO TO 45
ISN 0027 WRITE(6,110)TITLE
ISN 0028 WRITE(6,410)NREC,NEL
ISN 0029 CONTINUE
ISN 0030 45 L=L+1
ISN 0031 WRITE(6,500)I,J
ISN 0032 DO 50 K=1,4
ISN 0033 WRITE(6,310)((IH(K,KK,I,J),KK=1,4)
ISN 0034 50 CONTINUE
ISN 0035 40 CONTINUE

C
C FIND IH1, THE MATRIX OF COUNTS OF TRANSITION TYPES(WITHOUT REGARD
C TO CLASS LABELS) AND IH2, THE MATRIX OF JOINT OCCURENCES OF CLASS
C LABELS(WITH NO REGARD TO TRANSITION TYPES).
ISN 0036 CALL SVSCI(IH1,16,C)
ISN 0037 CALL SVSCI(IH2,M*N,C)
ISN 0038 DO 60 I=1,M
ISN 0039 DO 60 J=1,N

```



```

ISA 0140      DO 60 K=1,4
ISA 0141      DO 60 L=1,4
ISA 0142      IH1(K,L)=IH1(K,L)+IH(K,L,I,J)
ISA 0143      60  IH2(I,J)=IH2(I,J)+IH(K,L,I,J)
              C
              C PRINT IH1 AND THE CORRESPONDING SIMILARITY MEASURE.
ISA 0144      WRITE(6,110)TITLE
ISA 0145      WRITE(6,420) NREC,NEL
ISA 0146      DO 70 K=1,4
ISA 0147      70  WRITE(6,310)(IH1(K,L),L=1,4)
ISA 0148      ITR=C
ISA 0149      ISUM=0
ISA 0150      DO 90 I=1,4
ISA 0151      ITR=ITR+IH1(I,I)
ISA 0152      DO 90 J=1,4
ISA 0153      90  ISUM=ISUM+IH1(I,J)
ISA 0154      PCT=ITR*100./ISUM
ISA 0155      WRITE(6,600)ITR,ISUM,PCT
ISA 0156      WRITE(6,430)
              C
              C PRINT IH2 AND THE CORRESPONDING SIMILARITY MEASURE.
ISA 0157      DO 80 I=1,M
ISA 0158      80  WRITE(6,310)(IH2(I,J),J=1,N)
ISA 0159      ITR=0
ISA 0160      ISUM=0
ISA 0161      DO 95 I=1,M
ISA 0162      IF(I.LE.N) ITR=ITR+IH2(I,I)
ISA 0163      DO 95 J=1,N
ISA 0164      95  ISUM=IH2(I,J)+ISUM
ISA 0165      PCT=ITR*100./ISUM
ISA 0166      WRITE(6,700)ITR,ISUM,PCT
              C
ISA 0168      CALL SVSCI(INV1,M,C)
ISA 0169      CALL SVSCI(INV2,N,C)
ISA 0170      DO 85 I=1,M
ISA 0171      DO 85 J=1,N
ISA 0172      85  INV1(I)=INV1(I)+IH2(I,J)
ISA 0173      INV2(J)=INV2(J)+IH2(I,J)
ISA 0174      WRITE(6,800)(INV1(I),I=1,M)
ISA 0175      WRITE(6,810)(INV2(J),J=1,N)
ISA 0176      RETURN
ISA 0177      110 FORMAT('1'20X80A1)
ISA 0178      300 FORMAT('0'4I8,' | '4I8)
ISA 0179      310 FORMAT('0'15I8)
ISA 0180      400 FORMAT('/' MATRICES SHOWING COUNTS OF AGREEMENTS AND DISAGREEMENTS
              .FOR EACH TYPE OF TRANSITION/' MAP SIZE='15,' BY'15,'.)
ISA 0181      410 FORMAT('/' MATRICES SHOWING COUNTS OF EACH TYPE OF TRANSITION/'
              ./' MAP SIZE='15,' BY'15,'.)
ISA 0182      420 FORMAT('/' MATRIX SHOWING TOTALS OF EACH TYPE OF TRANSITION/'
              ./' MAP SIZE='15,' BY'15,'.)
ISA 0183      430 FORMAT('/' CONTINGENCY TABLE')
ISA 0184      500 FORMAT('/' CLASS NUMBER IN MAP 1='12,' CLASS NUMBER IN MAP 2='12)
ISA 0185      600 FORMAT(' TRANSITION SIMILARITIES='17,'; TOTAL='17,'; PERCENTAGE='

```

F7.2)

ISN 0180 703 FORMAT(' NUMBER OF POINT BY POINT SIMILARITIES='17,' TOTAL='17,
' PERCENTAGE='F7.2)
ISN 0187 800 FORMAT(/' INVENTORY OF MAP1:/'(1X15I8))
ISN 0188 810 FORMAT(/' INVENTORY OF MAP2:/'(1X15I8))
ISN 0189 END

172

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

CGMPILER OPTIONS - NAME= MAIN,OPT=Q2,LINECNT=56,SIZE=0000K,

SOURCE,EBCCDIC,NCLTST,NODECK,LOAD,MAP,NOEDIT,IO,NOXREF

