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

October 1976

CSC
COMPUTER SCIENCES CORPORATION
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
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

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2.</td>
<td>REGISTRATION AND GEOMETRIC CORRECTION</td>
<td>3</td>
</tr>
<tr>
<td>2.1</td>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>2.2</td>
<td>Effects Leading to Geometric Distortions of Images</td>
<td>4</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Effect of an Altitude Variation</td>
<td>4</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Effect Due to Varying Position Along Scan Line</td>
<td>6</td>
</tr>
<tr>
<td>2.3</td>
<td>Discussion</td>
<td>6</td>
</tr>
<tr>
<td>2.4</td>
<td>Examples</td>
<td>8</td>
</tr>
<tr>
<td>2.5</td>
<td>Geometric Correction</td>
<td>10</td>
</tr>
<tr>
<td>2.5.1</td>
<td>Statement of the Problem</td>
<td>10</td>
</tr>
<tr>
<td>2.5.2</td>
<td>Determination of Bounds on the Output Image</td>
<td>11</td>
</tr>
<tr>
<td>2.5.3</td>
<td>Modification of the Transformation</td>
<td>12</td>
</tr>
<tr>
<td>2.5.4</td>
<td>Computation of Input Image Coordinates</td>
<td>12</td>
</tr>
<tr>
<td>2.5.5</td>
<td>Computation of Output Pixel Values</td>
<td>13</td>
</tr>
<tr>
<td>2.5.6</td>
<td>Experimental Comparison of Transformation</td>
<td>13</td>
</tr>
<tr>
<td>3.</td>
<td>PROGRAM DESCRIPTIONS</td>
<td>14</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>14</td>
</tr>
<tr>
<td>3.2</td>
<td>Resource Requirements</td>
<td>14</td>
</tr>
<tr>
<td>3.3</td>
<td>Supervised Nearest-Neighbor Establishing Histogram Approach (SNEHA)</td>
<td>15</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Analysis Process</td>
<td>15</td>
</tr>
<tr>
<td>3.4</td>
<td>Parametric Recognition Imparting Trained Identification (PRITI)</td>
<td>16</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Analysis Process</td>
<td>16</td>
</tr>
<tr>
<td>3.5</td>
<td>Piecewise Linear Classifier</td>
<td>17</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Analysis Process</td>
<td>17</td>
</tr>
<tr>
<td>3.6</td>
<td>Performance Assessments</td>
<td>17</td>
</tr>
<tr>
<td>4.</td>
<td>CLASSIFICATION TECHNIQUE ASSESSMENT</td>
<td>19</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>19</td>
</tr>
<tr>
<td>4.2</td>
<td>Description of Data Sets</td>
<td>19</td>
</tr>
<tr>
<td>4.3</td>
<td>Preparation of the GTM</td>
<td>20</td>
</tr>
<tr>
<td>4.4</td>
<td>Comparisons with the GTM</td>
<td>24</td>
</tr>
<tr>
<td>4.4.1</td>
<td>HINDU Classifier</td>
<td>27</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Table Look-up (ELLTAB)</td>
<td>30</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Linear Sequential</td>
<td>30</td>
</tr>
<tr>
<td>4.4.4</td>
<td>Piecewise Linear (Histogram based)</td>
<td>30</td>
</tr>
<tr>
<td>4.4.5</td>
<td>Nearest Neighbor (Histogram based)</td>
<td>30</td>
</tr>
<tr>
<td>4.4.6</td>
<td>Maximum Likelihood (Histogram based)</td>
<td>30</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.5</td>
<td>Comparisons Between Classification Maps</td>
<td>39</td>
</tr>
<tr>
<td>4.6</td>
<td>Chi-Square Tests for a Typical Case</td>
<td>39</td>
</tr>
<tr>
<td>4.7</td>
<td>General Remarks</td>
<td>43</td>
</tr>
<tr>
<td>4.7.1</td>
<td>Qualifiers</td>
<td>43</td>
</tr>
<tr>
<td>4.7.2</td>
<td>Conclusions</td>
<td>43</td>
</tr>
<tr>
<td>4.7.3</td>
<td>Suggestions for Further Work</td>
<td>46</td>
</tr>
<tr>
<td>5.</td>
<td>TEMPORAL CHANGE DETECTION</td>
<td>47</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>47</td>
</tr>
<tr>
<td>5.2</td>
<td>Overview of the System</td>
<td>47</td>
</tr>
<tr>
<td>5.3</td>
<td>Description of the Change Detection System</td>
<td>49</td>
</tr>
<tr>
<td>5.4</td>
<td>Experimental Results</td>
<td>51</td>
</tr>
<tr>
<td>5.5</td>
<td>Discussion of Results and Concluding Remarks</td>
<td>51</td>
</tr>
<tr>
<td>6.</td>
<td>DATA COMPRESSION</td>
<td>58</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>58</td>
</tr>
<tr>
<td>6.2</td>
<td>The Karhunen-Loeve Transform</td>
<td>58</td>
</tr>
<tr>
<td>6.3</td>
<td>A Fast Karhunen-Loeve Transformation</td>
<td>59</td>
</tr>
<tr>
<td>6.4</td>
<td>Implementation of the Algorithm</td>
<td>62</td>
</tr>
<tr>
<td>6.5</td>
<td>Results</td>
<td>63</td>
</tr>
</tbody>
</table>

REFERENCES

PART II - I. INTRODUCTION

2. GEOMETRIC CORRECTION

3. THINNING OF BOUNDARY IMAGES

4. FINDING DISCONTINUITIES IN BOUNDARY DATA

5. SMOOTHING BOUNDARY DATA

6. IDENTIFICATION OF CONNECTED REGIONS

7. DELETION OF BOUNDARY POINTS

8. COMPUTATION OF CONTINGENCY MATRICES

9. COMPARISON OF SUPERVISED CLASSIFICATION MAPS

10. TEMPORAL CHANGE DETECTION AND DISPLAY

REFERENCES

197
LIST OF TABLES

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 $\chi^2$ Tests 44
Table 5.1 Temporal Change Detection Results 52
Table 6.1 Statistics of the Input and Output Images 65
**LIST OF ILLUSTRATIONS**

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

\[
\begin{align*}
    x &= (f \tan \eta) \sin \psi \\
    y &= (f \tan \eta) \sin \psi
\end{align*}
\]  

(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). \(\text{(Eq. (1) actually defines the instantaneous field of view of the sensor. Effects such as the skew produced as a result of the finite time required for the Landsat multispectral scanner to form an image are not included.)}\) The other quantities are defined as follows: \(f\) is a scale factor, \(\eta\) is the nadir angle subtended at the satellite by the great circle arc connecting the subpoint and the observed point and \(\psi\) 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 \(\eta\) is given by

\[
\tan \eta = \frac{R \sin \delta}{R (1 - \cos \delta) + H}
\]

(2)

where \(R\) is the Earth's radius, \(H\) is the orbital altitude, and \(\delta\) 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

\[
x = \frac{fR \sin \delta}{R (1 - \cos \delta) + H}
\]

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

\[
\frac{d\delta}{dH} = \frac{\sin \delta}{R (\cos \delta - 1) + H \cos \delta}
\]

(4)
Let \( u=R6 \) 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

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

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

\[
\frac{du}{dH} = \frac{u}{H} \tag{6}
\]

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

\[
v = C a^{-3/2} \tag{7}
\]

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

\[
\frac{dv}{v} = \frac{-3/2 \, da}{a} = \frac{-3/2 \, dH}{a} \tag{8}
\]
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 \( \frac{dx}{du} = \frac{1}{R} \frac{dx}{ds} \). From Eq. (3), it is given by

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

(9)

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

\[
\frac{dx}{du} = \frac{x}{H} \quad (10a)
\]

\[
x = \frac{f}{H} u = \text{(constant)} u \quad (10b)
\]

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 1 m — 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

<table>
<thead>
<tr>
<th>Date of Observation</th>
<th>Number of GCPs*</th>
<th>6-Parameter Linear Transformation</th>
<th>8-Parameter Nonlinear Transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Error at Fit Points</td>
<td>All Points</td>
</tr>
<tr>
<td>10/17/72</td>
<td>36</td>
<td>1.7912</td>
<td>1.7912</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>0.30430</td>
<td>1.9095</td>
</tr>
<tr>
<td>11/17/73</td>
<td>36</td>
<td>1.3308</td>
<td>1.3308</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>0.73284</td>
<td>1.4361</td>
</tr>
<tr>
<td>12/5/73</td>
<td>36</td>
<td>1.3264</td>
<td>1.3264</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>0.58774</td>
<td>1.4614</td>
</tr>
</tbody>
</table>

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

\[
\begin{bmatrix}
X \\
Y
\end{bmatrix} = A \begin{bmatrix}
U \\
V
\end{bmatrix} + \begin{bmatrix}
X_0 \\
Y_0
\end{bmatrix} + \begin{bmatrix}
\alpha \\
\beta
\end{bmatrix} UV
\]  

(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

\[
1 \leq X \leq X; \quad 1 \leq Y \leq Y
\]

(12)

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

\[
U \leq U \leq U; \quad V \leq V \leq V
\]

(13)

where \(U, U, V, V\) 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 $U, \overline{U}, V, \overline{V}$.

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 $U, \overline{U}, V, \overline{V}$ 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 $U, \overline{U}, V, \overline{V}$ are simply found by using

$$
\begin{bmatrix}
U \\
V
\end{bmatrix} = A^{-1} \begin{bmatrix} X - X_o \\
Y - Y_o \end{bmatrix}
$$

(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:

$$(a_{11} \beta - a_{21} \alpha)U^2 + [a_{11}a_{22} - a_{12}a_{21}] + \alpha(Y - Y_o) - \beta(X - X_o)]U$$

$$+ [a_{12}(Y - Y_o) - a_{22}(X - X_o)] = 0$$

(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, \( U, \overline{U}, \overline{V}, \overline{V} \) are obtained by finding the extremal values of \( U \) and \( V \) over the sets

\[
\begin{align*}
X &= 1; \quad 1 \leq Y \leq \overline{Y} \\
X &= \overline{X}; \quad 1 \leq Y \leq \overline{Y} \\
1 \leq X \leq \overline{X}; \quad Y = 1 \\
1 \leq X \leq \overline{X}; \quad Y = \overline{Y}
\end{align*}
\]  

(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

\[
R_1 = \text{Max} \left[ a_{12} + \alpha \overline{U} \left( \overline{V} - \overline{Y} \right), a_{12} + \alpha \overline{U} \left( \overline{V} - \overline{Y} \right) \right]
\]

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

\[
R_2 = \text{Max} \left[ a_{11} = \alpha \overline{V} \left( \overline{U} - \overline{U} \right), a_{11} = \alpha \overline{V} \left( \overline{U} - \overline{U} \right) \right]
\]

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

Also, it is desirable to have the output record and sample numbers increase with respect to those in the input image. Since \( \alpha \) and \( \beta \) 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 \( A \), \( \alpha \), \( \beta \), \( U \), \( U \) are negated and \( U, \overline{U} \) are interchanged. If \( a_{22} < 0 \), then the second column of \( A \), \( \alpha \), \( \beta \), \( V \), \( \overline{V} \) are negated and \( \overline{V}, \overline{V} \) 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 \( \overline{V} - \overline{Y} + 1 \) 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 \( \overline{X} \). If not, the \( Y \) value is set to 0 without any computation.
2.5.5 Computation of Output Pixel Values

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

\[
R_1 = \min \{X_i \mid 1 \leq i \leq N; Y_i \neq 0\}
\]
\[
R_2 = \max \{X_i \mid 1 \leq i \leq N; Y_i \neq 0\}.
\]

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 $1.4 \times 10^6$ 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, $\chi^2$-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

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

*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 GTMi 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

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

Table 4.3. Affine Transformation Parameters

<table>
<thead>
<tr>
<th>Date</th>
<th>No. of GCP's</th>
<th>PARAMETERS</th>
<th>RMS Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(a_{11})</td>
<td>(a_{12})</td>
</tr>
<tr>
<td>Oct. 17, 72</td>
<td>18</td>
<td>-2.4037</td>
<td>-12.2023</td>
</tr>
<tr>
<td>Nov. 17, 73</td>
<td>20</td>
<td>-2.3846</td>
<td>-12.2867</td>
</tr>
<tr>
<td>Dec. 5, 73</td>
<td>25</td>
<td>-2.3416</td>
<td>-12.3164</td>
</tr>
<tr>
<td>Apr. 10, 74</td>
<td>17</td>
<td>-2.3316</td>
<td>-12.1802</td>
</tr>
<tr>
<td>Jun. 21, 74</td>
<td>24</td>
<td>-2.4133</td>
<td>-12.2759</td>
</tr>
<tr>
<td>Jan. 5, 75</td>
<td>24</td>
<td>-2.3975</td>
<td>-12.3053</td>
</tr>
</tbody>
</table>
Table 4.4. Information for Registration w. r. t. GTM

<table>
<thead>
<tr>
<th>Date</th>
<th>Coordinates Matching (1, 1) on GTM*</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Row</td>
</tr>
<tr>
<td>Oct. 17, 72</td>
<td>248</td>
</tr>
<tr>
<td>Nov. 17, 73</td>
<td>336</td>
</tr>
<tr>
<td>Dec. 5, 73</td>
<td>345</td>
</tr>
<tr>
<td>April 10, 74</td>
<td>430</td>
</tr>
<tr>
<td>June 21, 74</td>
<td>431</td>
</tr>
<tr>
<td>Jan. 5, 75</td>
<td>396</td>
</tr>
</tbody>
</table>

*The GTM here refers to GTM10 described in Section 4.3 (Tape Number A0944)
<table>
<thead>
<tr>
<th>Date</th>
<th>Raw Data</th>
<th>HINDUCM</th>
<th>SNEHACM</th>
<th>PRITICM</th>
<th>LCM</th>
<th>PCM</th>
<th>ETCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oct. 17, 72</td>
<td>A0620</td>
<td>A0869</td>
<td>A0969</td>
<td>A0557</td>
<td>A0912</td>
<td>A0575</td>
<td>A0147</td>
</tr>
<tr>
<td>Nov. 17, 73</td>
<td>A0671</td>
<td>A0784</td>
<td>A1209</td>
<td>A0835</td>
<td>A0492</td>
<td>A0894</td>
<td>A1437</td>
</tr>
<tr>
<td>Dec. 5, 73</td>
<td>A0063</td>
<td>A0286</td>
<td>A0200</td>
<td>A0676</td>
<td>A0633</td>
<td>A0820</td>
<td>A0822</td>
</tr>
<tr>
<td>Apr. 10, 74</td>
<td>A1325</td>
<td>A0966</td>
<td>A0197</td>
<td>A0259</td>
<td>A0495</td>
<td>A0209</td>
<td>A0438</td>
</tr>
<tr>
<td>June 21, 74</td>
<td>A1287</td>
<td>A0868</td>
<td>A0846</td>
<td>A0626</td>
<td>A0374</td>
<td>A0440</td>
<td>A0331</td>
</tr>
<tr>
<td>Jan. 5, 75</td>
<td>A0828</td>
<td>A0109</td>
<td>A0016</td>
<td>A0279</td>
<td>A0287</td>
<td>A1388</td>
<td>A1368</td>
</tr>
</tbody>
</table>
Now, GTM2 had boundary lines with thickness greater than one pixel. (While thinner boundary lines could have been obtained by digitizing at 50μ 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>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Preliminary map (with disclaimer) supplied by U. S. Geological Survey.</td>
</tr>
<tr>
<td>1</td>
<td>Annotations removed and photographically reduced.</td>
</tr>
<tr>
<td>2</td>
<td>Digitized at 25 μ scanning interval.</td>
</tr>
<tr>
<td>3</td>
<td>Boundary lines thinned.</td>
</tr>
<tr>
<td>4</td>
<td>Regions identified by unique numbers (RIN) based on connectivity.</td>
</tr>
<tr>
<td>5</td>
<td>Discontinuities in boundary lines patched.</td>
</tr>
<tr>
<td>6</td>
<td>Regions identified by unique numbers (RIN) based on connectivity.</td>
</tr>
<tr>
<td>7</td>
<td>RIN's replaced by CIN's using a manually determined correspondence table with regions in doubt flagged with CIN = -1.</td>
</tr>
<tr>
<td>8</td>
<td>Geometrically corrected to the orientation of UTM coordinates with 20 pixels/km.</td>
</tr>
<tr>
<td>9</td>
<td>Reduced to Level I map by modifying the CIN's.</td>
</tr>
<tr>
<td>10</td>
<td>Boundary points replaced by CIN's occurring the largest number of times in their immediate neighborhoods.</td>
</tr>
</tbody>
</table>
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, Wetland 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

\[ \frac{\text{Tr} (A)}{S (A)} \]