```
ISN 0002 FUNCTION INTLOG(L)
ISN 0003 LOGICAL L
ISN 0004 INTLOG=0
ISN 0005 IF(L)INTLOG=1
ISN 0007 RETURN
ISN 0008 END
```

GTM V/S LCM(120573 MOBILE BAY DATA)

MATRICES SHOWING COUNTS OF AGREEMENTS AND DISAGREEMENTS FOR EACH TYPE OF TRANSITION

MAP SIZE= 1624 BY 866.

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

71601	12518	13958	4568	1	0	6259	6979	9136
792	479	196	282	1	396	97	392	306
798	182	526	267	1	399	364	164	309
146	77	92	140	1	292	100	121	103

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

0	3051	3144	4819	1	39858	12915	13902	8567
148	96	117	155	1	1415	612	393	394
132	150	126	173	1	1716	471	786	433
84	34	41	55	1	435	161	181	164

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

0	2159	2327	3147	1	51039	12571	12433	6324
627	38	275	73	1	1953	598	445	347
775	316	31	58	1	2273	530	692	353
527	54	55	9	1	502	153	167	156

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

0	41	156	173	1	5040	694	1113	676
121	0	54	4	1	263	21	78	14
46	23	0	1	1	113	52	15	26
72	0	7	0	1	36	3	20	9

MATRICES SHOWING COUNTS OF EACH TYPE OF TRANSITION

MAP SIZE= 1624 BY 866.

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

23867	6259	6979	4568
396	192	196	196
399	182	230	192
146	59	71	81

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

13286	5322	5682	4462
521	236	170	183
616	207	304	202
173	65	74	73

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

17013	4910	4920	3157
360	212	240	140
1116	282	241	137
343	69	74	55

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

1687	245	423	283
128	7	44	6
53	25	5	9
36	1	9	3

GTM V/S LCM(120572 MOBILE BAY DATA)

MATRIX SHOWING TOTALS OF EACH TYPE OF TRANSITION

MAP SIZE= 1524 BY 866.

971501 158283 112816 77171

15932 5276 4714 3316

16934 4847 6475 3907

5814 1529 1743 1480

TRANSITION SIMILARITIES= 984662; TOTAL=1341588; PERCENTAGE= 73.40

CONTINGENCY TABLE

44013 31576 33669 2957 6750 2070

45943 101374 31773 36 4576 86

36532 61249 391853 637 31889 168

1433 257 917 397292 5518 1685

938J 4320 36795 3900 41592 287

304J 2476 2508 520 750 1215

NUMBER OF PLANT BY POINT SIMILARITIES= 977339 TOTAL=1341588 PERCENTAGE= 72.85

INVENTORY OF MAP1:

121035 183790 522378 407102 96774 10509

INVENTORY OF MAP2:

140391 201752 497515 405342 91077 5511

9. COMPARISON OF SUPERVISED CLASSIFICATION MAPS

9.1 NAME

COMPSUMP

9.2 PURPOSE

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.

9.3 CALLING SEQUENCE

This is a main program with card inputs as follows:

```
          READ 100, TITLE
          READ 300, NREC, NEL, M, N
100      FORMAT (72A1)
300      FORMAT (4I6)
```

where

TITLE is a title of up to 72 characters (to be printed);
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.

9.4 INPUT-OUTPUT

9.4.1 Input

The input maps 1 and 2 to this program should be on units 8 and 10 respectively. They should have NREC records, NEL pixels per record and 4 bytes per pixel in unformatted FORTRAN readable form.

9.4.2 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.

9.4.3 File Storage

None

9.5 EXITS

Not applicable

9.6 USAGE

This program is in FORTRAN IV and implemented on the IBM 360 with the H compiler. The program and the associated subroutines are available as load modules on the users' library.

9.7 EXTERNAL INTERFACES

The subroutine linkage is indicated below:

Calling Program	Programs Called
COMPSUMP	SVSCL1 DCLARE
DCLARE	CLAM PRTMXP SVSCI SARN VMOV
CLAM	SVSCI
PRTMXP	MXINX

9.8 PERFORMANCE SPECIFICATIONS

9.8.1 Storage

This main program, presently limited to $NEL \leq 2000$ and $M*N \leq 200$, is 34932 bytes long, and requires 72K bytes of storage with the external references and buffers.

9.8.2 Execution Time

The execution time is approximately proportional to the number of pixels in the input maps. With $NREC = NEL = 2000$ the program will take approximately 7 minutes of CPU time on IBM 360/65.

9.8.3 Restrictions

$NEL \leq 2000$, $M*N \leq 200$

9.9 METHOD

This program first sets an M by N matrix LTABL by

```
LTABL(I, J) = 'H' for I ≠ J
LTABL(I, J) = blank for I = J.
```

Then it calls the routine DCLARE. This subroutine finds (call to CLAM) and prints (call to PRTMXP) the joint histogram between the two maps.

The next part of DCLARE is used to separate the types of differences between the two maps and indicate them by different symbols. The EBCDIC characters blank, '.', '+', and 'H' are used to indicate no difference between the maps, exterior points, 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 less than or 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 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.

9.10 COMMENTS

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

9.11 LISTINGS

The listings of the program and the important subroutines called by it are attached at the end of this section.

9.12 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.

COMPILER OPTIONS - NAME= MAIN,OPT=02,LINECNT=56,SIZE=0000K,

SOURCE, EBCDIC, NOLIST, NODECK, LOAD, MAP, NOEDIT, ID, NOXREF