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

and \( S (A) = \text{Sum of all elements of A.} \)

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 \( \frac{1 - \sum |p_{1i} - p_{2i}|}{\sum p_{1i}} \times 100 \) 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)

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR
Table 4.7. Joint Histogram Between GTM and HINDU Cluster Map

<table>
<thead>
<tr>
<th>CLASS NO.</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>72346</td>
<td>35816</td>
<td>2679</td>
<td>378</td>
<td>2814</td>
<td>1567</td>
<td>237</td>
<td>5093</td>
</tr>
<tr>
<td>2</td>
<td>81189</td>
<td>81745</td>
<td>203</td>
<td>31</td>
<td>6634</td>
<td>845</td>
<td>11401</td>
<td>1742</td>
</tr>
<tr>
<td>3</td>
<td>464658</td>
<td>47892</td>
<td>300</td>
<td>367</td>
<td>2315</td>
<td>3562</td>
<td>1632</td>
<td>1652</td>
</tr>
<tr>
<td>4</td>
<td>4439</td>
<td>1631</td>
<td>15025</td>
<td>18432</td>
<td>62</td>
<td>1220</td>
<td>10</td>
<td>1230</td>
</tr>
<tr>
<td>5</td>
<td>78792</td>
<td>3436</td>
<td>2656</td>
<td>502</td>
<td>42</td>
<td>8899</td>
<td>259</td>
<td>297</td>
</tr>
<tr>
<td>6</td>
<td>4856</td>
<td>2006</td>
<td>400</td>
<td>84</td>
<td>331</td>
<td>100</td>
<td>100</td>
<td>1912</td>
</tr>
</tbody>
</table>

GTM V/S HINDUCM(010575)

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

<table>
<thead>
<tr>
<th>Date</th>
<th>No. of Classes</th>
<th>Similarity Measures (%)</th>
<th>Inventory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Normal</td>
<td>Boundaries Ignored</td>
</tr>
<tr>
<td>Oct. 17, 72</td>
<td>9</td>
<td>67.42</td>
<td>69.90</td>
</tr>
<tr>
<td>Nov. 17, 73</td>
<td>8</td>
<td>70.96</td>
<td>73.54</td>
</tr>
<tr>
<td>Dec. 5, 73</td>
<td>9</td>
<td>73.37</td>
<td>76.08</td>
</tr>
<tr>
<td>Apr. 10, 74</td>
<td>8</td>
<td>69.67</td>
<td>72.39</td>
</tr>
<tr>
<td>June 21, 74</td>
<td>10</td>
<td>69.06</td>
<td>71.65</td>
</tr>
<tr>
<td>Jan. 5, 75</td>
<td>8</td>
<td>71.38</td>
<td>74.06</td>
</tr>
</tbody>
</table>
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 ELLTABC (Dec. 1973)
Table 4-9. Joint Histogram Between GTM and ELLTAB Classification Map

<table>
<thead>
<tr>
<th>CLASS NO.</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>39615</td>
<td>26065</td>
<td>40534</td>
<td>2111</td>
<td>7151</td>
<td>2582</td>
<td>2982</td>
</tr>
<tr>
<td>2</td>
<td>37030</td>
<td>97570</td>
<td>29733</td>
<td>19</td>
<td>6414</td>
<td>185</td>
<td>2339</td>
</tr>
<tr>
<td>3</td>
<td>28758</td>
<td>47429</td>
<td>409922</td>
<td>391</td>
<td>32377</td>
<td>199</td>
<td>3311</td>
</tr>
<tr>
<td>4</td>
<td>1592</td>
<td>342</td>
<td>915</td>
<td>393754</td>
<td>5394</td>
<td>1309</td>
<td>3796</td>
</tr>
<tr>
<td>5</td>
<td>8263</td>
<td>4384</td>
<td>35277</td>
<td>2753</td>
<td>43547</td>
<td>270</td>
<td>1880</td>
</tr>
<tr>
<td>6</td>
<td>2456</td>
<td>1596</td>
<td>2818</td>
<td>350</td>
<td>846</td>
<td>1039</td>
<td>1404</td>
</tr>
</tbody>
</table>

GTM V/S ELLTABCM(12C573)
Table 4.10. Joint Histogram Between GTM and Linear Sequential Classification Map

<table>
<thead>
<tr>
<th>CLASS NO.</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>43249</td>
<td>47676</td>
<td>21108</td>
<td>3125</td>
<td>4462</td>
<td>1411</td>
</tr>
<tr>
<td>2</td>
<td>23872</td>
<td>133034</td>
<td>22448</td>
<td>31</td>
<td>3445</td>
<td>960</td>
</tr>
<tr>
<td>3</td>
<td>41336</td>
<td>105547</td>
<td>259656</td>
<td>564</td>
<td>14984</td>
<td>291</td>
</tr>
<tr>
<td>4</td>
<td>1183</td>
<td>564</td>
<td>1199</td>
<td>38163</td>
<td>5664</td>
<td>1353</td>
</tr>
<tr>
<td>5</td>
<td>5991</td>
<td>3239</td>
<td>47075</td>
<td>3869</td>
<td>35515</td>
<td>92</td>
</tr>
<tr>
<td>6</td>
<td>1957</td>
<td>3526</td>
<td>2589</td>
<td>521</td>
<td>622</td>
<td>564</td>
</tr>
</tbody>
</table>

GTM V/S LCM(041074)

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

<table>
<thead>
<tr>
<th>Date</th>
<th>Similarity Measures (%)</th>
<th>Normal</th>
<th>Boundaries Ignored</th>
<th>Inventory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oct. 17, 72</td>
<td></td>
<td>65.77</td>
<td>68.57</td>
<td>85.75</td>
</tr>
<tr>
<td>Nov. 17, 73</td>
<td></td>
<td>72.00</td>
<td>74.95</td>
<td>96.99</td>
</tr>
<tr>
<td>Dec. 5, 73</td>
<td></td>
<td>73.85</td>
<td>75.81</td>
<td>97.22</td>
</tr>
<tr>
<td>Apr. 10, 74</td>
<td></td>
<td>72.15</td>
<td>75.14</td>
<td>91.75</td>
</tr>
<tr>
<td>June 21, 74</td>
<td></td>
<td>65.35</td>
<td>67.88</td>
<td>85.99</td>
</tr>
<tr>
<td>Jan. 5, 75</td>
<td></td>
<td>71.02</td>
<td>73.90</td>
<td>93.94</td>
</tr>
</tbody>
</table>
Figure 4.5. Mobile Bay PCM (Oct. 1972)
### Table 4.12. Joint Histogram Between GTM and Piecewise Linear Classification Map

<table>
<thead>
<tr>
<th>CLASS NO.</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>52834</td>
<td>4019</td>
<td>16241</td>
<td>3406</td>
<td>6316</td>
<td>2143</td>
</tr>
<tr>
<td>2</td>
<td>61697</td>
<td>9208</td>
<td>11507</td>
<td>2682</td>
<td>9699</td>
<td>6107</td>
</tr>
<tr>
<td>3</td>
<td>97128</td>
<td>130587</td>
<td>269307</td>
<td>7941</td>
<td>8107</td>
<td>8908</td>
</tr>
<tr>
<td>4</td>
<td>2430</td>
<td>1073</td>
<td>869</td>
<td>376362</td>
<td>5878</td>
<td>19554</td>
</tr>
<tr>
<td>5</td>
<td>18520</td>
<td>16070</td>
<td>26893</td>
<td>9230</td>
<td>18023</td>
<td>7605</td>
</tr>
<tr>
<td>6</td>
<td>2385</td>
<td>3887</td>
<td>1030</td>
<td>546</td>
<td>482</td>
<td>1613</td>
</tr>
</tbody>
</table>

**GTM V/S PCM (101772)**

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

<table>
<thead>
<tr>
<th>Date</th>
<th>Normal</th>
<th>Boundaries Ignored</th>
<th>Inventory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oct. 17, 72</td>
<td>60.45</td>
<td>63.15</td>
<td>81.30</td>
</tr>
<tr>
<td>Nov. 17, 73</td>
<td>70.08</td>
<td>72.65</td>
<td>87.37</td>
</tr>
<tr>
<td>Dec. 5, 73</td>
<td>73.42</td>
<td>76.32</td>
<td>97.34</td>
</tr>
<tr>
<td>Apr. 10, 74</td>
<td>70.91</td>
<td>73.69</td>
<td>87.55</td>
</tr>
<tr>
<td>June 21, 74</td>
<td>68.41</td>
<td>71.00</td>
<td>96.19</td>
</tr>
<tr>
<td>Jan. 5, 75</td>
<td>69.76</td>
<td>72.52</td>
<td>90.02</td>
</tr>
</tbody>
</table>
Figure 4.6. Mobile Bay SNEHACM (Nov. 1973)
Table 4.14. Joint Histogram Between GTM and Nearest Neighbor Classification Map

<table>
<thead>
<tr>
<th>CLASS NO.</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18113</td>
<td>21173</td>
<td>60411</td>
<td>2036</td>
<td>6416</td>
<td>12832</td>
</tr>
<tr>
<td>2</td>
<td>32216</td>
<td>63975</td>
<td>68203</td>
<td>17</td>
<td>4572</td>
<td>14807</td>
</tr>
<tr>
<td>3</td>
<td>19226</td>
<td>20367</td>
<td>463351</td>
<td>345</td>
<td>13918</td>
<td>5177</td>
</tr>
<tr>
<td>4</td>
<td>1294</td>
<td>1518</td>
<td>1721</td>
<td>387582</td>
<td>11437</td>
<td>2358</td>
</tr>
<tr>
<td>5</td>
<td>2104</td>
<td>2270</td>
<td>50547</td>
<td>4110</td>
<td>37109</td>
<td>525</td>
</tr>
<tr>
<td>6</td>
<td>1819</td>
<td>1367</td>
<td>4108</td>
<td>596</td>
<td>690</td>
<td>2228</td>
</tr>
</tbody>
</table>

GTM V/S SNEHACM (111773)

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

<table>
<thead>
<tr>
<th>Date</th>
<th>Similarity Measures (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Normal</td>
</tr>
<tr>
<td>Oct. 17, 72</td>
<td>59.27</td>
</tr>
<tr>
<td>Nov. 17, 73</td>
<td>72.56</td>
</tr>
<tr>
<td>Dec. 5, 73</td>
<td>66.22</td>
</tr>
<tr>
<td>April 10, 74</td>
<td>55.28</td>
</tr>
<tr>
<td>June 21, 74</td>
<td>58.71</td>
</tr>
<tr>
<td>Jan. 5, 75</td>
<td>55.18</td>
</tr>
</tbody>
</table>
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 $\chi^2$ 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.
Table 4.16. Joint Histogram Between GTM and Histogram-based
Maximum Likelihood Classification Map

<table>
<thead>
<tr>
<th>CLASS NO.</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>27870</td>
<td>9586</td>
<td>68571</td>
<td>1893</td>
<td>8735</td>
<td>4380</td>
</tr>
<tr>
<td>2</td>
<td>27822</td>
<td>63236</td>
<td>84056</td>
<td>82</td>
<td>7368</td>
<td>1226.</td>
</tr>
<tr>
<td>3</td>
<td>21806</td>
<td>24030</td>
<td>429624</td>
<td>2726</td>
<td>37259</td>
<td>7933</td>
</tr>
<tr>
<td>4</td>
<td>10474</td>
<td>73</td>
<td>1651</td>
<td>373863</td>
<td>9604</td>
<td>1019</td>
</tr>
<tr>
<td>5</td>
<td>2692</td>
<td>858</td>
<td>47654</td>
<td>4666</td>
<td>39343</td>
<td>522</td>
</tr>
<tr>
<td>6</td>
<td>2402</td>
<td>732</td>
<td>3446</td>
<td>446</td>
<td>1591</td>
<td>1179</td>
</tr>
</tbody>
</table>

GTM V/S PRITICM(062174)

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

<table>
<thead>
<tr>
<th>Date</th>
<th>Similarity Measures (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Normal</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Oct. 17, 72</td>
<td>62.98</td>
</tr>
<tr>
<td>Nov. 17, 73</td>
<td>71.22</td>
</tr>
<tr>
<td>Dec. 5, 73</td>
<td>73.45</td>
</tr>
<tr>
<td>April 10, 74</td>
<td>70.75</td>
</tr>
<tr>
<td>June 21, 74</td>
<td>70.77</td>
</tr>
<tr>
<td>Jan. 5, 75</td>
<td>69.01</td>
</tr>
</tbody>
</table>
Table 4.18. Similarity Measures between CM's for Dec. 73 Data Set

<table>
<thead>
<tr>
<th>Map 1</th>
<th>Map 2</th>
<th>Similarity Measure (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETCM</td>
<td>LCM</td>
<td>89.07</td>
</tr>
<tr>
<td>ETCM</td>
<td>PCM</td>
<td>85.78</td>
</tr>
<tr>
<td>ETCM</td>
<td>PRITICM</td>
<td>86.07</td>
</tr>
<tr>
<td>SNEHACM</td>
<td>LCM</td>
<td>82.09</td>
</tr>
<tr>
<td>SNEHACM</td>
<td>PCM</td>
<td>85.44</td>
</tr>
</tbody>
</table>
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 $x^2$ 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 $x^2$ Tests

<table>
<thead>
<tr>
<th>CLASSIFICATION ACCLTACY (PERCENT):</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RANDOM</td>
<td>24.26</td>
</tr>
<tr>
<td>ACTUAL</td>
<td>65.77</td>
</tr>
<tr>
<td>OPTIMUM (INVENTORY)</td>
<td>85.75</td>
</tr>
</tbody>
</table>

PERCENT OF OPTIMUM ACCURACY = 67.45

<table>
<thead>
<tr>
<th>TEST</th>
<th>FEATURE</th>
<th>D.C.F.</th>
<th>CHI-SQUARE TESTS</th>
<th>CHI-SQUARE</th>
<th>FREQUENCY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>2.80764166e-05</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>2.50451250e-05</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>4.37546562e-05</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>5.24296641e-05</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1 5</td>
<td>5.7624375e-05</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2 5</td>
<td>2.49255575e-05</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>3 5</td>
<td>3.31562063e-05</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>4 5</td>
<td>6.27563375e-05</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>5 5</td>
<td>1.7166125e-05</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>5 5</td>
<td>5.96310347e-03</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>6 5</td>
<td>1.65646632e-05</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>2 5</td>
<td>2.7482062e-05</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>3 5</td>
<td>2.7371375e-05</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>4 5</td>
<td>8.2362273e-05</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>5 5</td>
<td>1.19564382e-05</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>6 5</td>
<td>1.8215553e-05</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>1 5</td>
<td>1.24467222e-06</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>1 5</td>
<td>5.75363072e-04</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>2 5</td>
<td>2.1415956e-05</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>3 5</td>
<td>1.49260875e-05</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>4 5</td>
<td>4.4257556e-05</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>5 5</td>
<td>1.7135153e-03</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>6 5</td>
<td>5.1365755e-06</td>
<td>C.C</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 viewpoints 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
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

\[
\text{IPIXEL} (\cdot) = (\text{IX} (\cdot) - 1) \times M + \text{LCC} (\text{IY} (\cdot))
\]

where \(\text{IX} (\cdot)\) 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, \(\text{IY} (\cdot)\) 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 \(\text{LCC} (\cdot)\) 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

<table>
<thead>
<tr>
<th>No.</th>
<th>Type of Change</th>
<th>DATA SETS CONSIDERED</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>INTER-SEASONAL</td>
<td>INTRA-SEASONAL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>No. of pixels changed</td>
<td>Percentage change</td>
<td>No. of pixels changed</td>
</tr>
<tr>
<td>1.</td>
<td>Total changes in all classes</td>
<td>442013</td>
<td>31.99</td>
<td>322484</td>
</tr>
<tr>
<td>2.</td>
<td>Changes in interior regions and all classes</td>
<td>158100</td>
<td>18.22</td>
<td>73566</td>
</tr>
<tr>
<td>3.</td>
<td>Changes in the boundaries of all classes</td>
<td>283913</td>
<td>55.24</td>
<td>248913</td>
</tr>
<tr>
<td>4.</td>
<td>Additions to class 1 (URBAN)</td>
<td>80745</td>
<td>5.84</td>
<td>67516</td>
</tr>
<tr>
<td>5.</td>
<td>Deletions from the interior of class 3 (FOREST)</td>
<td>112692</td>
<td>33.49</td>
<td>37738</td>
</tr>
<tr>
<td>6.</td>
<td>Changes in the boundaries of class 4 (WATER)</td>
<td>4500</td>
<td>40.77</td>
<td>4938</td>
</tr>
</tbody>
</table>
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)
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:

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

where the sequence $\{\varepsilon_i\}$ 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 $\lambda_i$ 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]

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

where $\lambda_j^2 = \frac{1-p^2}{1-2p \cos \omega_i + p^2}$ 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 $\omega$ 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

$$x_i = \tilde{x}_i + \tilde{v}_i$$

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

$$x_i = \frac{p}{1+p^2} (x_{i+1} + x_{i-1}) + \tilde{v}_i.$$
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 ($N \times N$, tridiagonal) has the form $B^2Q$ where

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

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

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

$$Q \mathbf{X} = \mathbf{V} + \mathbf{B}$$

where $\mathbf{B}$ 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

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

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

$$WQ\mathbf{X} = \mathbf{WV} + \mathbf{WB},$$

or

$$\hat{\mathbf{X}} = \frac{\mathbf{V}}{\lambda} + \frac{\mathbf{B}}{\lambda} ,$$

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

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

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.
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 \( \hat{x}_{ij} = \frac{\hat{v}_{ij}}{\lambda_{ij}} \).