```

ISN 0002      DIMENSION IX(6000),IY(2000)
ISN 0003      LOGICAL*1 LY(2000),LTABL(200)
ISN 0004      LOGICAL*1 BLANK/' ',ETCH/'H'/'...
ISN 0005      LOGICAL*1 TITLE(72)
ISN 0006      READ 100,TITLE
ISN 0007      PRINT 200,TITLE
ISN 0008      READ 300,NREC,NEL,M,N
ISN 0009      PRINT 400,NREC,NEL,M,N
ISN 0010      CALL SVSCL1(LTABL,M*N,ETCH)
ISN 0011      MN=MING(M,N)
ISN 0012      DO 10 I=1,MN
ISN 0013      10  LTABL((I-1)*M+1)=BLANK
ISN 0014      CALL DCLARE(NREC,NEL,M,N,LTABL,IX,IY,LY)
ISN 0015      STOP
ISN 0016      100  FORMAT(72A1)
ISN 0017      200  FORMAT('I'25(//)30X72A1)
ISN 0018      300  FORMAT(4I6)
ISN 0019      400  FORMAT(25X' IMAGE SIZE='15,' BY'15,'. NUMBERS OF CLASSES IN MAPS 1
                   AND 2 ARE'13,' AND'13,'.')
ISN 0020      END

```

081

COMPILER OPTIONS - NAME= MAIN,OPT=C2,LINECNT=56,SIZE=0000K,

SOURCE, EBCDIC, NOLIST, NOCHECK, LOAD, MAP, NOEDIT, ID, NOXREF

ISN 0002 SUBROUTINE DCLARE(NREC,NEL,M,N,LTABL,IX,IY,IY)

C

THE PURPOSE OF THIS ROUTINE IS TO FIND AND PRINT THE JOINT HISTOGRAM

BETWEEN TWO MAPS, EACH WITH NREC LINES AND NEL PIXELS PER LINE.

THE FIRST MAP SHOULD HAVE M CLASSES OR LESS, AND THE SECOND, N OR

LESS. THE ARRAY LTABL IS USED TO SPECIFY ERRONEOUS COMBINATIONS

OF CLASS NUMBERS. LTABL IS AN INPUT ARRAY TREATED AS AN M BY N MATRIX

WITH BLANKS AND H'S. LTABL(I,J) = BLANK INDICATES THAT CLASS

NUMBERS I AND J IN MAP 1 AND 2 CORRESPOND. THE INPUT MAPS 1 AND 2

SHOULD BE ON UNITS 8 AND 10 HAVING INTEGER*4 NUMBERS C THRU M AND

O THRU N, RESPECTIVELY (UNFORMATTED FORTRAN).

THE OUTPUTS OF THIS ROUTINE ARE:

1) PRINT OF THE JOINT HISTOGRAM, THE NUMBER AND PERCENTAGE

OF CORRECT CLASSIFICATIONS, ERRORS AT BOUNDARY POINTS AND

ERRORS AT INTERIOR POINTS;

2) DIFFERENCE MAP ON UNIT 12 CONTAINING CODES (BLANK, ., +

AND H) WRITTEN AS LOGICAL*1 RECORDS.

C

ISN 0003 DIMENSION IX(NEL,3),IY(NEL),IH(256)

ISN 0004 LOGICAL*1 LY(NEL),LTABL(M,N),BDY

ISN 0005 LOGICAL*1 ETCH/'/'/,PLUS/'+'/,BLANK/' '/,DOT/'.'/

ISN 0006 DATA IH/'/'/

ISN 0007 CALL CLAM(9,10,NREC,NEL,IH,IX,IY,M,N)

ISN 0008 REWIND 9

ISN 0009 REWIND 10

ISN 0010 PRINT 500

ISN 0011 .500 FORMAT(//50X'JOINT HISTOGRAM'//)

ISN 0012 CALL PRMXP(IH,M,N)

ISN 0013 CALL SVSCI(IH,256,5)

ISN 0014 CALL SARN(9,IX,NEL*4)

ISN 0015 CALL VMOV(IX,NEL,IX(1,2))

ISN 0016 DO 10 I=1,NREC

ISN 0017 IF(I.LT.NREC)CALL SARN(8,IX(1,3),NEL*4)

ISN 0018 READ(10)IY

ISN 0019 DO 20 J=1,NEL

ISN 0020 IF(IX(J,2).LE.0.OR.IY(J).LE.0)GO TO 30

ISN 0021 LY(J)=LTABL(IX(J,2),IY(J))

ISN 0022 IF(LY(J).EQ.BLANK)GO TO 35

ISN 0023 BDY=IX(J,2).NE.IX(J,1).OR.IX(J,2).NE.IX(J,3)

.OR.J.GT.1.AND.IX(J-1,2).NE.IX(J,2)

.OR.J.LT.NEL.AND.IX(J+1,2).NE.IX(J,2)

ISN 0024 IF(BDY)LY(J)=PLUS

ISN 0025 GO TO 35

ISN 0026 30 LY(J)=DOT

ISN 0027 35 IY=LY(J)

ISN 0028 IH(IY)=IH(IY)+1

ISN 0029 20 CONTINUE

ISN 0030 CALL VMOV(IX(1,2),NEL,IX(1,1))

ISN 0031 CALL VMOV(IX(1,3),NEL,IX(1,2))

ISN 0032 10 WRITE(12)LY

ISN 0033 NIDT=0

ISN 0034 DO 65 I=1,256

```
ISN 0039      LY(1)=1
ISN 0040      IF(LY(1).EQ.DOT)GO TO 65
ISN 0042      NTOT=NTOT+IH(1)
ISN 0043      65  CONTINUE
ISN 0044      FAC=100./NTOT
              C
              PRINT HISTOGRAM AND PERCENTAGE OCCUPENCIES.
              C
ISN 0045      DO 60 I=1,256
ISN 0046      IF(IH(I).EQ.0)GO TO 60
ISN 0048      PERCNT=IH(I)*FAC
ISN 0049      LY(1)=I
ISN 0050      IF(LY(1).EQ.DOT)GO TO 60
ISN 0052      PRINT 100,LY(1),IH(I),PERCNT
ISN 0053      60  CONTINUE
ISN 0054      100  FORMAT(/40XA1,10XI8,10XF7.2)
ISN 0055      PRINT 200
ISN 0056      200  FORMAT(/10X' ". "= EXTERIOR, " " = NO ERROR, "*" = ERROR AT BOUNDAR
              .Y POINT, "H" = ERROR AT NONBOUNDARY POINT')
ISN 0057      RETURN
ISN 0058      END
```

182

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

COMPILER OPTIONS - NAME= MAIN,OPT=C2,LINECNT=56,SIZE=000CK,
SOURCE,EPCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,NOXREF