(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

<table>
<thead>
<tr>
<th></th>
<th>Input Image</th>
<th>Output Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>13.00</td>
<td>10.07</td>
</tr>
<tr>
<td>Maximum</td>
<td>82.00</td>
<td>67.01</td>
</tr>
<tr>
<td>Range</td>
<td>69.00</td>
<td>56.94</td>
</tr>
<tr>
<td>Mean</td>
<td>20.63</td>
<td>20.50</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>4.00</td>
<td>4.03</td>
</tr>
</tbody>
</table>

Mean Squared Error Between Images 2.34
Figure 6.2. Histogram of Input Image.
<table>
<thead>
<tr>
<th>Level</th>
<th>Density Level</th>
<th>Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>5210</td>
</tr>
<tr>
<td>5</td>
<td>17</td>
<td>8200</td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>7720</td>
</tr>
<tr>
<td>7</td>
<td>19</td>
<td>7915</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td>7992</td>
</tr>
<tr>
<td>9</td>
<td>21</td>
<td>7968</td>
</tr>
<tr>
<td>10</td>
<td>22</td>
<td>7971</td>
</tr>
<tr>
<td>11</td>
<td>23</td>
<td>5153</td>
</tr>
<tr>
<td>12</td>
<td>24</td>
<td>3563</td>
</tr>
<tr>
<td>13</td>
<td>25</td>
<td>2539</td>
</tr>
<tr>
<td>14</td>
<td>26</td>
<td>1426</td>
</tr>
<tr>
<td>15</td>
<td>27</td>
<td>1503</td>
</tr>
<tr>
<td>16</td>
<td>28</td>
<td>722</td>
</tr>
<tr>
<td>17</td>
<td>29</td>
<td>577</td>
</tr>
<tr>
<td>18</td>
<td>30</td>
<td>621</td>
</tr>
<tr>
<td>19</td>
<td>31</td>
<td>227</td>
</tr>
<tr>
<td>20</td>
<td>32</td>
<td>221</td>
</tr>
<tr>
<td>21</td>
<td>33</td>
<td>127</td>
</tr>
<tr>
<td>22</td>
<td>34</td>
<td>46</td>
</tr>
<tr>
<td>23</td>
<td>35</td>
<td>61</td>
</tr>
<tr>
<td>24</td>
<td>36</td>
<td>47</td>
</tr>
<tr>
<td>25</td>
<td>37</td>
<td>23</td>
</tr>
<tr>
<td>26</td>
<td>38</td>
<td>10</td>
</tr>
<tr>
<td>27</td>
<td>39</td>
<td>60</td>
</tr>
<tr>
<td>28</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>29</td>
<td>41</td>
<td>16</td>
</tr>
<tr>
<td>30</td>
<td>42</td>
<td>15</td>
</tr>
<tr>
<td>31</td>
<td>43</td>
<td>11</td>
</tr>
<tr>
<td>32</td>
<td>44</td>
<td>13</td>
</tr>
<tr>
<td>33</td>
<td>45</td>
<td>7</td>
</tr>
<tr>
<td>34</td>
<td>46</td>
<td>3</td>
</tr>
<tr>
<td>35</td>
<td>47</td>
<td>7</td>
</tr>
<tr>
<td>36</td>
<td>48</td>
<td>4</td>
</tr>
<tr>
<td>37</td>
<td>49</td>
<td>4</td>
</tr>
<tr>
<td>38</td>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>39</td>
<td>51</td>
<td>1</td>
</tr>
<tr>
<td>40</td>
<td>52</td>
<td>1</td>
</tr>
<tr>
<td>41</td>
<td>53</td>
<td>0</td>
</tr>
<tr>
<td>42</td>
<td>54</td>
<td>0</td>
</tr>
<tr>
<td>43</td>
<td>55</td>
<td>0</td>
</tr>
<tr>
<td>44</td>
<td>56</td>
<td>0</td>
</tr>
<tr>
<td>45</td>
<td>57</td>
<td>1</td>
</tr>
<tr>
<td>46</td>
<td>58</td>
<td>2</td>
</tr>
<tr>
<td>47</td>
<td>59</td>
<td>1</td>
</tr>
<tr>
<td>48</td>
<td>60</td>
<td>0</td>
</tr>
<tr>
<td>49</td>
<td>61</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>62</td>
<td>0</td>
</tr>
<tr>
<td>51</td>
<td>63</td>
<td>0</td>
</tr>
<tr>
<td>52</td>
<td>64</td>
<td>0</td>
</tr>
<tr>
<td>53</td>
<td>65</td>
<td>1</td>
</tr>
<tr>
<td>54</td>
<td>66</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 6.3. Histogram of Output Image.
Figure 6.5. Original and reconstructed images of the City of Mobile.
REFERENCES:

Chapter 1


Chapter 2


Chapter 3


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.

REFERENCES (continued)


Chapter 4


Chapter 5


Chapter 6


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

\[
\begin{bmatrix}
X \\
Y
\end{bmatrix} = A \begin{bmatrix}
XP \\
YP
\end{bmatrix} + \begin{bmatrix}
XO \\
YO
\end{bmatrix} + \begin{bmatrix}
ALFA \\
BETA
\end{bmatrix} \times XP \times YP
\]

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

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.

<table>
<thead>
<tr>
<th>Calling Program</th>
<th>Programs Called</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEOCORS</td>
<td>SARN</td>
</tr>
<tr>
<td></td>
<td>DAWN</td>
</tr>
<tr>
<td></td>
<td>GEOMM</td>
</tr>
<tr>
<td>GEOMM</td>
<td>GEOM1</td>
</tr>
<tr>
<td></td>
<td>DARN</td>
</tr>
<tr>
<td></td>
<td>VNATS</td>
</tr>
<tr>
<td></td>
<td>GEOM2</td>
</tr>
<tr>
<td></td>
<td>GEOM3</td>
</tr>
<tr>
<td></td>
<td>SAWN</td>
</tr>
<tr>
<td>GEOM1</td>
<td>VMOV</td>
</tr>
<tr>
<td></td>
<td>GEOM1B</td>
</tr>
<tr>
<td></td>
<td>XCHNGE</td>
</tr>
<tr>
<td>GEOM3</td>
<td>GEOM4</td>
</tr>
<tr>
<td>GEOM1B</td>
<td>GEOM1A</td>
</tr>
<tr>
<td>GEOM4</td>
<td>DARN</td>
</tr>
</tbody>
</table>

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 1200x1200 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 ≤ 1500; NEL ≤ 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.
2.12 Tests

The program has been checked out using a test pattern, applying a $45^\circ$ rotation to it and printing the results. Also, a transformation with a small nonlinear term added to the $45^\circ$ 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=0, LIMEXT=56, SIZE=DOCK

COMMON/GEOM/XO,YO,ALFA,BETA,DET1,DET2,LO,HI,JHI,JHII,JHIII,NREAD

DIMENSION A(2,2),B(2,2)
DIMENSION IX(4096),LX(11)

EQUIVALENCE(IX(4096),LX(11))
LOGICAL IX,LX(1500)

DATA MAXC/48000/
DEFINE FILE 9011500,1500,1,AMAL

READ_INPUT IMAGE SIZE.

READ 100,NREC,NEL

COPY INPUT IMAGE TO DISK.

DO 10 I=1,NREC
CALL SARNIC,IX,NEL4).
DO 20 J=1,NEL
CALL DAWN(90,1,IX,NEL)
READ TRANSFORMATION PARAMETERS.
THE EIGHT PARAMETERS, XO, YO, ALFA, BETA, ARE THOSE FOUND BY GEODUCE.
SX AND SY ARE SCALE FACTORS IN THE E AND N DIRECTIONS IN PIXELS/KM.

READ 200,A,ALFA,BETA,SY

MODIFY THE TRANSFORMATION ACCOUNTING FOR SCALE FACTORS.

A(1,1)=A(1,1)/SX
A(2,1)=A(2,1)/SX
A(1,2)=A(1,2)/SY
A(2,2)=A(2,2)/SY
ALFA=ALFA/SX/SY
BETA=BETA/SX/SY

APPLY THE TRANSFORMATION.

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

STOP

FORMAT(2I6)
FORMAT(2I3)
END
SUBROUTINE GEOMM(INAXCNRECNEL,NTAPUI)

PURPOSE: TO APPLY GEOMETRIC TRANSFORMATION TO A LARGE IMAGE STORED ON A DIRECT ACCESS DEVICE SET UNIT 9C. THE OUTPUT WILL APPEAR ON SEQUENTIAL ACCESS DEVICE NTAPD.

ONLY ONE ARRAY, IX OF MAXC WORDS, IS SUPPLIED AS A WORK AREA AND THE ALLOCATIONS FOR VARIOUS SUBARRAYS ARE HANDLED INTERNALLY.

INPUTS: A, XU, YO, ALFA, BETA ARE EIGHT PARAMETERS DEFINING A TRANSFORMATION OF THE FORM:

\[(X, Y) \mapsto (X', Y') = (X, Y) + (ALFA, BETA)(X, Y)\]

WHERE THE ' DENOTES TRANSPOSITION, X, Y ARE THE COORDINATES IN THE INPUT IMAGE, XP, YP ARE THOSE IN THE OUTPUT IMAGE. THE IMAGE ON UNIT 9C SHOULD BE IN BYTES (1 BYTE/PIXEL).


DEFINE FILE INREC, NEL, IAV

FIRST, COMPUTE THE OUTPUT IMAGE SIZE AND MODIFY THE SUPPLIED TRANSFORMATION FOR CONVENIENCE OF IMPLEMENTATION.

CALL GEOMM(INREC, NEL, NRECD, NELO)

WRITE(6,100)INREC, NEL, NRECD, NELO

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

MAXCP=MAXC-NELO+4

NB=MAXCP+4/(NEL+4)

WRITE(6,200)MAXC, MAXCP, NB

FIND STARTING ADDRESSES FOR WORK AREAS.

X COORDINATES

IAD1=MAXCP+1

Y COORDINATES

IAD2=IAD1+NELO
C CODES INDICATING WHICH PARTS OF AN OUTPUT RECORD ARE TO BE COMPUTED
ISN 0013. IAD3=IAD2+NELO
ISN 0014. IAD4=IAD3+NELO
ISN 0015. BUFFER POINTERS
IAD5=IAD1=NR
ISN 0016. INITIALIZE CIRCULAR BUFFER AND BUFFER POINTER ARRAYS.
ISN 0017. IRI=1
ISN 0018. IRE=1
ISN 0019. CALL DARN(9C1,IX,NEL)
ISN 0019. NREAD=1
ISN 0020. CALL VNATS(IX(IAD5),NR)
ISN 0021. COMPUTE OUTPUT RECORDS.
NELO=NELO+4
ISN 0022. DO I0=1,NRECO
C FIND X, Y COORDINATES IN THE INPUT IMAGE CORRESPONDING TO EACH OF
C THE OUTPUT POINTS IN THE I' TH RECORD.
ISN 0023. CALL GEOM2(IX(IAD1),IX(IAD2),IX(IAD3),IX(IAD4),IX(IAD5),NR,NREC)
C COMPUTE I' TH OUTPUT RECORD.
ISN 0024. CALL GEOM3(IX,IX(IAD1),IX(IAD2),IX(IAD3),IX(IAD4),IX(IAD5),NR,NREC)
C WRITE TO OUTPUT RECORD.
ISN 0025. WRITE(I6,3RR)NREAD
ISN 0026. END
SUBROUTINE GEOM1(NRECNEL,NECELL)

COMMON/GECM/A,XO,YO,ALFA,BETA,DET1,DET2,IL0,IL1,JH0,JHI,B,M0,EAD

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 PRINT THE INPUT PARAMETERS.

WRITE(6,100)
WRITE(6,200)((A(I,J),J=1,2),I=1,2)
WRITE(6,300),XO,YO,ALFA,BETA
CALL_VH(DA(4,4),B)
DET1=A(1,1)+A(2,2)-A(1,2)*A(2,1)

WRITE(6,310),(A(I,J),J=1,2),I=1,2)
WRITE(6,310),ALFA,BETA
ISN 0014 BETA=-BETA

CALL_XCHNGE(A(1,1),A(1,2))
CALL_XCHNGE(A(2,1),A(2,2))

FIND MAXIMUM NUMBER OF INPUT RECORDS NEEDED TO PRODUCE A RECORD OF
OUTPUT. IF THE GIVEN TRANSFORMATION IS USED, THE ROLES OF
XP AND YP ARE INTERCHANGED.

R1=ABS(A(1,2)+ALFA*IL0)*(JH0-JH1)
R2=ABS((A(1,1)+ALFA*PHI)*(JHI-JL0))
R2=MAX(R1,R2)

CALL_XCHNGE(A(1,1),A(2,1))
CALL_XCHNGE(A(2,1),A(2,2))
CALL_XCHNGE(JH0,JH1)

MODIFY THE TRANSFORMATION TO INTERCHANGE THE ROLES OF XP AND YP.

CONTINUE

MODIFY THE TRANSFORMATION, IF NECESSARY, TO ENSURE THAT X INCREASES
WITH XP.

IF(A(1,1).GE.0.16D0) GO TO 20
A(1,1)=A(1,1)
A(2,1)=A(2,1)
BETA=BETA
CALL_XCHNGE(IL0,IL1)
H0=H0
CONTINUE

MODIFY THE TRANSFORMATION, IF NECESSARY, TO ENSURE THAT Y INCREASES
WITH YP.

IF(A(2,1).GE.0.16D0) GO TO 30
A(2,1)=A(1,1)
A(2,2)=A(2,2)

CONTINUE
ALPHA = ALFA,
BETA = BETA,

CALL XCHNGE(JLO, JHI)
JLO = JLO,
JHI = JHI

CONTINUE

NRECO = JHI - JLO + 1
NELO = JHI - JLO + 1

C
PRINT MODIFIED TRANSFORMATION

WRITE(6,400)
WRITE(6,200)(A(I,J), J=1,2), I=1,2
WRITE(6,300)(X0, Y0, ALFA, BETA)
WRITE(6,500)(JLO, JHI, JHI)
RETURN

FORMAT(///' GIVEN TRANSFORMATION PARAMETERS:
FORMAT(///' MATRIX/(1X2E15.7)
FORMAT(///' SHIFT VECTOR: 2E15.7/
FORMAT(///' COEFFICIENTS OF PRODUCT TERM: 2E15.7)
FORMAT(///' MODIFIED TRANSFORMATION PARAMETERS:
FORMAT(///' TOP LEFT CORNER=16, 16, 16, 16, 16, 16)
BOTTOM RIGHT CORNER=1, 1, 1, 1, 1, 1)
END
C FIND X2, YP GIVEN X, Y.