```
ISN 0002 SUBROUTINE PRMXP(IA,M,N)
ISN 0003 DIMENSION IA(M,N)
ISN 0004 CALL MXINX(IA,M,N,M1,NJ)
ISN 0005 DO 50 I=1,M1
ISN 0006 50 PRINT 300,((IA(I,J),J=1,NJ)
ISN 0007 300 FORMAT(10X15I8)
ISN 0008 RETURN
ISN 0009 END
```

10. TEMPORAL CHANGE DETECTION AND DISPLAY

10.1 NAME

CHARADE

10.2 PURPOSE

To provide a difference image coded so as to indicate uniquely the class numbers in each of the two input images, and whether each pixel is an interior or boundary pixel; and to identify, display, and tabulate particular types of change.

10.3 CALLING SEQUENCE

Each of the two tasks described is controlled by a main program.

10.4 INPUT-OUTPUT

The inputs are certain parameters and the two data sets containing the classification maps. The parameters read are listed for the two tasks.

Difference Coding:

NSL: Number of scan lines in the map,
IS: number of samples per line,
M, N: maximum numbers on the two input map data sets,
ITAB: look-up table to reassign the class numbers in one of the maps,
IFLG: an integer which was added to the class numbers at all
boundary points (see program FLGBDRYS).

The first two of these parameters are read from cards; the remainder have been written on external storage by the program MXSMLRTY.

Change Detection:

NREC: number of records in the data,
NEL: number of elements per record,
M: number of classes in the maps,
IBUF: an array containing codes specifying the types of changes
to be detected and displayed.

The output is the difference coded map, a printer plot and external storage file depicting the changes, and an inventory of the changed pixels.

10.5 EXITS

Not Applicable

10.6 USAGE

The programs are written in FORTRAN and were compiled using the H compiler on the IBM 360. They are on the user library in executable form.

10.7 EXTERNAL INTERFACES

Certain other subroutines are called for generating the change depiction image on the printer, for I/O, and for vector manipulation. These are also available on the user library, and are the following subroutines:

PLTPIX
SARN
SAWN
SVSCI
VMOV

10.8 PERFORMANCE SPECIFICATIONS

Storage:

Difference coding - 42K bytes (K=1024)
Change detection - 56K bytes