ISN 0005 XXO=X-XO

ISN 0006 YYYY=YY

ISN 0007 BB=DE+ALPHA*YY-BETA*XX

ISN 0008 CC=YYA+1.2-XXA2+1.2

ISN 0009 IF(ABS(DE+),GT,1.E-10)*XP=-BB+SIGN*SQRT(BB-BB-4*DE+CC)/

(2*DE2)

ISN 0011 IF(ABS(DE2)+,LE,1.E-9)*XP=-CC/BB

ISN 0013 *YP=(YYA+1.2,1.1*XP)/(1.1,2,BETA*XP)

ISN 0014 RETURN

ISN 0015 END
C FIND MIN AND MAX VALUES OF XP, YP FOR X, Y RANGING FROM 1 TO NREC
C AND 1 TO NEL RESPECTIVELY.

ISN 0029 CALL_GEOMIA1(1,1,XP,YP,1)
ISN 0029 CALL_GEOMIA1(1,1,XP,YP,1)
ISN 0029 CALL_GEOMIA1(1,1,XP,YP,1)
ISN 0029 CALL_GEOMIA1(1,1,XP,YP,1)
ISN 0029 CALL_GEOMIA1(1,1,XP,YP,1)

IF (ABS(XPM) .GE. ABS(XP)) GO TO 10

ISN 0006 XP = XPM
ISN 0006 YP = YPM

ISN 0017 XMIN = XP
ISN 0017 XMAX = XP

ISN 0018 YMIN = YP
ISN 0018 YMAX = YP

ISN 0019 NREC = MAX(NREC - 1, 1)
ISN 0019 NEL = MAX(NEL-1, 1)

ISN 0020 DO 20 K = 1, 2
ISN 0021 TEL = NEL
ISN 0022 IF (K .EQ. 1) GO TO 30
ISN 0024 IREC = NREC
ISN 0025 IEL = 1

ISN 0026 CONTINUE
ISN 0027 DO 20 J = 1, NREC, IREC
ISN 0028 DO 20 J = 1, NEL, IEL
ISN 0029 K = 1
ISN 0030 Y = 1

ISN 0031 CALL_GEOMIA1(X, Y, XP, YP, 1, ISIGN)
ISN 0032 XMAX = AMAX1(XMAX, XP)
ISN 0033 YMAX = AMAX1(YMAX, YP)

ISN 0036 CONTINUE
ISN 0037 CONTINUE
ISN 0038 CONTINUE
ISN 0039 CONTINUE
ISN 0040 RETURN

ISN 0042 END
SUBROUTINE CEOM2(XX,XY,NELEN,RECNEL)

COMMON/GECH/A,XO,YO,ALFA,BETA,DET1,DET2,ILO,IHI,JLO,JHT,NREAD

DIMENSION A(2,2),B(1,2)

C FIND_ARRAYS.DE X AND Y COORDINATES FOR THE I TH OUTPUT RECORD.

DIMENSION X(NELO),Y(NELO)

I=1:JLO=1

DO 10 J=1,NELO

JJ=J+JLO-1

IF(J.GT.JLO).OR.J(J)GT.NELCJ RETURN

X(J)=A(I,1)+X(I)+ALFA(J)+JJ

Y(J)=C.

IF(J.GT.JLO).OR.X(J)GT.NREC GO TO 10

Y(J)=AL2,1+AL2,1+JJ+Y(J)+BETA(J)+JJ

10 CONTINUE

RETURN

END
COMPUTE ONE RECORD OF RESAMPLED OUTPUT BY READING THE NECESSARY INPUT RECORDS FROM THE INPUT ACCESS DEVICE (UNIT 9).

C THIS PROGRAM IS DESIGNED TO HANDLE 
C THE COMPUTATIONS OF JAI, JIF SHOULD BE CHANGED IN THE CASE OF 
C INTERPOLATION FOR RESAMPLING.

C IN ORDER TO READ FORWARD AND COMPUTE PART OF THE OUTPUT RECORD. IF JRI is less than JRF read forward and compute part of the output record.
C IF JRI is greater than JRF read backward and compute part of the output record.
C IF JRI is less than or equal to JRF AND JRI is greater than or equal to JRF read forward and compute part of the output record.
C IF JRI is less than or equal to JRF AND JRI is less than JRF read backward and compute part of the output record.

C COMPILE ONE RECORD OF RESAMPLED OUTPUT BY READING THE NECESSARY INPUT RECORDS FROM THE INPUT ACCESS DEVICE (UNIT 9).
IF IPASS.EQ.1 THEN IT IS THE FIRST CALL TO GEOH4 CORRESPONDING TO
THE PRESENT RECORD. THEREFORE, COMPUTE AS MUCH AS POSSIBLE WITH-
OUT READING ANY NEW RECORDS.

IF (IPASS.EQ.1) THEN

CALL GEOH4 (IDIR, NLFFT, IRT, IRF, NR, JRI, JRF, IN, NEL, IC, NELO, IN, X, Y)

IF (NLFFT.GT.0) GO TO 3C

RETURN

END
LEVEL 21.9 (JUN.74.)  O5/36 FORTRAN H  DATE 76.22/20,5.29

COMPILER OPTIONS -- NAME = MAIN, OPT = 02, LINECNT = 56, SIZE = 602.

ISN_0003. LOGICAL IN
ISN_0004. DIMENSION NLSEL,IC(N1),IREM(N1,XN),(N1),(N1)
ISN_0005. DIMENSION XL2(21,2L2,2)
ISN_0006. COMMON/GEOM,XO,YO,ALPHA,BETA,ETA1,ETA2,ILR,ILG,JHI,JLH,JBI,B,NREAD
C COMPARE A PART 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.
C
ISN_0007. NR2=NR2+1
ISN_0008. IRI=IRI + I
ISN_0009. IF(IRI>IRI+20,19)
ISN_0110. IRF=MAX(IRF+NR-1,IRF)
ISN_0111. IRI=IRI+I
ISN_0112. IB2=IRF+1
ISN_0114. GO TO 40
C IDIR = CT. FIND IRI, IRF TO READ BACKWARD.
ISN_0115. IF(IRF-IDIR<NR-1,IRF)
ISN_0116. IRI=MIN(IRI+NR-1,IRF)
ISN_0117. IRF=MAX(IRF-IDIR,IRF-1)
ISN_0118. IRI=IRF+1
ISN_0119. IB2=IRF
C MODIFY ICB SUCH THAT ICB(I) GIVES ADDRESS OF I-TH INPUT RECORD.
ISN_0120. INC=IIR1-IIR2+NR21
ISN_0121. DO 50 I=1,IR2
ISN_0122. ICBI(I)=INC, I, INC, NR)+1
C READ THE NECESSARY INPUT RECORDS.
ISN_0123. IF(IRI+NR+IRF-IDIR,IRF)
ISN_0124. NREAD=NREAD+1
ISN_0125. GO TO 70
ISN_0126. CALL DARP(ON,1,IN(I),ICBI(I)+INC, ICBI(I)+INC, IR)
C COMPUTE PARTS OF THE OUTPUT ARRAY THAT CAN BE COMPUTED.
ISN_0127. NLEFT=0
ISN_0128. DO 50 J=3,NFL
ISN_0129. IF(IREM(N1,J),NE.111,GO TO 70
ISN_0130. IF(XI1>1.5
ISN_0131. IF(AIR>1,IRF,IRF+70)
ISN_0132. IR=ICB(I)-IR1+1
ISN_0133. IC=I+1
ISN_0134. ICB(I+1)=ICBI(I)
ISN_0135. GOTO 70
ISN_0136. IREMC(I)+1
ISN_0137. GO TO 70
ISN_0138. NLEFT=NLEFT+1

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR
SUBROUTINE VNATS(IX,N)

DIMENSION IX(N)

DO 10 I=1,N

IX(I)=1

10 RETURN

END
SUBROUTINE XCHANGE(X,Y)

W = X

Y = W

RETURN

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 ≥ IT (< 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 NTAP0 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 4000 x 2100 level H 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.
<table>
<thead>
<tr>
<th>Calling Program</th>
<th>Program Called</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEELS</td>
<td>PET</td>
</tr>
<tr>
<td></td>
<td>SARN*</td>
</tr>
<tr>
<td></td>
<td>VLTHR</td>
</tr>
<tr>
<td></td>
<td>CMPRES</td>
</tr>
<tr>
<td></td>
<td>DAWN*</td>
</tr>
<tr>
<td></td>
<td>PEELER</td>
</tr>
<tr>
<td></td>
<td>DARN</td>
</tr>
<tr>
<td></td>
<td>EXPBDY</td>
</tr>
<tr>
<td>CMPRES</td>
<td>ISTORE*</td>
</tr>
<tr>
<td>PEELER</td>
<td>SVSCI</td>
</tr>
<tr>
<td></td>
<td>PEELR1</td>
</tr>
<tr>
<td></td>
<td>PEELRO</td>
</tr>
<tr>
<td></td>
<td>DAWN*</td>
</tr>
<tr>
<td>EXPBDY</td>
<td>ILOAD*</td>
</tr>
<tr>
<td>PEELR1</td>
<td>DARN</td>
</tr>
<tr>
<td></td>
<td>BLSFTV</td>
</tr>
<tr>
<td></td>
<td>BRSFTV*</td>
</tr>
<tr>
<td>PEELRO</td>
<td>ILOAD+</td>
</tr>
<tr>
<td></td>
<td>ISTORE+</td>
</tr>
<tr>
<td>BLSFTV</td>
<td>ILOAD+</td>
</tr>
<tr>
<td></td>
<td>ISTORE+</td>
</tr>
<tr>
<td>BRSFTV</td>
<td>ILOAD+</td>
</tr>
<tr>
<td></td>
<td>ISTORE+</td>
</tr>
</tbody>
</table>

* 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:

\[
\text{IF (IT.GE.0) LY(I) = LX(I).GE.IT}
\]
\[
\text{IT (IT.LT.0) LY(I) = LX(I).LE.IABS(IT)}
\]

for \( I = 1, \text{NEL} \).

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

\[
\begin{array}{ccc}
\text{a} & \text{b} & \text{c} \\
\text{d} & \text{e} & \text{f} \\
\text{g} & \text{h} & \text{i}
\end{array}
\]
where each letter represents a binary pixel. It is to be decided whether e, which is presently equal to 1 should be changed to 0. The conditions for a 'top peel' will be derived below and those for peeling from the other directions 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 \( bh = \). (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 \( ad = 1 \) or \( cf = 1 \). Therefore, the condition for a top peel is that

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

Equivalently, to perform a top peel we set

\[
e = e (b+h+ad+cf).
\]

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, \( \text{NP} \), of words of input that were changed is counted during each call to PEELER. If \( \text{NP} = 0 \) or the number of calls to PEELER has been MPASS, the iterations are stopped.

3.9.3 Converting to SIC

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.
SUBROUTINE PEELS(NTI,NTD,HREC,NEL,IT,HPASS,MDEV,NDEV,LX,LY,IBDY)

C DIMENSION LX(36*(INEL-1)/32+1),LY(INEL-I/4+i)

C DEFINE FILE MDEV(NREC,(NEL-1)/32+1),U,IAV1

C DEFINE FILE NDEV(NREC,(NEL-1)/32+1),U,IAV2)

N=(NEL-1)/32+1

CALL PET(2)

DO 10 I=1,NREC

CALL SARN(NTI,LX,NEL)

CALL VTTHR(LX,NEL)

CALL DAVN(MDEV,I,LXN*4)

CALL PET(2)

DO 20 IPASS=1,MPASS

IF(IPASS.EQ.1) CALL PEELER(HDEV,NDEV,NREC,N,LX,LX(12*N+1),

LY(24*N+1),LY,NP)

IF(IPASS.EQ.0) CALL PEELER(NDEV,MDEV,NREC,N,LX,LX(12*N+1),

LY(24*N+1),LY,NP)

PRINT IO,IPASS,NP

CALL PET(2)

IF(NP.EQ.0) GO TO 30

CONTINUE

IPASS=MPASS

IF(IPASS.EQ.0) GO TO 30

JDEV=HDEV

DO 30 I=1,NREC

CALL DAVN(JDEV,I,LX,NP)

CALL EXPBDY(LX,N,NEL,IBDY,J)

WRITE(NTDJ,I,BDY(L)*L=1,J)

CALL PET(2)

RETURN

FORMAT(5X,'DURING PASS NUMBER*13,* THROUGH PEELER*16,* WORDS OF CON. 
-PRESSED BOUNDARY INFORMATION WERE CHANGED.*')

END


SUBROUTINE BLSFTV(IX,N,IY)

DO 10 I=1,N

IY(I)=ILOAD(IX(I+1),32,1)

10 IY(I)=ISTORE(IX(I),IY(I),32,31)

RETURN

END

SUBROUTINE BRSFTV(IX,N,IY)

DO 20 I=2,N

IY(I)=ILOAD(IX(I),32,31)

20 IY(I)=ISTORE(IX(I-1),IY(I),32,1)

RETURN

END
SUBROUTINE BPKINT (LX, NELY, LYL, LI)

DIMENSION LX(N)

LOGICAL LOAD

LOGICAL*1 LY(NEL), LO, LJ

JWRD = 1

JBIT = 33

DO 10 I = 1, NEL

LY(I) = 0

JBIT = JBIT - 1

IF (JBIT .NE. 0) GO TO 20

JBIT = 32

JWRD = JWRD + 1

IF (LOAD(LX(JWRD), JBIT, 1), LY(I) = 1

CONTINUE

PRINT 100, LY

100 FORMAT (1X, 4(Saza, 1X))

RETURN

END
COMPILER OPTIONS - NAME = MAIN, OPT = 02, LINES = 56, SIZE = 0000K,
SOURCE, EBCDIC, NOLIST, NODECK, LOAD, MAP, NODEIT, SD, NJXREF

SUBROUTINE CMRES(LX,NEL,LX)

LOGICAL*1 LX(NEL)
DIMENSION LY(I)
JWRO = 1
DO 10 I = 1, NEL
IF (JBIT.NE.0) GO TO 20
10 CONTINUE
JBIT = JBIT - 1
JWRO = JWRO + 1
I = LX(I)
LY(JWRO) = ISTORE(I, LX(JWRO), JBIT, 1)
10 CONTINUE
RETURN
END
SUBROUTINE EXPBDY(LX, NEL, IBDY, J)

DIMENSION LX(NEL), IBDY(NEL)

LOGICAL ILOAD

JWRO=1

DO 10 I=1, NEL

JBIT=JBIT-1

IF(JBIT.NE.0) GO TO 20

JBIT=32

10 CONTINUE

RETURN

END
SUBROUTINE PEE1R(PDEV, NDEV, NREC, N, LX, LXR, LXL, LY, NP)

NREC1 = NREC + 1
NREC2 = NREC + 2
NREC3 = NREC + 3
NREC4 = NREC + 4

J(1) = 1
J(2) = 2
J(3) = 3

CALL SVSCI(LX(N, 3, 4), LY(N), J(3))

NREC = NREC + 1
DO 10 I = 1, NREC

CALL SVSCI(LXR(N, 3, 4), J(3), N, 0)

CALL SVSCI(LXL(N, 3, 4), J(3), N, 0)

NP = 0

IF I LE NREC GO TO 20

CALL SVSCI(LX(I, J(3), K), N, 0)

CALL SVSCI(LXR(I, J(3), K), N, 0)

CALL SVSCI(LXL(I, J(3), K), N, 0)

CONTINUE

IF I LT NREC CALL PEE1R(PDEV, I, LY, J, N, LXR, LXL)

CALL PEE1R(LX(I, 1), 1, LX, I, 1, 1, LXL(I, 1, 1), J, N, 1,

J(3), 2), LXL(I, J(3), 2), NP)

CALL PEE1R(LX(I, 2), 1, LX, I, 2, 2, LXL(I, 2, 2), J, N, 2,

J(3), 3), LXL(I, J(3), 3), NP)

CALL PEE1R(LX(I, 3), 1, LX, I, 3, 3, LXL(I, 3, 3), J, N, 3,

J(3), 4), LXL(I, J(3), 4), NP)

IF I LT NREC CALL PEE1R(LX(I, 4), 1, LX, I, 4, 4, LXL(I, 4, 4), J, N, 4,

LY(0, 0, NP)

IF I LT NREC CALL PEE1R(PDEV, I, LY, 4, N)

DO 40 K = 1, 4

J(K) MOD 4, J(K), 31 + 1

CONTINUE

RETURN
END
SUBROUTINE PEELR1(INDEV, LX, N, LXR, LXL)

CALL DARN(INDEV, LX(1, J(I)), N(4))
CALL BLSFTV(LX(1, J(I)), N(4), LXL(I, J(I)))
CALL BLSFTV(LX(1, J(I)), N(4), LXR(I, J(I)))