Execution Time:

For map sizes 1624 records by 866 samples (1,406,384 pixels):

Difference coding - 1 minute
Change detection - 1 minute

I/O Load:

The difference coded image is placed on external storage for passing between steps. The change detection image is written on external storage for use in plotting on the printer.

10.9 METHOD

The two classification maps are read, class numbers are reassigned by table look-up if desired, and the difference coded map is written on external

storage. The user's request array is read to determine the type of change to be considered. The difference coded map is read in order to produce an inventory of the changes as well as a change depiction image. This output image is written on external storage and displayed as a printer plot.

10.10 COMMENTS

This section is devoted to a guide in the coding of the user's request array.

The user is expected to input his request of change detection and display through a data card (using the format (26I3)) which is read into the array IBUF by the system. Accordingly, the first entry into the array, IBUF (1), should be set to indicate the user's region of interest using the following code.

- IBUF (1) = 1: all regions (pixels) in the scene
- = 2: interior regions (pixels) only
- = 3: boundary pixels only

The second entry, IBUF (2), is set to denote the user's desire as to the type or category of change to be depicted using the ensuing code.

- IBUF (2) = 1: each change from each class (in the set to be defined for Map 1) to a corresponding individual class (in the set to be defined for Map 2).
- = 2: change from each class (in the set to be defined for Map 1) to every class (other than itself in the set to be defined for Map 2).
- = 3: change from the union of classes (in the set to be defined for Map 1) to the union of classes (in the set to be defined for Map 2) excepting of course those pixels that remain unchanged.
- = 4: change from the union of classes (in the set to be defined for Map 1) to individual classes (in the set to be defined for Map 2) disregarding of course the pixels that were originally in the corresponding class.
- = 5: change from individual classes (in the set to be defined for Map 1) to union of classes (in the set to be defined for Map 2) excepting those pixels that remain in the same class.

NC1=IBUF (3), the third entry defines the number of classes to be considered in Map 1 with IBUF (4) through IBUF (NC1+3) specifying the actual class numbers in the set of classes. However, if IBUF (3) is set to zero, then this - denotes that a sequential set of classes is to be considered and in that case IBUF (4) defines the number of classes to be defined internally in sequential order for Map 1. Similarly, the next entry NC2=IBUF (NEXT) (where

$$\begin{array}{ll} \text{NEXT} = 4+\text{NC1}, & \text{IBUF (3)} \neq 0 \\ & = 5, & = 0 \end{array}$$

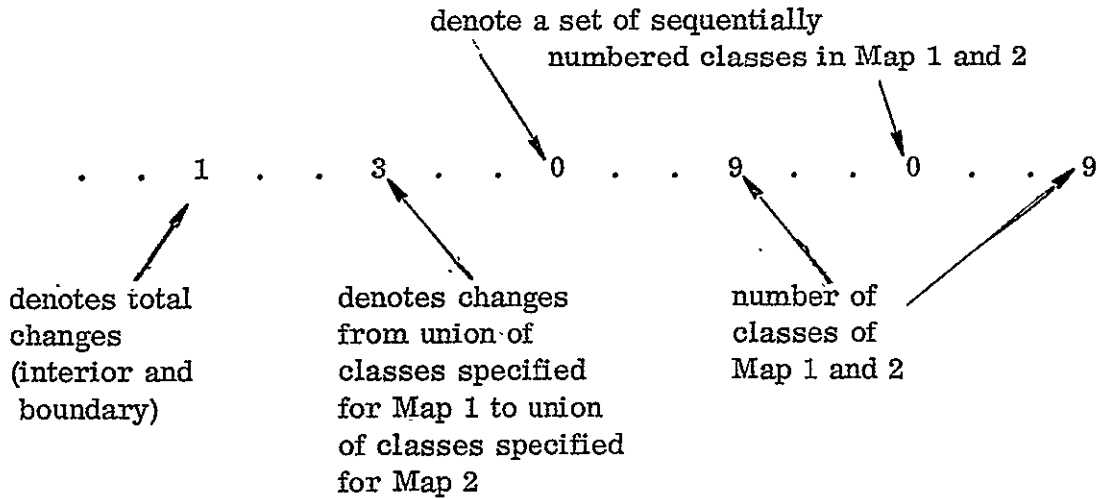
defines the number of classes to be considered in map 2) with IBUF (NEXT+1) to IBUF (NEXT+NC2) the actual class numbers in the second set. As before, if IBUF (NEXT) = 0, then this denotes that a sequential set of classes is to be considered in Map 2 and in that case IBUF (NEXT+1) defines the number of classes to be defined internally in sequential order for Map 2.



Typical Examples to illustrate the Code for input of User's request:

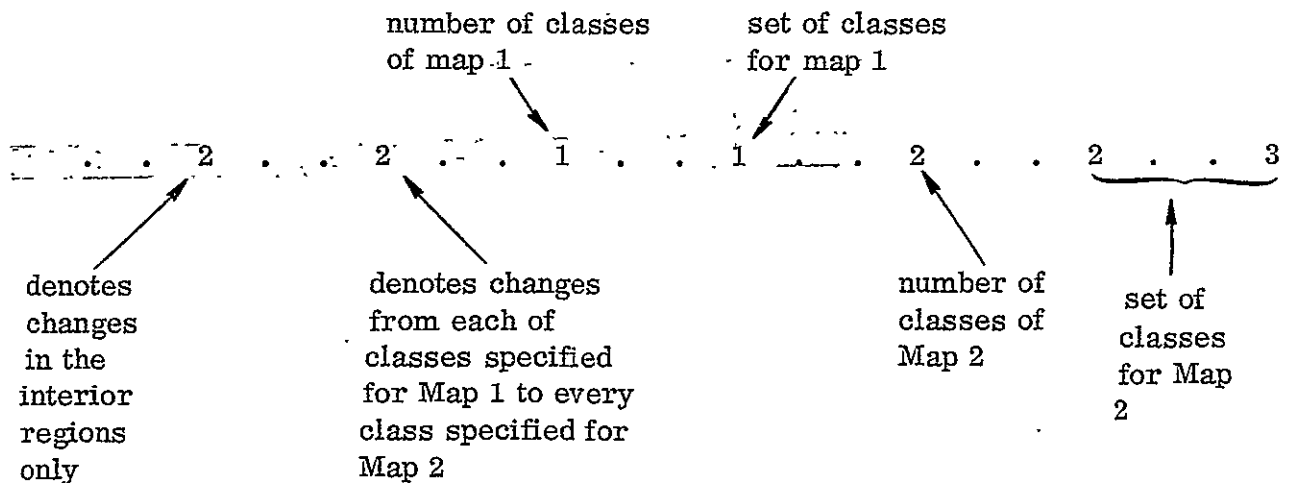
1. User's request: Depiction of the Union of Changes occurring in the entire scene from all 9 classes in Map 1 to all 9 classes in Map 2.

The code will be



2. User's request: Depiction of changes in the interior of Class 1 (Map 1) to Class 2 and 3 (Map 2) separately.

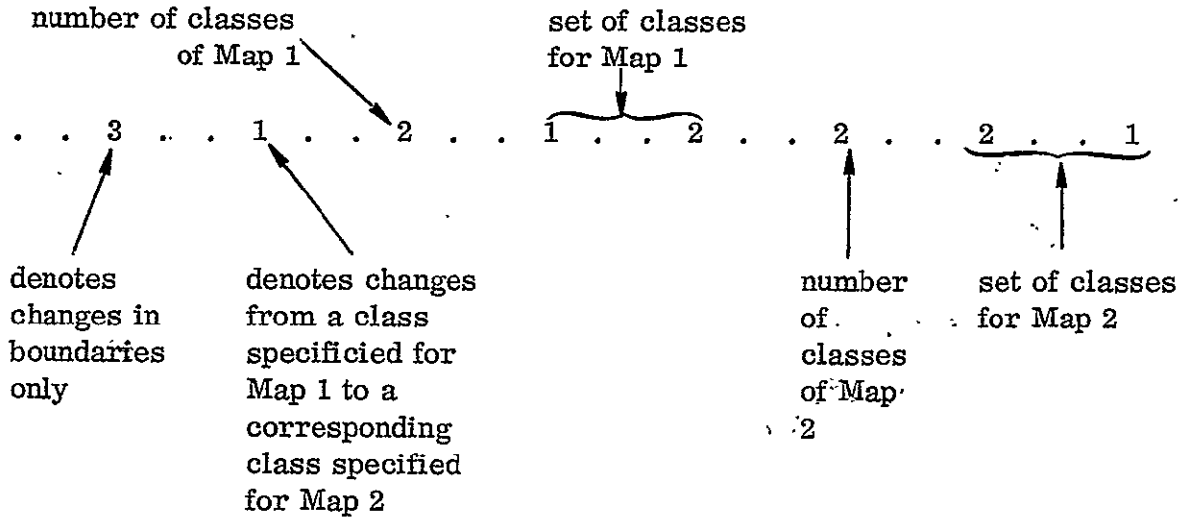
The code is



REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

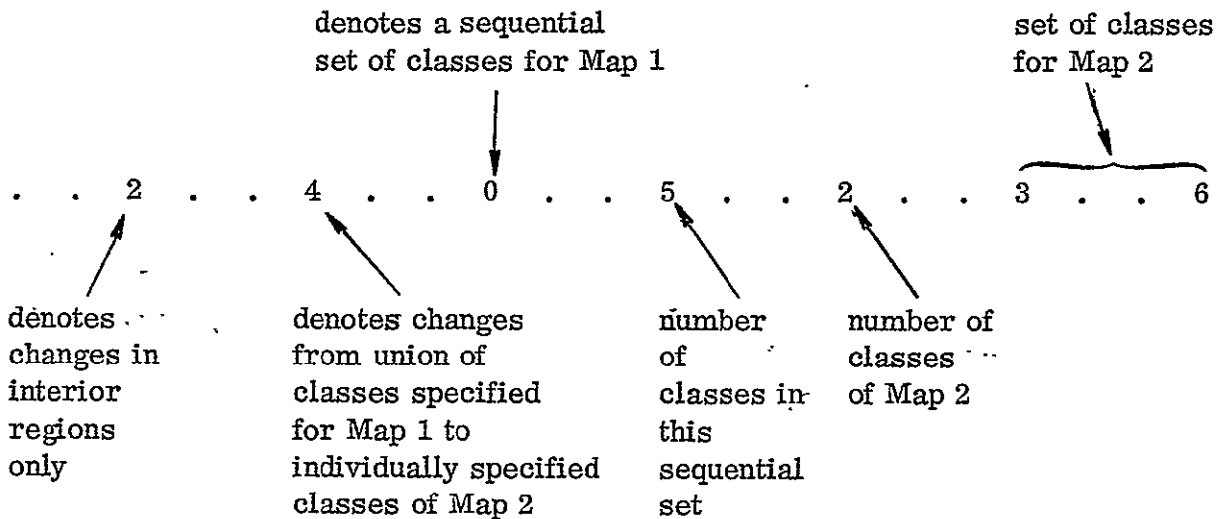
3. User's request: Depiction of changes in the boundaries from Class 1 to Class 2 and Class 2 to Class 1 separately.

The code for this case is:



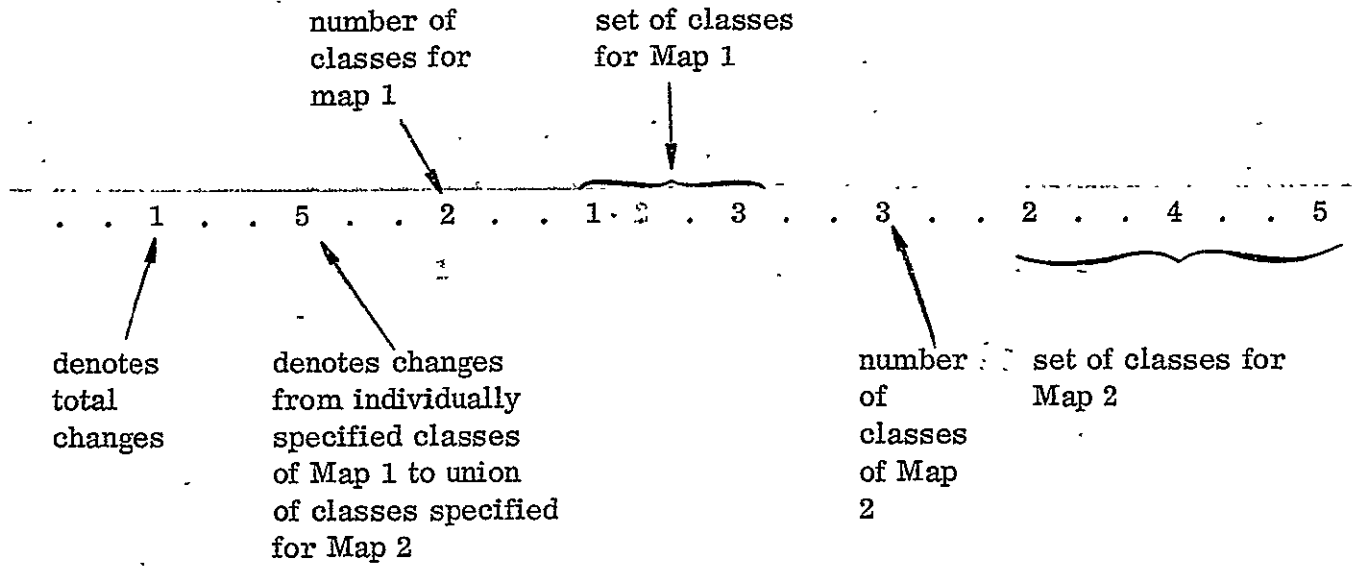
4. User's request: Depiction of changes occurring in the interior of classes 1 through 5 taken together into classes 3 and 6 individually.

The code for this case is:



5. User's request: Depiction of total changes from Class 1 and 3 individually to Classes 3, 4, and 5 taken together.

The code for this case is:



10.11 LISTING