RETURN
END
SUBROUTINE PEELRO(LX, LXR, LYL, LYN, ISIDE, LLY, LLYN, NP)

      DIMENSION LX(N,3), LXR(N,3), LYL(N,3), LYN(N,3), LLY(N), LLYN(N)

      DO 60 I = 1, N
      LYL(1) = LYL(I, J(2))
      IF(LY(I).EQ.0) GO TO 60
      GO TO (10, 20, 30, 40), ISIDE

      10 IW(1) = IOR(LX(I,J(1)), ICMP(LX(I,J(1)), 32, 32))
      IW(2) = IAND(LXR(I,J(1)), ICMP(LXR(I,J(1)), 32, 32))
      IW(3) = IAND(LXL(I,J(1)), ICMP(LXL(I,J(1)), 32, 32))
      GO TO 50

      20 IW(1) = IOR(LXR(I,J(2)), ICMP(LXR(I,J(2)), 32, 32))
      IW(2) = IAND(LXR(I,J(2)), ICMP(LXR(I,J(2)), 32, 32))
      IW(3) = IAND(LXR(I,J(3)), ICMP(LXR(I,J(3)), 32, 32))
      IW(2) = IAND(LXR(I,J(2)), ICMP(LXR(I,J(2)), 32, 32))
      IW(3) = IAND(LXR(I,J(3)), ICMP(LXR(I,J(3)), 32, 32))
      GO TO 60

      30 IW(1) = IOR(LX(I,J(3)), ICMP(LX(I,J(3)), 32, 32))
      IW(2) = IAND(LXR(I,J(3)), ICMP(LXR(I,J(3)), 32, 32))
      IW(3) = IAND(LXR(I,J(3)), ICMP(LXR(I,J(3)), 32, 32))
      GO TO 60

      40 IW(1) = IOR(LX(I,J(2)), ICMP(LX(I,J(2)), 32, 32))
      IW(2) = IAND(LXR(I,J(2)), ICMP(LXR(I,J(2)), 32, 32))
      IW(3) = IAND(LXR(I,J(3)), ICMP(LXR(I,J(3)), 32, 32))
      IW(2) = IAND(LXR(I,J(2)), ICMP(LXR(I,J(2)), 32, 32))
      IW(3) = IAND(LXR(I,J(3)), ICMP(LXR(I,J(3)), 32, 32))
      GO TO 60

      50 IW(1) = IOR(LX(I,J(1)), LLY(I))
      IW(2) = IOR(LX(I,J(1)), LLY(I))
      IW(3) = IAND(LXL(I,J(1)), LLY(I))
      IF(LY(I).NE.LYL(I)) NP = NP + 1
      CONTINUE
      IF(ISIDE.EQ.4) RETURN
      CALL BSFTLY(LY, N, LYL)
      CALL BSFTLY(LN, N, LLY)
      RETURN
      END
SUBROUTINE VLTHR(LX,N,LY)

LOGICAL*I LX(N),LY(N)F/.FALSE./,T. TRUE./

C THRESHOLD A VECTOR LX OF 8 BIT INTEGERS TO GET A T-F VECTOR.
C IF IT.GE.0, LX(II).GE.IT IMPLIES LY(II)=T. IF IT<0 LX(III).LE.IABS(IT).
C IMPLIES LY(II)=T.

ITT=IABS(IT)

IF(IT.LT.0)GO TO 10

DO 20 I=1,N

20 IF(LX(I).GE.ITT)LY(I)=T

IF(LX(I).LE.IITT)LY(I)=F

RETURN

10 DO 30 I=1,N

30 IF(LX(I).LE.IITT)LY(I)=T

RETURN

END
4. 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.

<table>
<thead>
<tr>
<th>Calling Program</th>
<th>Program(s) Called</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOUDIM</td>
<td>BOUDIS</td>
</tr>
<tr>
<td>BOUDIS</td>
<td>JCOUNT</td>
</tr>
</tbody>
</table>

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 ith 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 ith 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(NDISK), 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.
SUBROUTINE BOUDIBDYNTAPI.NRTCNELIRCNDNOISI

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

DO 10 I=1,3
IND(I)=1
10 NB(I)=0

READ(NTAPl,NA2,IBDY(J,2),J=1,NA2)

DO 20 I1,NREC
IBDY(J,IND(NB3),NB3=0
20 IND(J)=MOD(IND(J)+1,NDI

RETURN

END
C
C DIMENSION IBDY(NEL,3),IND(INB(3)
C
C IBDY(I,J,N1),IND(I,J) 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. 0)ICOUNT=
              * ICOUNT+1
ISN 0014      IF(J+1.NB2.AND.IBDY(J+1,IND(2))=IBDY(J,IND(2)) .EQ. 0)ICOUNT=
              * ICOUNT+1
              * ICOUNT+1
              * ICOUNT+1
ISN 0016      IF(ICOUNT.GE.21)GO TO 10
ISN 0018      IF(NB1.NE.0)ICOUNT=
              * ICOUNT+1
              * ICOUNT+1
              * ICOUNT+1
ISN 0020      IF(ICOUNT.GE.21)GO TO 10
ISN 0022      IF(NB3.NE.0)ICOUNT=ICOUNT+
              * ICOUNT+1
              * ICOUNT+1
              * ICOUNT+1
ISN 0024      IF(ICOUNT.GE.21)GO TO 10
ISN 0026      NDIS=NDIS+1
ISN 0027      WRITE(I,100)NDIS(IR,IBDY(J,IND(2))),ICOUNT
100   FORMAT(' DISCONTINUITY NO.',5x,AT ('14.', '14.'), ICOUNT='12,1')
ISN 0030      IRCC(NDIS,1)=IR
ISN 0039      IC(IND(3),2)=IBDY(J,IND(2))
ISN 0030      IRCC(NDIS,3)=ICOUNT
ISN 0031      10 CONTINUE
ISN 0033      RETURN
ISN 0034      END
FUNCTION JCOUNT(I,J,J1,J2)
DIMENSION I(I)
C JCOUNT NO. OF VALUES OF JJ SUCH THAT J1.LE.JJ.LE.J2
C AND I(JJJ)-I(JJJ).LE.1
JCOUNT =
IF(J1.IF(J2).ETUAN
K =
IF(J1.IF(J2).ETUAN
K =
JCOUNT = K
RETURN
END
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, IW, 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:

<table>
<thead>
<tr>
<th>Calling Program</th>
<th>Programs Called</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMOB2</td>
<td>PATCH3</td>
</tr>
<tr>
<td>PATCH3</td>
<td>SVSCI</td>
</tr>
<tr>
<td></td>
<td>PATCH1</td>
</tr>
<tr>
<td></td>
<td>SORT</td>
</tr>
<tr>
<td></td>
<td>ELIRPT</td>
</tr>
<tr>
<td>PATCH1</td>
<td>CONTEL</td>
</tr>
<tr>
<td></td>
<td>PATCH4</td>
</tr>
<tr>
<td></td>
<td>PATCH2</td>
</tr>
<tr>
<td></td>
<td>PRTVEC</td>
</tr>
<tr>
<td>SORT</td>
<td>MVMRMR</td>
</tr>
<tr>
<td>ELIRPT</td>
<td>VMOV</td>
</tr>
</tbody>
</table>
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.
SN 0002 COMMON/PATCHAD/1,J1,J2,J3,np
SN 0003 DIMENSION IRC(105,2), IRC(1), IRC(1,1), IRC(1,2)
C D'N IBYI(MAX. EXPECTED NO. OF BOUNDARY POINTS IN A LINE AFTER SMOOTHING)
C
C D'N IVI(K21**2), IV2(K21**2) WHERE K21=20K+1.
SN 0004 COMMON/PATCHAD/1,J1,J2,J3,np
SN 0005 CALL PATCH3(IDIS, IBY, IRC(1,1), IRC(1,2), IRC(1,3), K, IW1, IW2, NRECNODE)
SN 0006 DO 20 I=IDIS,NDIS
SN 0010 IF(I2.NE.0)PRINT 100,1,IRC(I,1),IRC(I,2),IRC(I,3),K,1W1,1W2,NRECNODE
SN 0007 IF(I1.NE.0.AND.I2.EQ.0)PRINT 101,1,IRC(I,1),IRC(I,2),IRC(I,3)
SN 0011 IF(I1.EQ.0)PRINT 102,1,IRC(I,1),IRC(I,2)
SN 0013 CONTINUE
SN 0014 RETURN
SN 0015 FORMAT1X5,**(*15,*15,') JOINED TO (*15,*15,') AND (*15,*15,')
SN 0016 * 15,'*')
SN 0017 FORMAT1X5,**(*15,*15,') JOINED TO (*15,*15,')
SN 0018 FORMAT1X5,**NO PATCH POINTS PRODUCED AT (*15,*15,')
SN 0019 END
COMPILER OPTIONS - NAME = MAIN, OPT = 02, LINECNT = 56, SIZE = 00000K,

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

ISN 0002 SUBROUTINE CONELIM(IW1, IW2, N)

ISN 0003 C DIMENSION IW1(M,N), IW2(2,1)

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

C

ISN 0004 K=1

ISN 0005 L=1

ISN 0006 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 JJ=MAX0(I-1,I,1)

ISN 0011 J2=MAX0(J-1,J,1)

ISN 0012 J2=MIN0(J+1,J,N)

ISN 0013 JJ=MIN0(J+1,J,1)

ISN 0014 DO 10 I=1,11,12

ISN 0015 DN 10 JJ=J1,J2

ISN 0016 IF(IW1(I,J).EQ.0)GO TO 10

ISN 0017 IW2(1,1)=I

ISN 0018 IW2(1,L)=J

ISN 0019 IW2(2,L)=JJ

ISN 0020 IW2(2,1)=J1

ISN 0021 IW1(I,J)=0

ISN 0022 10 CONTINUE

ISN 0023 20 CONTINUE

ISN 0024 IF(L.EQ.K)RETURN

ISN 0025 K=K+1

ISN 0026 LL=L+1

ISN 0027 DO 30 KK=K1,L

ISN 0028 DO 30 LK=K1,L

ISN 0029 IW2(1,LK)=IW2(1,KK)

ISN 0030 IW2(2,LK)=IW2(2,KK)

ISN 0031 30 IF(IW2(2,LK).EQ.0)GO TO 40

ISN 0032 K=LL

ISN 0033 L=LL

ISN 0034 GO TO 40

ISN 0035 END

REPRODUCIBILITY OF TH.
ORIGINAL PAGE IS POOR
LEVEL 21.9  (JUN 74)  US/360 FORTRAN H
DATE 76.159/10.56.40

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

ISN_0002  SUBROUTINE PATCH(I,I1,I2,H,M,N,J,IJ,J1,J2,J)
ISN_0003  COMMON/PATCH2/I1,I2,H,M,N,J,IJ,J1,J2,J
ISN_0004  DATA IPASS=0PASS+1
ISN_0005  IF(IPASS.LE.40)CALL PRTVEC(I1,H,M,N,J)
C  GENERATE PATCH POINTS IN I,J STARTING FROM (I,J) TO THE NEAREST
C  (2-ICOUNT) NONCONTIGOUS NEIGHBORS.
C  SEE CONTEL FOR DIMENSIONING INFO FOR IW2.
ISN_0008  DIMENSION IW1(H,N),IW2(2,1)
ISN_0009  IW2(1,1)=I
ISN_0110  IW2(2,1)=J
C ELIMINATE POINTS CONTIGUOUS WITH (I,J).
ISN_0011  CALL CONTEL(IW1,IW2,H,M,N)
C FIND NEAREST NEIGHBOR OF (I,J) WHICH IS SET.
ISN_0012  I2=0
ISN_0013  CALL PATCH2(IW1,I2,H,M,N,IJ,J1,J)
C NOW (I1,J1) IS THE NEAREST NEIGHBOR.
ISN_0014  IF(IJ,J1).NE.0.OR.I1.EQ.0.GO TO 10
C ELIMINATE POINTS CONTIGUOUS WITH (I1,J1).
ISN_0016  IW2(I1,1)=I1
ISN_0017  IW2(I2,1)=J1
ISN_0018  CALL CONTEL(I1,I2,IW2,H,M,N)
ISN_0019  CALL PATCH2(I1,I2,H,M,N,I1,J1,I2,J2,J)
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,J1).EQ.0.RETURN
C PRODUCE PATCH ADDRESSES IN IW2.
ISN_0025  CALL PATCH2(IW2,MP,I1,IJ,J1,J)
ISN_0026  IF(I1,J1).EQ.10.CALL PATCH2(IW2,MP+1,MP+2)
ISN_0027  10 MP=MP+1
ISN_0028  IF(IPASS.LE.40)CALL PRTVEC(IW2,NP+1)
ISN_0029  RETURN
ISN_0032  END
TO GENERATE COORDINATES OF LINE JOINING \((i_1, j_1)\) AND \((i_2, j_2)\).

\[
\begin{align*}
\text{IM} & = \text{MIN}(i_1, i_2) + 1 \\
\text{JMN} & = \text{MIN}(j_1, j_2) + 1 \\
\text{JMX} & = \text{MAX}(j_1, j_2) - 1 \\
\text{I12} & = i_1 - i_2 \\
\text{J12} & = j_1 - j_2 \\
\text{RI12} & = i_1 + j_1 \\
\text{RIJ12} & = i_1 + j_2 \\
\text{NP} & = 0 \\
\text{IF}(\text{IM} > \text{JMN}, \text{JMX} - \text{MIN} + 10) & = 10 \quad \text{IF}(\text{JMN} > \text{JMX}) \quad \text{return} \\
\text{DP} & = 0 \quad \text{return} \\
\text{I} & = (i_1 + (j_1 - 1)) \times 12 / (j_1 + 5) \\
\text{NE} & = \text{NP} + 1 \\
\text{IW1} & = \text{NP} + 1 \\
\text{IW2} & = \text{NP} + 1 \\
\text{IW3} & = \text{NP} + 1 \\
\text{return} \\
\text{IQ} & = \text{NP} + 1 \\
\text{J} & = (i_1 + 11) \times 12 / (j_1 + 5) \\
\text{IW1} & = \text{NP} + 1 \\
\text{IW2} & = \text{NP} + 1 \\
\text{IW3} & = \text{NP} + 1 \\
\text{return} \\
\text{END}
\end{align*}
\]
**LEVEL 21.8 (JUN 74)**

** compiler options - NAME = main,OPT=02,LINENT=50,SIZE=0000K, SOURCE,EBCDIC,NMLST,NOID,WLL,CLOAD,PAED,NOEDIT,DN,NOXREF. **