```

DIMENSION IX(1000),IY(1000),ITAB(20)
DATA IN,JN,KN/8,9,10/
READ(5,10) NSL,IS
10  FORMAT(2I6)
    READ(1) M,N,{ITAB(I),I=1,N},IFLG
    M = M - IFLG
    WRITE(KN) M
    CALL      OVRMSK(IN,JN,KN,IX,IY,ITAB,NSL,IS,M)
100  FORMAT(10X,'DIFFERENCE MAP CODING HAS BEEN COMPLETED WITH M =',I3)
    WRITE(6,100) M
    STOP
    END

```

```

SUBROUTINE OVRMSK(IN,JN,KN,IX,IY,ITAB,NSL,IS,M)
DIMENSION IX(IS),IY(IS),ITAB(1)
C TO GENERATE A MASK WHICH INDICATES WHETHER A GIVEN PIXEL IS INTERIOR
C (OR BOUNDARY) AND THE CLASSES TO WHICH IT IS ASSIGNED IN THE TWO MAPS
C ON UNITS IN,JN. THE OUTPUT APPEARS ON KN. ITAB IS A TABLE INDICATING
C THE ASSIGNMENT OF CLASSES IN MAP 2 TO MAP 1.IFLG CONSTANT ADDED TO
C CLASS NUMBERS IN MAP 1 TO INDICATE THAT A GIVEN PIXEL IS ON THE
C BOUNDARY.
DO 11 I = 1,NSL
  READ(IN) IX
  READ(JN) IY
  DO 12 J = 1,IS
    IF(IX(J).EQ.0) GO TO 12
    IF(IY(J).LE.0) GO TO 13
  C IX(J).EQ.0 INDICATES THE POINT IS OUTSIDE THE REGION OF INTEREST
    IX(J) = (IX(J)-1)*M + ITAB(IY(J))
    GO TO 12
  13 IX(J) = ^
  12 CONTINUE
- 11 WRITE(KN) IX
  RETURN
  END

```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

```
DIMENSION IX(1000),IH(256),ISET(10),JSET(10),KSET(10),IBUF(26)
INTEGER TABLE(250)
LOGICAL #1 LX(1000)
CALL CHANGE(IX,LX,ISET,JSET,KSET,IH,IBUF,TABLE)
STOP
END
```

```
SUBROUTINE VNATS(IX,N)
DIMENSION IX(N)
DO 10 I = 1,N
10 IX(I) = I
RETURN
END
```

```

SUBROUTINE CHANGE(IX,LX,ISET,JSET,KSET,IH,IBUF,TABLE)
DIMENSION IX(1000),IH(256),ISET(10),JSET(10),KSET(10),IBUF(26)
LOGICAL *1 LX(1000)
INTEGER TABLE(250)
READ(5,10) NREC,NEL
READ(20) M
900 FORMAT(1H1,10X,'THE INPUT CHANGE DEPICTION REQUEST CODE IS:',26I3)
800 FORMAT(26I3)
10 FORMAT(2I6)
12 READ(5,800,END=1) IBUF
C DEFINE SETS OF CLASSES TO BE CONSIDERED
IF(IBUF(3).EQ.0) GO TO 30
NC1 = IBUF(3)
CALL VMOV(IBUF(4),NC1,ISET)
NEXT = NC1 + 4
GO TO 40
30 NC1 = IBUF(4)
CALL VNATS(ISET,NC1)
NEXT = 5
40 IF(IBUF(NEXT).EQ.0) GO TO 50
NC2 = IBUF(NEXT)
CALL VMOV(IBUF(NEXT+1),NC2,JSET)
GO TO 60
50 NC2 = IBUF(NEXT+1)
CALL VNATS(JSET,NC2)
60 DO 15 L = 1,26
I = 27-I
IF(IBUF(I).NE.0) GO TO 16
15 CONTINUE
16 WRITE(6,900) (IBUF(J),J=1,I)
C NOW, ISET(I),I=1,NC1 AND JSET(I),I=1,NC2 ARE THE CLASS NUMBERS USED
KC1 = NC1
KC2 = 1
IF(IBUF(2).EQ.3.OR.IBUF(2).EQ.4) KC1 = 1
IF(IBUF(2).EQ.2.OR.IBUF(2).EQ.4) KC2 = NC2
DO 70 I = 1,KC1
DO 70 J = 1,KC2
CALL SVSCI(IH,256,0)
CALL CREATE(I,J,IBUF,TABLE,ISET,JSET,NC1,NC2,M)
DO 80 L = 1,NREC
CALL SARN(20,IX,NEL*4)
DO 85 LL = 1,NEL
LX(LL) = TABLE(IX(LL) + 1)
MX = LX(LL) + 1
85 IH(MX) = IH(MX) + 1
CALL SAWN(8,LX,NEL)
80 CALL SAWN(10,LX,NEL)
REWIND 8
CALL PLTPIX(1,8,1,NREC,1,NEL,255,0,LX)
CALL PRTLBL(IBUF,ISET,JSET,KSET,NC1,NC2,I,J,IH)
REWIND 8
REWIND 20
READ(20)
70 CONTINUE
GO TO 2
1 CONTINUE
RETURN
END

```