** ISN 0002 SUBRURINE PATCH3 (IBDY,I,J,ICOUNT,K,1M1,1M2,NREC,NDEV).**

** ISN 0003 COMMON/PTCHAD/I1,11,2,1,J2,NP,**

** c DIMENSION IW1(K21+1,1M2(2,1),HP), WHERE NP=MAXNUM OF PATCH POINTS.**

** C EXPECTED TO BE GENERATED BY PATCH2, DIMENSION REQUIRED BY CONTENL.**

** ISN 0004 DIMENSION IW1(1,11,1M2(2,1),1BDY(1).**

** ISN 0005 K21=K2+1,**

** ISN 0006 K1=MAXO(I-K,1),**

** ISN 0007 K2=MINO(I+K,NREC).**

** ISN 0008 K21=K2-K1+1,**

** ISN 0009 CALL RVCSC(IW1,K21,K21,0).**

** ISN 0010 ICLMMK=J-K,**

** ISN 0011 ICLMMK=J+K,**

** ISN 0012 DO 10 KK=K1,K2,**

** ISN 0013 READ(NDEV,KBK11MW1(1BDY(I1,11=1,N).**

** ISN 0014 IF(N.EQ.O)GO TO 10,**

** ISN 0016 DO 20 L=1,N,**

** ISN 0017 IF(1BDY(L),L,ICLMMK)GO TO 20,**

** ISN 0019 IF(1BDY(L),L,ICLMMK)GO TO 10,**

** C TREAT IW1 AS A 2D ARRAY DIMENSIONED K21*K21 WITH THE GIVEN POINT.**

** C AT (I-K1+1,K+1)TH LOCATION.**

** ISN 0021 IW1(1BDY(L),ILMMSK*K21*KK-K1+1)=1,**

** ISN 0022 20 CONTINUE,**

** ISN 0023 10 CONTINUE,**

** C GENERATE PATCH ADDRESSES IN IW2.**

** C CALL PATCH1(IW1,IW2,K21,K21,I-K1+1,K+1,ICOUNT).**

** C MERGE ADDRESSES FOUND IN IW2 WITH THE BOUNDARY ADDRESSES ON DISC.**

** ISN 0024 CALL PATCH1(IW1,IW2,K21,K21,I-K1+1,K+1,ICOUNT).**

** C MERGE ADDRESSES FOUND IN IW2 WITH THE BOUNDARY ADDRESSES ON DISC.**

** ISN 0025 IF(NP.EQ.O)RETURN,**

** ISN 0027 IP=1,**

** ISN 0028 IP1=1,**

** ISN 0029 30 IP=IP1,**

** C FIND NEXT CHANGE IN IW2(IP).**

** C IF(IP<IP1 GT NP)GO TO 60,**

** ISN 0030 IF(IP+1,IP1,EQ.IP+1)GO TO 30,**

** ISN 0031 IF(IP+1,IP1,EQ.IP-1)GO TO 30,**

** ISN 0032 IF(IP+1,IP1,EQ.IP-1)GO TO 30,**

** ISN 0033 IF(IP+1,IP1,EQ.IP-1)GO TO 30,**

** ISN 0034 IF(IP+1,IP1,EQ.IP-1)GO TO 30,**

** ISN 0035 IF(IP+1,IP1,EQ.IP-1)GO TO 30,**

** ISN 0036 IF(IP+1,IP1,EQ.IP-1)GO TO 30,**

** ISN 0037 IF(IP+1,IP1,EQ.IP-1)GO TO 30,**

** ISN 0038 IF(IP+1,IP1,EQ.IP-1)GO TO 30,**

** ISN 0039 IF(IP+1,IP1,EQ.IP-1)GO TO 30,**

** ISN 0040 IF(IP+1,IP1,EQ.IP-1)GO TO 30,**

** ISN 0041 IF(IP+1,IP1,EQ.IP-1)GO TO 30,**

** ISN 0042 IF(IP+1,IP1,EQ.IP-1)GO TO 30,**

** ISN 0043 IP1=IP**
IF(IP.LE.NP)GO TO 30

RETURN

END
SUBROUTINE PATCH(I1,H,N,L,J,J0)

DIMENSION I1(H,N)

I=0

RETURN

IDMIN=I+2*N**2+1

IF(I1(I1,JJ).EQ.0) GO TO 10

ID=I+1

ID=M

GO TO 10

RETURN

END
SUBROUTINE PRTVEC(INX, IFMT)

DIMENSION IX(NS

IF(IFMT.EQ.1)PRINT 100,IX

IF(IFMT.EQ.2)PRINT 200,IX

RETURN

FORMAT(10X111)

FORMAT(X4013)

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:

<table>
<thead>
<tr>
<th>Calling Program</th>
<th>Programs Called</th>
</tr>
</thead>
<tbody>
<tr>
<td>REGIONS</td>
<td>PET</td>
</tr>
<tr>
<td></td>
<td>VMAXI4</td>
</tr>
<tr>
<td></td>
<td>VMINI4</td>
</tr>
<tr>
<td></td>
<td>RIDER</td>
</tr>
<tr>
<td></td>
<td>SVSCI</td>
</tr>
<tr>
<td></td>
<td>DARN</td>
</tr>
<tr>
<td></td>
<td>SEQLS</td>
</tr>
<tr>
<td></td>
<td>SAWN</td>
</tr>
<tr>
<td>RIDER</td>
<td>SVSCI2</td>
</tr>
<tr>
<td></td>
<td>SVSCL1</td>
</tr>
<tr>
<td></td>
<td>SORT</td>
</tr>
<tr>
<td></td>
<td>RIDER1</td>
</tr>
<tr>
<td></td>
<td>RIDER4</td>
</tr>
<tr>
<td></td>
<td>DAWN</td>
</tr>
<tr>
<td></td>
<td>VMOV2</td>
</tr>
<tr>
<td></td>
<td>RIDER2</td>
</tr>
<tr>
<td></td>
<td>PRTVE2</td>
</tr>
<tr>
<td></td>
<td>DARN</td>
</tr>
<tr>
<td></td>
<td>VMAXI2</td>
</tr>
<tr>
<td>SEQLS</td>
<td>SORT</td>
</tr>
<tr>
<td></td>
<td>FLIPV</td>
</tr>
<tr>
<td>SORT</td>
<td>MVMRMR</td>
</tr>
<tr>
<td>RIDER1</td>
<td>SVSCI2</td>
</tr>
<tr>
<td>RIDER4</td>
<td>RIDER5</td>
</tr>
<tr>
<td></td>
<td>SVSCI2</td>
</tr>
<tr>
<td></td>
<td>PRTVE2</td>
</tr>
<tr>
<td></td>
<td>RIDER7</td>
</tr>
<tr>
<td></td>
<td>RIDER6</td>
</tr>
<tr>
<td></td>
<td>SVSCL1</td>
</tr>
</tbody>
</table>
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), IRRES(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 non-zero 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 \( \{IX(J) \mid 1 \leq J \leq N\} \) is defined as \( \{IY(J) \mid 1 \leq J \leq N\} \) if \( IY(IX(J)) = J \). 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.
**LEVEL 21.7 (JAN 73)**

**COMPILER OPTIONS**
- NAME = MAIN, UPT = 60, LINET = 56, SIZE = 0000K,
- SOURCE, EBCDIC, LIST, NOD wed, LOAD, MAP, NUE DIT, ID, N3XAE

| ISN 0002 | DIMENSION I(1X(4000), RES(21)) |
| ISN 0003 | INTEGER*2 (MIL(2100), IAI(2100), ITABL(8300), I14(4001)) |
| ISN 0005 | LOGICAL*1 IDENT(300,300) |
| ISN 0006 | DECLE_FILE 90(4000,4000, L, IAV) |

**C**
- DIM IX(MAX(2XK+2+N)) WHERE N = MAX NO. OF REGIONS EXPECTED AND
- N = MAX NO. OF BOUNDARY POINTS EXPECTED IN ANY RECORD

**ISN 0007**
- DATA NREC, MR, MSEG/4000, 300, 20/

**ISN 0008**
- DATA NEL/2100/

**ISN 0009**
- MAXX=1000000

**ISN 0010**
- MIND = 1000000

**ISN 0011**
- CALL STRTMR

**ISN 0012**
- CALL PET(0)

**ISN 0013**
- DU 10 1=N, NREC

**ISN 0014**
- READ (BIN, IX(I), J=1, N)

**ISN 0015**
- IF(N.EQ.0) GO TO 10

**ISN 0017**
- CALL VMAX(IX, N=MAXX)

**ISN 0018**
- CALL VMIN(IX, N=MIND)

**ISN 0019**
- 10 CONTINUE

**ISN 0020**
- CALL PET(1)

**ISN 0021**
- PRINT 600, MINX, MAXX, NEL

**ISN 0022**
- NEL=MAXX-MINX+1

**ISN 0023**
- PRINT 600, MINX, MAXX, NEL

**ISN 0024**
- PRINT 100, NREC, NEL

**ISN 0025**
- PRINT 200 FORMAT(I15, T15, '*'

**ISN 0026**
- NDUM=1

**ISN 0027**
- PRINT 1000, NDUM

**ISN 0028**
- 1000 FORMAT(*'

**ISN 0029**
- CALL SVSCLI(I, IX, NREC, NEL, 90, HIND, IX, NREC, NEL, 90, HIND, IX, NREC, NEL)

**ISN 0030**
- CALL PET(2)

**ISN 0031**
- PRINT 200 FORMAT(I10, 'REGION NO.' I10X 'NO. OF PIXELS')

**ISN 0032**
- CALL SVSCI(I, IX, NREC)

**ISN 0033**
- CALL SVSCI(I, IX, NREC)

**ISN 0034**
- CALL PET(2)

**ISN 0035**
- CALL PET(2)

**ISN 0036**
- PRINT 200 FORMAT(I10X 'REGION NO.' I10X 'NO. OF PIXELS')

**ISN 0037**
- CALL SVSCI(I, IX, NREC)

**ISN 0038**
- DO...1=1, NREC

**ISN 0039**
- CALL SVSCI(I, IX, NREC)

**ISN 0040**
- CALL SVSCI(I, IX, NREC)

**ISN 0041**
- CALL DARN(90, 1, XI, NEL)

**ISN 0042**
- CALL DARN(90, 1, XI, NEL)

**ISN 0043**
- CALL PET(2)

**ISN 0044**
- CALL PET(2)

**ISN 0045**
- NR=NR+1
**ISN 0046**  
**DU.40.I=1,NRI**  
**ISN 0047**  
**J=1-1**  
**ISN 0048**  
**IF(I(I),NE.O)PRINT 300,J,I(I)**  
**ISN 0050**  
**40 CONTINUE**  
**ISN 0051**  
**300 FORMAT(11X16,16X19)**  
**ISN 0052**  
**CALL PET(2)**

C REARRANGE NUMBERS IN DESCENDING ORDER OF POPULATIONS.
C LEAVE 0 AND 1 UNCHANGED SINCE THEY CORRESPOND TO EXTERIOR AND
C BOUNDARY POINTS RESPECTIVELY.

**ISN 0053**  
**CALL SEQLS(I(3),I(NRI+1),NRI-1,NRI+1)**  
**ISN 0054**  
**400 FORMAT(1X REGIONS AFTER REASSIGNMENTS:*)**  
**ISN 0055**  
**PRINT 200**  
**ISN 0056**  
**DO 50 I=1,NRI**  
**ISN 0057**  
**J=I-1**  
**ISN 0058**  
**IF(I.LE.Z)PRINT 300,J,I(I)**  
**ISN 0059**  
**CONTINUE**  
**ISN 0060**  
**DO 350 J=I(2)+1,I(I)+2**  
**ISN 0061**  
**CONTINUE**  
**ISN 0062**  
**DO 350 J=I(2)+1,I(I)+2**  
**ISN 0063**  
**CONTINUE**  
**ISN 0064**  
**DO 350 J=I(2)+1,I(I)+2**  
**ISN 0065**  
**CONTINUE**  
**ISN 0066**  
**DO 350 J=I(2)+1,I(I)+2**  
**ISN 0067**  
**CONTINUE**  
**ISN 0068**  
**CALL PET(1)**

C MODIFY REGION NUMBERS ACCORDING TO NEW ASSIGNMENTS FOUND IN IX.

**ISN 0070**  
**DO 70 I=1,NREC**  
**ISN 0071**  
**CALL DARNL90,1,W=NEL**  
**ISN 0072**  
**DO 60 1W=1,W=NEL**  
**ISN 0073**  
**CALL SAVW12**  
**ISN 0074**  
**CALL PET(2)**  
**ISN 0075**  
**STOP**  
**ISN 0076**  
**END**
SUBROUTINE FLIPV(X,N,Y)
DIMENSION X(N),Y(N)

C EQ*CE(X,Y)

N2 = N/2
N1 = N + 1
DO 10 I = 1, N2
   W = X(I)
   Y(I) = W
   Y(I) = X(I)
   X(I) = W
10 CONTINUE
RETURN
END
SUBROUTINE PET(I)
  IF(I.EQ.0) GO TO 10
  CALL TIMER(I TIME1)
  TIME=0.
  WRITE(6,200)
  200  FORMAT(10X:'BEGINNING TIMING... TIME NOW IS 0')
  RETURN
  10  CALL TIMER(I TIME2)
  TIME=(TIME2-TIME1)/100.
  TIME=TIME+TIME
  ITIME=ITIME2
  WRITE(6,100):TIME,TIME
  100  FORMAT(10X:'TIME ELAPSED SINCE LAST PRINTING OF TIME='E12.3,
             'SEC., TOTAL TIME ELAPSED='E12.3'SEC.')
  RETURN
END
SUBROUTINE PRV2(IX,N)
INTEGER*2 IX(N)
PRINT 100,IX
100 FORMAT(1X25I5)
RETURN
END
SUBROUTINE RIDER1(NTAPB, NREC, NEL, IDEVO, ICMN, IBDY, IW1, IW2, ITABL, IDENT, MR, NR, LR, MSEG, IRES, IS)

C TO IDENTIFY ALL DISTINCT CONNECTED REGIONS IN A PICTURE SEPARATED
C BY BOUNDARY LINES. THE BOUNDARY DATA ARE GIVEN AS NREC RECORDS
C SEQUENTI4L 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 REGIONS
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 DEFINE FILE IDEVO(NREC, NEL*2, L, IAV)

C DIMENSION IBDY(1)
C LOGICAL +1 IDENT(MR, NR)
C LOGICAL = 1 LW(MR, NR)
C INTEGER = 2 IM(NEL), IW2(NEL), ITABL(MR, MSEG), IS(1)
C DIMENSION IRES(MSEG)

C INITIALIZE A WORK ARRAY IW1 WITH 1'S AND IDENT WITH FALSE.
C CALL VSCL2(IW1, NEL, 1)
C CALL VSCL1(IDENT, MR, NR, FALSE.)
ISEG=1
IRES(ISEG)=1
NR=1

C LOOP ON RECORDS
C DO 10 IREC=1, NREC

C READ ONE RECORD OF BOUNDARY INFO.
C READ(NTAPB) N, (IBDY(I), I=1, N)
IF (N.NE.8) CALL SORT(IBDY, 1, N, N, 1, T, T)

C USE IW1 AND IBDY TO SET ARRAY IW2 AND MATRIX IDENT
C 30 CONTINUE
C CALL RIDER1(IW1, IBDY, ICMN, NEL, N, IW2, IDENT, MR, NR).
IF (NR.GT.0) GO TO 20
PRINT 209, IREC, NR
209 FORMAT (10X, (IREC, NR) = 216)
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 4C
IRES(ISEG)=IREC

C CCLUMN. IT IS NECESSARY THAT
C DEFINE FILE IDEVO(NREC, NEL*2, L, IAV)
IF(NR.LE.MR) GO TO 30

PRINT 163, IREC
NR = 0

100 FORMAT('ERROR CONDITION IN RIDER. SUPPLIED MR OR MSEG WAS EXCEEDEC AT RECORD NUMBER' I6.', RETURNING WITH NR=0')

RETURN

20 CALL DAWN(IDEV0, IREC, IW2, NEL*2)
CALL VMG2(IW2, NEL, I=1)

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('SEGMENT NUMBER' I3)
DO 60 CEST=1, ISEG
CALL PRTVE2(ITABL, 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(IDEV0, 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(IDEV0, IREC, IW1, NEL*2)
NR = 0
CALL VMAXI2(ITABL, MR=ISEG, NR)
RETURN
END
LEV

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-

...-
CALL 'SVSCLZ' (L1, L2, CI, JR)

IDENT (N, N) = TRUE

CALL 'SVSCLZ' (L1, L2, CI, JR)

CONTINUE

CALL 'SVSCLZ' (L1, L2, CI, JR)

IDENT (I, J) = TRUE

CONTINUE

RETURN

END
SUBROUTINE RIDER2(ITABL, NR)

CHANGE REGION NUMBERS SUCH THAT CONSECUTIVE NUMBERS ARE USED.

INTEGER*2 ITABL(NR, 2)

FIND THE SET OF NUMBERS IN ITABL(*, 1).

DO 5 I = 1, NR
   ITABL(I, 2) = 0
5 CONTINUE

DO 10 I = 1, NR
   J = ITABL(I, 1)
   IF (J .NE. 0) ITABL(J, 2) = ITABL(J, 2) + 1
   PRINT 100
   CALL PRTVE2(ITABL(1, 2), NR)
10 CONTINUE

DO 20 J = 6, 12
   DO 30 I = 1, NR
      IF (ITABL(I, 2), EQ. 0) GO TO 20
      J = J + 1
      ITABL(I, 2) = J
   20 CONTINUE
   PRINT 200
   CALL PRTVE2(ITABL(1, 2), NR)

DO 30 I = 1, NR
   ITABL(I, 1) = ITABL(ITABL(I, 1), 2)
30 RETURN

100 FORMAT("NUMBERS OF OCCURENCES OF REGION NUMBERS IN THE ABOVE TABLE(*, 1)"

200 FORMAT("LOOK-UP TABLE TO CHANGE NUMBERS IN THE ABOVE TABLE(S)")
SUBROUTINE RIDER4(IDENT,LW,MR,NR,ITABL,ISEG,IW1,NEL,IS,LAST)
LOGICAL LAST
LOGICAL*2 IDENT(MR,MR)
INTEGER*2 ITABL(MR,1),LW(MR),IS(1),IW1(NEL)

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 OCCURRING IN THE IREC'TH RECORD
C ARE FOUND; A CORRESPONDENCE ARRAY IS BETWEEN CURRENT AND NEXT SEG-
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 INFOR-
C MATION ON THE CONNECTIONS BETWEEN REGIONS IN IREC'TH RECORD.