```

SUBROUTINE PRTLBL(IBUF,ISET,JSET,KSET,NC1,NC2,I,J,IH)
DIMENSION IBUF(1),ISET(NC1),JSET(NC2)
DIMENSION IH(1),KSET(1)
201 FORMAT(10X,'CHANGE DEPICTION MAP SHOWING THE ENTIRE PIXEL SET OF')
202 FORMAT(10X,'CHANGE DEPICTION MAP SHOWING THE INTERIOR PIXELS OF')
203 FORMAT(10X,'CHANGE DEPICTION MAP SHOWING THE BOUNDARY PIXELS OF')
204 FORMAT(//,10X,'NO. OF PIXELS',5X,'SYMBOL',10X,'CHANGE')
301 FORMAT(1H+,63X,'MAP 1 - CLASS(ES):',15I3)
302 FORMAT(10X,I13,5X,'OVERPRINT',7X,'TO CLASS(ES):',15I3)
303 FORMAT(10X,'(EXCLUDING OFCOURSE THOSE PIXELS THAT REMAIN IN THEIR
.ORIGINAL CLASSES)')
304 FORMAT(10X,I13,9X,'I',12X,'NONE')
305 FORMAT(10X,I13,9X,'*',11X,'TO ALL OTHER CLASSES')
306 FORMAT(//,10X,'THE REST OF THE SCENE OF INTEREST IS SHOWN AS BLANK
.S WITH THE OUTSIDE OF THE SCENE DEPICTED BY PLUS SIGN')
IB1 = IBUF(1)
IB2 = IBUF(2)
GO TO (10,20,30),IB1
10 PRINT 201
GO TO 40
20 PRINT 202
GO TO 40
30 PRINT 203
40 GO TO (50,60,70,80,90),IB2
50 PRINT 301,ISET(I)
PRINT 204
IF(ISET(I).NE.JSET(I)) PRINT 302,IH(256),JSET(I)
GO TO 99
60 PRINT 301,ISET(I)
PRINT 204
IF(ISET(I).NE.JSET(J)) PRINT 302,IH(256),JSET(J)
GO TO 99
70 PRINT 301,(ISET(L),L=1,NC1)
PRINT 204
PRINT 302,IH(256),(JSET(L),L=1,NC2)
PRINT 303
GO TO 99
80 LK = 0
DO 93 L = 1,NC1
IF(ISET(L).EQ.JSET(J)) GO TO 93
LK = LK + 1
KSET(LK) = ISET(L)
93 CONTINUE
PRINT 301,(KSET(L),L=1,LK)
PRINT 204
PRINT 302,IH(256),JSET(J)
GO TO 99
90 LK = 0
PRINT 301,ISET(I)
PRINT 204
DO 94 L = 1,NC2
IF(JSET(L).EQ.ISET(I)) GO TO 94

```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR


```
LK = LK + 1
KSET(LK) = JSET(L)
94 CONTINUE
PRINT 302, IH(256), (KSET(L), L=1, LK)
9 PRINT 304, IH(81)
IF (IH(46).NE.0) PRINT 305, IH(46)
PRINT 306
ITOT = IH(46) + IH(256)+IH(81)
PCT = IH(256)*100.0/ITOT
WRITE(6,700) PCT
700 FORMAT(10X, 'CHANGE OF INTEREST:', F5.2, '%')
RETURN
END
```

```

SUBROUTINE CREATE(I,J,IBUF,TABLE,ISET,JSET,NC1,NC2,M)
DIMENSION IBUF(2),TABLE(1),ISET(NC1),JSET(NC2)
INTEGER TABLE
C   DIMENSION TABLE(MC33=MC1*MC2*2 +1)   WHERE MC1 AND MC2 ARE THE
C   NUMBER OF CLASSES IN MAP 1 AND 2
MS = (M+1)*M
MM = MS + M*M
IB1 = IBUF(1)
IB2 = IBUF(2)
TABLE(1) = 60
CALL SVSCI(TABLE(2),MM,C)
I2 = I
I3 = I
J2 = J
J3 = J
GO TO (20,30,40,40,60),IB2
20  J2 = I
    J3 = I
    GO TO 30
40  I2 = 1
    I3 = NC1
    IF(IB2.EQ.4) GO TO 30
60  J2 = 1
    J3 = NC2
30  CONTINUE
DO 12 I1 = I2,I3
L = (ISET(I1)-1)*M + 1
DO 11 J1 = 1,M
K = L + J1
IF(ISET(I1).EQ.J1) GO TO 15
IF(IB1.LT.3) TABLE(K) = 45
IF(IB1.NE.2) TABLE(K+MS) = 45
GO TO 11
15  CONTINUE
IF(IB1.LT.3) TABLE(K) = 80
IF(IB1.NE.2) TABLE(K+MS) = 80
11  CONTINUE
DO 10 J1 = J2,J3
K = L + JSET(J1)
IF(ISET(I1).EQ.JSET(J1)) GO TO 10
IF(IB1.LT.3) TABLE(K) = 255
IF(IB1.NE.2) TABLE(K+MS) = 255
10  CONTINUE
12  CONTINUE
RETURN
END

```

Part II

REFERENCES:

1. H. K. Ramapriyan, "GEOGREF - A Program for Determining the Parameters for Geographic Referencing", CSC Memorandum to File, 5E3090-1-8, July 30, 1976.
2. M. Lybanon, "Kohoutek Photometry Data Analysis Program Documentation - Partial Draft, CSC Memorandum to File, 5E3010-16, March 29, 1976.
3. H. K. Ramapriyan, "Geometric Correction of Remotely Sensed Images", CSC Memorandum to File 5E3030-1-1, September 31, 1976.
4. H. K. Ramapriyan, "Programs for Digital Manipulation of Curves such as Political Boundaries", CSC Memorandum to File, 5E3030-1-2-3, January 24, 1975.
5. A. Rosenfeld, Picture Processing by Computer, Academic Press, 1969.
6. R. R. Jayroe, Jr., "Evaluation Criteria for Software Classification Inventories, Accuracies and Maps", NASA TMX (in publication)

CSC

COMPUTER SCIENCES CORPORATION

Major Offices and Facilities Throughout the World