C SECTION 1.

DO 50 I=1,MR
DO 50 J=1,MR
50 IDENT(I,J)=IDENT(I,J).OR.IDENT(J,I)
CALL RIDER5(IDENT,MR,NR,ITABL(1,ISEG),LW)
ISEG=(ISEG-1)+MR
DO 10 I=1,MR
10 IF(ITABL(I,ISEG).GT.1)ITABL(I,ISEG)=ITABL(I,ISEG)+ISEG
IF(MR.GT.NR)CALL SVSCI2(ITABL(NR+1,ISEG),MR-NR,0)
PRINT 163,ISEG
CALL PRTVE2(ITABL(1,ISEG),MR)

C SECTION 2.

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

C SECTION 3.

60 IF(LAST)RETURN
CALL RIDER6(IW1,NEL,IS,NR)
NCR=NR
PRINT 369, NR
CALL PRTVE2(IS,NR)
DO 70 I=1,NR
70 IS(I)=ITABL(IS(I),ISEG)
PRINT 460
CALL PRTVE2(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
C CALL SVSCL1(IDENT,MR,MR,.FALSE.)
DO 20 I=1,NR
IDENT(I,1)=.TRUE.
IF(I.EQ.NR)GO TO 20
II=I+1
DO 30 J=1,II,NR
30 IDENT(I,J)=IS(I).EQ.IS(J)
20 CONTINUE
RETURN
100 FORMAT(/• THE FOLLOWING IS A PRELIMINARY TABLE FOR MODIFYING SEGMENT NUMBER 'I3')
200 FORMAT(' THE FOLLOWING IS AN UPDATED TABLE FOR MODIFYING SEGMENT NUMBER '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 NEXT SEGMENT)
400 FORMAT(' ASSIGNMENTS FOR THE ABOVE REGIONS. FROM THE PRELIMINARY LO UP TABLE')
END
SUBROUTINE RIDERS(IDENT, MD, N, IT, N)
C
C TO GENERATE A TABLE IT-MAPPING J=1,...,N TO I=IT(J)=SMALLEST K
C SUCH THAT THERE EXISTS A SEQUENCE (K(ID)), ID=1,...,L) WITH K(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)=1
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
SUBROUTINE 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
SUBROUTINE SIDERT(IX, IS, N, IY, M)
INTEGER*2 IX(N), IS(N), IY(M)

C MODIFY RELEVANT ENTRIES IN IY ACCORDING TO CONNECTIVITIES FOUND IN IS.
C
IF(N.EQ.1)RETURN
MAX=IS(2)
MIN=IS(2)
CALL VMAXI2(IS(2), N-1, MAX)
CALL VMINT2(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=1, N
IF(IY(J).NE.IS(I))GO TO 20
IY(J)=IX(I)
GO TO 10
20 CONTINUE
10 CONTINUE
RETURN
END
SUBROUTINE SEQLS(X,ISEQ,N,ND)
DIMENSION X(ND,2),ISEQ(ND),T(2),TT(2)

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

C

IFLAG=0

ENTRY SEQLS(X,ISEQ,N,ND)

IFLAG=1

10 DU TO 10

ENTRY SEQLS(X,ISEQ,N,ND)

IFLAG=1

10 DU 20 I=1,N

20 ISEQ(I)=I

CALL SORT(X,1,N,NO,2,T,TT)

1IF1IFLAG=0 RETURN

CALL FLIPV(X,N,X)

CALL FLIPV(ISEQ,N,ISEQ)

RETURN

END
ISN 0066 CALL HMSRHR(A,HH,NN,T,1,K+1)
ISN 0067 IF(A(1,1).LE.T(1))GO TO 90
ISN 0069 K=1
ISN 0070 100 CALL HMSRHR(A,HH,NN,A,HH,K,K+1)
ISN 0071 K=K+1
ISN 0072 IF(T(1).LT.A(K,1))GO TO 100
ISN 0073 CALL HMSRHR(T,1,NN,A,HH,1,K+1)
ISN 0074 GO TO 90
ISN 0076 END

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR
DIMENSION IX(I)
LEVEL 21.7  JAN 73

DATE 74.3.12 06:43:19

COMPILATION OPTIONS - NAME = "MAINOPT="L3"" UTTERN = "56" SIZE = "CK"
SOUP = BCD NULL = LIST = DECK = LCAU = MAP = N UX =

150  I002   SUBROUTINE SYSC (IX, N, IS)
150  I003   INTEGER IX(N)
150  I004   DATA 1 = 1, N
150  I005   10 IX(I) = I+5
150  I006   RETURN
150  I007   END

150
LEVEL 21: I. JAN-73 -
05/360 FORTRAN I

DATE: 74.294/14.4712

COMPILER OPTIONS:
NAME: MAIN
SOURCE, BCD, NOLIST, NODECK, LOAD, MAP, NOEDIT:

ISN 0002:
SUBROUTINE: VSCL(N, X, N, L)

ISN 0003:
LOGICAL 1 X(N), L

ISN 0004:
DO 10 I = 1, N

ISN 0005:
10 IX(N) = L

ISN 0006:
RETURN

ISN 0007:
END
REPRODUCIBILITY OF THE PAGE IS POOR.
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, NTAPI, NTPO, IX, IY)

where

NREC = Number of records in the input image;
NEL = Number of pixels per record;
NEL2 = NEL+2;
NTAPI, NTPO 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.
I
LVI.
J
1
f
P
T
H.

1

74

15
f

TR
H.

AT
8

I JON
74

1

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.

1

74

1

f

P
T
H.
C   FIND THE MOST FREQUENTLY OCCURRING NUMBER AMONG THE EIGHT NEIGHBORS
C   OF IX(J,12). NOTE THAT I.LT.J.LT.NFL.

C

ISN  0004  LABEL(I)=IX(J-1,11)
ISN  0005  NUMBER(I)=1
ISN  0006  N=1
ISN  0007  J2=J-2
ISN  0008  DO 30 I=1,J
ISN  0009    IF(I.EQ.1)II=11
ISN  0010    IF(I.EQ.2)II=12
ISN  0011    IF(I.EQ.3)II=13
ISN  0012    IF(I.EQ.4)II=14
ISN  0013    IF(I.EQ.5)II=15
ISN  0014    IF(I.EQ.6)II=16
ISN  0015    IF(I.EQ.7)II=17
ISN  0016    INC=1
ISN  0017    IF(I.EQ.2) INC=2
ISN  0018  DO 20 K=KM,3,INC
ISN  0019  Do 20 L=1,N
ISN  0020  20 IF(IX(J2+K,11).EQ.LABFL(L))GO TO 40
ISN  0021  CONTINUE
ISN  0022  40 NUMBER(L)=NUMBER(L)+1
ISN  0023  10 CONTINUE
ISN  0024  30 CONTINUE
ISN  0025  LOOP
ISN  0026  GO TO 10
ISN  0027  LABEL(N)=IX(J2+K,11)
ISN  0028  NUMBER(N)=1
ISN  0029  GO TO 10
ISN  0030  60 NUMBER(N)=NUMBER(N)+1
ISN  0031  RETURN
ISN  0032  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
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:

<table>
<thead>
<tr>
<th>Calling Program</th>
<th>Programs Called</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTMATS</td>
<td>CONMAT CONMXP</td>
</tr>
<tr>
<td>CONMAT</td>
<td>SARN VMOV SVSCI INTLOG</td>
</tr>
<tr>
<td>CONMXP</td>
<td>SVSCI</td>
</tr>
</tbody>
</table>

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 III [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.
MAIN PROGRAM TO FIND AND PRINT CONTINGENCY MATRICES AND RELATED RESULTS.

INPUTS: TITLE IS AN 8 CHARACTER TITLE TO BE PRINTED ON TOP OF EACH PAGE OF OUTPUT. NREC, NEL, M, N ARE THE NUMBER OF RECORDS, NUMBER OF PIXELS PER RECORD, NUMBER OF CLASSES IN MAP 1 AND NUMBER OF CLASSES IN MAP 2, RESPECTIVELY.

THE INPUT MAPS SHOULD BE IN UNFORMATTED FORTRAN READABLE RECORDS ON UNITS 9 AND 10, ONE PIXEL/WORD.

READ(5,15) TITLE
READ(5,20) NREC, NEL, M, N

FIND AND PRINT MATRICES SHOWING NUMBERS OF AGREEMENTS AND DISAGREEMENTS OF TRANSITIONS FOR EACH PAIR OF CLASSES.

CALL CMATP(TITLE, NREC, NEL, M, N)
CALL CMATP(TITLE, NREC, NEL, M, N)
STOP

FORMAT(9(5I1))
FORMAT(4(5I1))
END
SUBROUTINE CCNIAT(TITAP, N, N, X, Y, IH)
DIMENSION IX(NEL1, 2), IY(NEL2, 2)

C THIS PROGRAM FINDS CONTINGENCY MATRICES INDICATING AGREEMENTS AND DISAGREEMENTS 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 IN MAP 2. LEFTHALF OF IH GIVES A COUNT OF AGREEMENTS AND THE RIGHHALF, DISAGREEMENTS.
C ROWS OF IH(I,J,J) CORRESPOND TO MAP 1, AND COLUMNS TO MAP 2.
C NEW NUMBERS 1, 2, 3, 4 INDICATE NO BOUNDARY, CHANGE IN VERTICAL DIRECTION, CHANGE IN HORIZONTAL DIRECTION, CHANGE IN BOTH DIRECTIONS, RESPECTIVELY, IN MAP 1. SIMILARLY COLUMN NUMBERS INDICATE TYPES OF TRANSITIONS IN MAP 2.
C
C INITIALIZE THE ARRAYS IX AND IY. THE "PREVIOUS" ROW TO ROW 1 IS CONSIDERED IDENTICAL TO ROW 1.
C
C THE PROGRAM HANDLES THE PRESENT ROW OF THE MAP 1 IN IX(0, I2) AND THE IMMEDIATELY PREVIOUS ROW IN IX(I, I1). THE ROWS OF MAP 2 ARE HANDLED SIMILARLY IN IY.
C
C LOOP ON RECORDS.
DC 10 = 1, NREC
C
C LOOP ON PIXELS.
DC 20 = 1, NEL
C
C NONPOSITIVE VALUES OF MAP LABELS ARE NOT OF INTEREST.
C
C FIND ROW AND COLUMN NUMBERS IN IH TO BE INCREASED.
C
C CHECK THE NATURE OF THE BOUNDARIES IN BOTH THE MAPS.
C
C FIND THE INCREMENTS.
C INC = NUMBER OF AGREEMENTS; INC3 = NUMBER OF DISAGREEMENTS.
C INC = INTLOG(IX(I,J), EQ.IY(I,J)) + INTLOG(IX(JP, J), EQ.IY(JP, J))
C EXCHANGE I1 AND I2

C READ NEXT RECORDS INTO IX(*,I2),IY(*,I2)

IF(I.EQ.NREC)GO TO 10
CALL SARN(NTAP1,IX(1,I2),NEL4)
CALL SARN(NTAP2,IY(1,I2),NEL4)
CONTINUE
RETURN
END
SUBROUTINE CXP

LOGICAL T, C

PRINT MATRICES SHOWING NUMBERS OF AGREEMENTS AND DISAGREEMENTS
OF TRANSITIONS FOR EACH PAIR OF CLASSES.
INPUTS: TITLE IS AN 8-CHARACTER STRAIN TITLE TO BE PRINTED ON TOP
OF EACH PAGE OF OUTPUT. NREL, N, M, N ARE THE NUMBER OF
RECORDS, NUMBER OF PIXELS PER RECORD, NUMBER OF CLASSES IN
MAP 1 AND NUMBER OF CLASSES IN MAP 2, RESPECTIVELY.

L = 0
DO 10 I = 1, M
DO 10 J = 1, N
IF (MOD (I, 4) .NE. 0) GO TO 15
WRITE (6, 10) I, J
CONTINUE
10
L = L + 1
WRITE (6, 20) L
CONTINUE
20
DO 30 I = 1, M
DO 30 J = 1, N
IF (MOD (J, 4) .NE. 0) GO TO 35
WRITE (6, 30) I, J
CONTINUE
30
DO 40 J = 1, N
WRITE (6, 40) J
CONTINUE
40
C FIND AND PRINT MATRICES SHOWING CLASSES OF EACH TYPE OF TRANSITION
OF CLASSES (I,J) IN MAPS 1, 2, THE NUMBERS OF JOINT OCCURRENCES
OF EACH TYPE OF TRANSITION IN THE TWO MAPS.

IF (MOD (I, 4) .NE. 0) GO TO 55
WRITE (6, 50) I, J
CONTINUE
50
DO 60 I = 1, M
DO 60 J = 1, N
WRITE (6, 60) I, J
CONTINUE
60
C FIND THE, THE MATRICES SHOWING CLASSES OF TRANSITION TYPES (WITHOUT REGARD
TO CLASS LABELS) AND (H2, THE MATRICES OF JOINT OCCURRENCES OF CLASS
LABELS (WITH NO REGARD TO TRANSITION TYPES).

CALL SVSCI (IHI, IHI2, M, N)

CALL SVSCI (IHI2, N, M, N)
C PRINT IH1 AND THE CORRESPONDING SIMILARITY MEASURE.
C
F 20 DO 60 K=1,4
F 20 IH(I,L)=IH(I,L)+IH(K,L,J)
C
C PRINT IH2 AND THE CORRESPONDING SIMILARITY MEASURE.
C
F 20 DO 80 I=1,N
F 20 IH2(I,J)=IH2(I,J)+IH2(I,J)
F 20 ITR=F
F 20 IF(I.LE.N)ITR=ITR*IH2(I,J)
C
C CALL SVSCI(INV1,Y,C)
C CALL SVSCI(INV2,Y,C)
C RETURN
FORMAT(' NUMBER OF POINT BY POINT SIMILARITIES=17, TOTAL=17, PERCENTAGE=7.21

FORMAT(// INVENTRY OF MAP1=/(1X1518))
FORMAT(// INVENTRY OF MAP2=/(1X1518))
END

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR.
FUNCTION INTLOG

LOGICAL L

IF (INTLOG = 0)

RETURN

END
## GTM V/S LC4120573 MOBILE BAY DATA

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

Map Size: 1624 by 866.

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>24</td>
<td>24</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>28</td>
<td>28</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>32</td>
<td>32</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>33</td>
</tr>
<tr>
<td>34</td>
<td>34</td>
</tr>
<tr>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>36</td>
<td>36</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td>37</td>
</tr>
<tr>
<td>38</td>
<td>38</td>
</tr>
<tr>
<td>39</td>
<td>39</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>41</td>
</tr>
<tr>
<td>42</td>
<td>42</td>
</tr>
<tr>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td>44</td>
<td>44</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>46</td>
<td>46</td>
</tr>
<tr>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>48</td>
<td>48</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>49</td>
<td>49</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>51</td>
<td>51</td>
</tr>
<tr>
<td>52</td>
<td>52</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>53</td>
<td>53</td>
</tr>
<tr>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>56</td>
<td>56</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>57</td>
<td>57</td>
</tr>
<tr>
<td>58</td>
<td>58</td>
</tr>
<tr>
<td>59</td>
<td>59</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>61</td>
<td>61</td>
</tr>
<tr>
<td>62</td>
<td>62</td>
</tr>
<tr>
<td>63</td>
<td>63</td>
</tr>
<tr>
<td>64</td>
<td>64</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>65</td>
</tr>
<tr>
<td>66</td>
<td>66</td>
</tr>
<tr>
<td>67</td>
<td>67</td>
</tr>
<tr>
<td>68</td>
<td>68</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>69</td>
<td>69</td>
</tr>
<tr>
<td>70</td>
<td>70</td>
</tr>
<tr>
<td>71</td>
<td>71</td>
</tr>
<tr>
<td>72</td>
<td>72</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>73</td>
<td>73</td>
</tr>
<tr>
<td>74</td>
<td>74</td>
</tr>
<tr>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td>76</td>
<td>76</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>77</td>
<td>77</td>
</tr>
<tr>
<td>78</td>
<td>78</td>
</tr>
<tr>
<td>79</td>
<td>79</td>
</tr>
<tr>
<td>80</td>
<td>80</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>81</td>
<td>81</td>
</tr>
<tr>
<td>82</td>
<td>82</td>
</tr>
<tr>
<td>83</td>
<td>83</td>
</tr>
<tr>
<td>84</td>
<td>84</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>85</td>
<td>85</td>
</tr>
<tr>
<td>86</td>
<td>86</td>
</tr>
<tr>
<td>87</td>
<td>87</td>
</tr>
<tr>
<td>88</td>
<td>88</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>89</td>
<td>89</td>
</tr>
<tr>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td>91</td>
<td>91</td>
</tr>
<tr>
<td>92</td>
<td>92</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>93</td>
<td>93</td>
</tr>
<tr>
<td>94</td>
<td>94</td>
</tr>
<tr>
<td>95</td>
<td>95</td>
</tr>
<tr>
<td>96</td>
<td>96</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLASS NUMBER IN MAP 1</th>
<th>CLASS NUMBER IN MAP 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>97</td>
<td>97</td>
</tr>
<tr>
<td>98</td>
<td>98</td>
</tr>
<tr>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>
### Pathways Showing Counts of Each Type of Transition

**Map Size:** 1624 by 966

#### Class Number in Map 1 = 1, Class Number in Map 2 = 1

<table>
<thead>
<tr>
<th>Class Number in Map 1</th>
<th>23967</th>
<th>6259</th>
<th>6979</th>
<th>4568</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Number in Map 2</td>
<td>306</td>
<td>192</td>
<td>196</td>
<td>198</td>
</tr>
<tr>
<td></td>
<td>349</td>
<td>192</td>
<td>236</td>
<td>142</td>
</tr>
<tr>
<td></td>
<td>146</td>
<td>59</td>
<td>71</td>
<td>81</td>
</tr>
</tbody>
</table>

#### Class Number in Map 1 = 1, Class Number in Map 2 = 2

<table>
<thead>
<tr>
<th>Class Number in Map 1</th>
<th>1329e</th>
<th>5322</th>
<th>5682</th>
<th>4462</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Number in Map 2</td>
<td>521</td>
<td>236</td>
<td>17C</td>
<td>163</td>
</tr>
<tr>
<td></td>
<td>616</td>
<td>207</td>
<td>304</td>
<td>202</td>
</tr>
<tr>
<td></td>
<td>173</td>
<td>65</td>
<td>74</td>
<td>73</td>
</tr>
</tbody>
</table>

#### Class Number in Map 1 = 1, Class Number in Map 2 = 3

<table>
<thead>
<tr>
<th>Class Number in Map 1</th>
<th>17013</th>
<th>4910</th>
<th>4920</th>
<th>3117</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Number in Map 2</td>
<td>369</td>
<td>212</td>
<td>240</td>
<td>140</td>
</tr>
<tr>
<td></td>
<td>1116</td>
<td>282</td>
<td>241</td>
<td>197</td>
</tr>
<tr>
<td></td>
<td>343</td>
<td>69</td>
<td>74</td>
<td>55</td>
</tr>
</tbody>
</table>

#### Class Number in Map 1 = 1, Class Number in Map 2 = 4

<table>
<thead>
<tr>
<th>Class Number in Map 1</th>
<th>1887</th>
<th>245</th>
<th>423</th>
<th>283</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Number in Map 2</td>
<td>128</td>
<td>7</td>
<td>44</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>53</td>
<td>25</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>1</td>
<td>9</td>
<td>3</td>
</tr>
</tbody>
</table>
Matrix showing totals of each type of transition

Map size = 1024 by 866.

Transition similarities = 984662; total = 1341588; percentage = 73.46

Contingency table

<table>
<thead>
<tr>
<th></th>
<th>44016</th>
<th>31576</th>
<th>33669</th>
<th>29971</th>
<th>6426</th>
<th>2976</th>
</tr>
</thead>
<tbody>
<tr>
<td>45423</td>
<td>191374</td>
<td>31773</td>
<td>3476</td>
<td>66</td>
<td></td>
<td></td>
</tr>
<tr>
<td>35323</td>
<td>61245</td>
<td>391893</td>
<td>537</td>
<td>3189</td>
<td>168</td>
<td></td>
</tr>
<tr>
<td>14318</td>
<td>257</td>
<td>917</td>
<td>307292</td>
<td>5516</td>
<td>41552</td>
<td>7685</td>
</tr>
<tr>
<td>99743</td>
<td>4326</td>
<td>35795</td>
<td>39000</td>
<td>41552</td>
<td>287</td>
<td></td>
</tr>
<tr>
<td>3497</td>
<td>2476</td>
<td>2508</td>
<td>526</td>
<td>752</td>
<td>1213</td>
<td></td>
</tr>
</tbody>
</table>

Number of point by point similarities = 577399; total = 1341588; percentage = 72.95

Inventory of Map 1

| 12135 | 163796 | 522378 | 407102 | 96774 | 10569 |

Inventory of Map 2

| 143391 | 201752 | 497515 | 605342 | 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:

<table>
<thead>
<tr>
<th>Calling Program</th>
<th>Programs Called</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPSUMP</td>
<td>SVSCL1</td>
</tr>
<tr>
<td></td>
<td>DCLARE</td>
</tr>
<tr>
<td>DCLARE</td>
<td>CLAM</td>
</tr>
<tr>
<td></td>
<td>PRTMXP</td>
</tr>
<tr>
<td></td>
<td>SVSCI</td>
</tr>
<tr>
<td></td>
<td>SARN</td>
</tr>
<tr>
<td></td>
<td>VMOV</td>
</tr>
<tr>
<td>CLAM</td>
<td>SVSCI</td>
</tr>
<tr>
<td>PRTMXP</td>
<td>MXINX</td>
</tr>
</tbody>
</table>

9.8 PERFORMANCE SPECIFICATIONS

9.8.1 Storage

This main program, presently limited to $\text{NEL} \leq 2000$ and $M \times N \leq 200$, 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 $NREC = \text{NEL} = 2000$ the program will take approximately 7 minutes of CPU time on IBM 360/65.

178
9.8.3 Restrictions

\[ \text{NEL} \leq 2000, \text{M} \times \text{N} \leq 200 \]

9.9 METHOD

This program first sets an M by N matrix \( LTABLE \) by

\[
LTABLE(i, j) = 'H' \quad \text{for} \quad i \neq j
\]

\[
LTABLE(i, j) = \text{blank} \quad \text{for} \quad i = j.
\]

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=2, LINESIZE=56, SIZE=8000K,
SOURCE, EBCDIC, NPLIST, NODECK, LOAD, MAP, NOEDIT, TD, NOREF

DIMENSION IX(160), IY(200)
LOGICAL*1 LY(200), LTABL(200)
LOGICAL*1 BLANK/*ETC/H/*
LOGICAL*1 TITLE(72)
READ 1*, TITLE
PRINT 2*, TITLE
READ, 910, NREC, NEL, M, N
PRINT 410, NREC, NEL, M, N
CALL SVCL((LTABL(I, M), N, ETCH)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
MN=MING(M, N)
SUBROUTINE DCLAREXHREC, NEL, MN, LTABL, IX, IY, LY...

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 PLUS, MINUS, BLANK INDICATES THAT CLASS NUMBERS 1 AND 2 IN MAP 1 AND 2 CORRESPOND. THE INPUT MAPS 1 AND 2 SHOULD BE ON UNITS 8 AND 10 HAVING INTEGER NUMBERS 0 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 AS INTERIOR POINTS;
2) DIFFERENCE MAP ON UNIT 12 CONTAINING CODES (BLANK, PLUS, MINUS) AND H, WRITTEN AS LOGICAL 1 RECORDS.

DIMENSION IX(NEL,3), IY(NEL), IH(256)
LOGICAL*1 LY(NEL), LTABL(MN), BDY
LOGICAL*1 ETCH/F, PLUS/+, BLANK/-, DGT/0/
DATA 1w/0/ CALL PRTMXP(IH,M,N)
CALL SYSTIL(IN,IV,256,5)
CALL SARN(IH,IV,254)
CALL VMOV(IH,NEL,IX(1), 21)
DO 10 I=1, NREC
IF(IX(J).LE.XOR.IY(J)).LE.A) GO TO 30
LY(J) = LTABL(IX(J),1,Y(J))
IF(LY(J).EQ.BLANK) GO TO 35
BDY = IX(J), NE. IX(J), 11, OR. IX(J), 21, NE. IX(J), 31 OR. J, 11, AND IX(J), 1, 21, NE. IX(J), 1, 21 OR. J, LT, NEL, 10, IX(J), 1, 21, NE. IX(J), 1, 21.
IF(60Y)LY(J) = PLUS
GO TO 35
LY(J) = 0
CONTINUE
CALL VMOV(IX(1), 21, NEL, IX(1), 1)
CALL VMOV(IX(1), 31, NEL, IX(1), 2)
WRITE(12)LY I=0,256
DO 65 I=1,256
L(Y(1)) = 1
IF(LY(1).EQ.DOT) GO TO 65
NIOT=NIOT+1
IF(LY(1),EQ,DOT) GO TO 60
L(Y(1)) = 1
IF(LY(1),EQ,DOT) GO TO 60
CONTINUE
PRINT 100,L(Y(1),,IHH),PERCT
CONTINUE
PRINT 100,L(Y(1),,IHH),PERCT
RETURN
END

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR.
SUBROUTINE PRTMxPMN(IA,H,M)

DIMENSION IA(H,M)

CALL MXINX(IA,H,M,1,NJ)

DO 50 I=1,M

PRINT 300, (IA(I,J), J=1,NJ)

FORMAT(1X,15I3)

50 RETURN

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:

PLTPX
SARN
SAWN
SVSCI
VMOV

10.8  PERFORMANCE SPECIFICATIONS

Storage:

Difference coding - \( \frac{42K}{K=1024} \) bytes
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.

\[
\begin{align*}
IBUF (1) & = 1: \text{ all regions (pixels) in the scene} \\
& = 2: \text{ interior regions (pixels) only} \\
& = 3: \text{ boundary pixels only}
\end{align*}
\]

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.

\[
\begin{align*}
IBUF (2) & = 1: \text{ each change from each class (in the set to be defined for Map 1) to a corresponding individual class (in the set to be defined for Map 2).} \\
& = 2: \text{ change from each class (in the set to be defined for Map 1) to every class (other than itself in the set to be defined for Map 2).} \\
& = 3: \text{ change from the union of classes (in the set to be defined for Map 1) to the union of classes (in the set to be defined for Map 2) excepting of course those pixels that remain unchanged.} \\
& = 4: \text{ change from the union of classes (in the set to be defined for Map 1) to individual classes (in the set to be defined for Map 2) disregarding of course the pixels that were originally in the corresponding class.} \\
& = 5: \text{ change from individual classes (in the set to be defined for Map 1) to union of classes (in the set to be defined for Map 2) excepting those pixels that remain in the same class.}
\end{align*}
\]
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

\[ \text{NEXT} = 4 + \text{NC1}, \quad \text{IBUF (3)} \neq 0 \]
\[ = 5, \quad \text{IBUF (3)} = 0 \]

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:

- Denote a set of sequentially numbered classes in Map 1 and 2.
- Denotes total changes (interior and boundary).
- Denotes changes from union of classes specified for Map 1 to union of classes specified for Map 2.
- Number of classes of Map 1 and 2.

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:

- Denotes changes in the interior regions only.
- Denotes changes from each of classes specified for Map 1 to every class specified for Map 2.
- Number of classes of Map 2.
- Set of classes for Map 2.
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:

```
number of classes
of Map 1

3

1

2

set of classes
for Map 1

1

2

2

1

denotes changes
in boundaries
only

denotes changes from a class specified for Map 1 to a corresponding class specified for Map 2

number of classes
of Map 2

set of classes
for Map 2
```

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:

```
denotes a sequential set of classes for Map 1

set of classes
for Map 2

2

4

0

5

2

3

6

denotes changes in interior regions only

denotes changes from union of classes specified for Map 1 to individually specified classes of Map 2

number of classes
of Map 2

number of classes in this sequential set
```
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=', 13)
WRITE(6, 100) M
STOP
END

SUBROUTINE OVRMSK(IN, JN, KN, IX, IY, ITAB, NSL, IS, M)
DIMENSION IX(IS), IY(IS), ITAB(IS)
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
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 L=(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(TX,LX,ISET,JSET,KSET,IH,IBUF,TABLE)
DIMENSION IX(IOOO),IH(256),ISET(10),JSET(10),KSET(10),IBUF(26)
LOGICAL *1 IX(IOOO)
INTEGER TABLE(256)
READ(5,10) NREC,NF
READ(20) M
990 FORMAT(1H1,10X,'THE INPUT CHANGE DEPICTION REQUEST CODE IS: ',2613)
800 FORMAT(2613)
10 FORMAT(2616)
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 5C
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-L
IF(IBUF(I).NE.G) GO TO 16
C NOW, ISET(1),I=1,NC1 AND JSET(1),2=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,NF1,*4)
DO 85 LL = 1,NEL
LX(LL) = TABLE(TX(LL) + 1)
85 MX = LX(LL) + 1
IHMX = IH(MX) + 1
CALL SAWN(8,LX,NEL)
80 CALL SAWN(10,LX,NEL)
REWIND 8
CALL PLTPIX(1,8,1,NREC,1,NEL,255,2,LX)
CALL PRTLVL(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 PR1BL (IBUF, JSET, JSET, KSET, NC1, NC2, I, J, IH)

DIMENSION IBUF(I), JSET(NC1, JSET(NC2)

DIMENSION TH(I), 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,'CHANGE DEPICTION MAP SHOWING THE ENTIRE PIXELS OF')

301 FORMAT(10X,'EXCLUDING OFCOURSE THOSE PIXELS THAT REMAIN IN THEIR')

302 FORMAT(10X,'THE REST OF THE SCENE OF INTEREST IS SHOWN AS BLANK')

303 FORMAT(10X,'WITH THE OUTSIDE OF THE SCENE DEPICTED BY PLUS SIGN')

304 FORMAT(10X,'THE ORIGINAL PAGE IS POOR')

305 FORMAT(10X,'REPRODUCIBILITY OF THE')

306 FORMAT(10X,'ORIGINAL PAGE IS POOR')

IB1 = IBUF(1)
IB2 = IBUF(2)

GO TO (10, 20, 30), IB1

GO TO 20

GO TO 40

PRINT 201

GO TO 40

PRINT 202

GO TO 20

PRINT 203

GO TO (50, 60, 70, 80, 90), IB2

PRINT 301, JSET(I)

PRINT 204

IF(ISFT(I), NE, JSET(I)) PRINT 302, IH(256), JSET(I)

GO TO 99

PRINT 301, JSET(I)

PRINT 204

IF(ISFT(I), NE, JSET(J)) PRINT 302, IH(256), JSET(J)

GO TO 99

PRINT 301, (ISFT(L), L = 1, NC1)

PRINT 204

PRINT 302, IH(256), (JSET(L), L = 1, NC2)

PRINT 303

GO TO 99

LK = 0

DO 93 L = 1, NC1

IF(ISF(L), EG, JSET(J)) GO TO 93

LK = LK + 1

KSET(LK) = JSET(L)

CONTINUE

PRINT 301, (KSET(L), L = 1, LK)

PRINT 204

PRINT 302, IH(256), JSET(J)

GO TO 99

LK = N

PRINT 301, JSET(I)

PRINT 204

DO 94 L = 1, NC2

IF(JSET(L), EQ, JSET(I)) GO TO 94

94 CONTINUE

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR
\( LK = LK + 1 \)

\[ \text{KSET}(LK) = \text{JSET}(L) \]

**CONTINUE**

PRINT 302, IH(256), (KSET(L), L=1, LK)

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 (1X, '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 **

NUMBER OF CLASSES IN MAP 1 AND 2

MS = (M+1)*M
MM = MS + M*M
IB1 = IBUF(1)
IB2 = IBUF(2)
TABLE(1) = 6C
CALL SVSC(TABEL(2),MM,C)
I2 = I
I3 = J
J2 = J
J3 = J
GO TO (20,30,40,50,60),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 II = I2,I3
L = (ISET(II)-I)*M + 1
DO 11 J1 = 1,M
K = L + J1
IF(ISET(II).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) = 85
IF(IB1.NE.2) TABLE(K+MS) = 85
11 CONTINUE
DO 10 J1 = J2,J3
K = L + JSET(J1)
IF(ISET(II).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
REFERENCES:


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.


CSC

COMPUTER SCIENCES CORPORATION

Major Offices and Facilities Throughout the World