Edge Detection and Adjacency Classification in Digital Data Via the Gradient Vector

Michael John Russell

Follow this and additional works at: https://openprairie.sdstate.edu/etd

Recommended Citation
Russell, Michael John, "Edge Detection and Adjacency Classification in Digital Data Via the Gradient Vector" (1976). Electronic Theses and Dissertations. 4970.
https://openprairie.sdstate.edu/etd/4970

This Thesis - Open Access is brought to you for free and open access by Open PRAIRIE: Open Public Research Access Institutional Repository and Information Exchange. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of Open PRAIRIE: Open Public Research Access Institutional Repository and Information Exchange. For more information, please contact michael.biondo@sdstate.edu.
EDGE DETECTION AND ADJACENCY CLASSIFICATION
IN DIGITAL DATA VIA THE GRADIENT VECTOR

BY

MICHAEL JOHN RUSSELL

A thesis submitted
in partial fulfillment of the requirements for the
degree Master of Science, Major in
Engineering, South Dakota
State University
1976

SOUTH DAKOTA STATE UNIVERSITY LIBRARY
This thesis is approved as a creditable and independent investigation by a candidate for the degree, Master of Science, and is acceptable for meeting the thesis requirements for this degree. Acceptance of this thesis does not imply that the conclusions reached by the candidate are necessarily the conclusions of the major department.
EDG E DETECTION AND ADJACENCY CLASSIFICATION
IN DIGITAL DATA VIA THE GRADIENT VECTOR

Abstract

MICHAEL J. RUSSELL

Under the supervision of Professor Gerald D. Nelson

Detection of edges in imagery represented by a matrix of data and classification of the data surrounded by an edge are investigated. An edge detection algorithm is developed and used to locate edges in digitized imagery. An algorithm called the adjacency classifier is developed to classify all groups of adjacent data points that are surrounded by edges. These two algorithms incorporate three important characteristics of the gradient vector: high gradients are inherent at edges, low gradients are inherent within homogeneous objects and gradient vector directions are perpendicular to the edge direction.

Computer programs which implement both algorithms are documented.

Digitized images of LANDSAT-1 satellite multispectral scanner data are analyzed with both algorithms. The accuracy of the edge detection algorithm is evaluated for LANDSAT-1 satellite scenes of land-lake edges. The documentation for SYMOVER, a gray-tone mapping program developed to display digitized images or digital data with a line printer is included.
ACKNOWLEDGEMENTS

The author would like to express appreciation to Dr. Donald G. Moore, Soil Scientist at the Remote Sensing Institute and to Dr. Gerald D. Nelson of the Electrical Engineering Department who were helpful in giving their time and comments.

The author wishes to thank Mr. Victor I. Myers, Director of the Remote Sensing Institute and the Remote Sensing Institute staff for enabling the research to be performed. The work was supported by NASA SKYLAB contract NAS 9-13337. Aircraft data collected by the Remote Sensing Institute under USGS contract number 14-08-001-13576 and USBR contract number 14-06-700-7466 were analyzed.

The subroutines written by Ronald Greve of the SDSU computer center and the programing help were invaluable to the software written for the research performed.

The author is grateful to his wife, Helen, who transformed a rough draft into typing and provided the inspiration to finish this thesis.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>i</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>ii</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>iii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>vii</td>
</tr>
<tr>
<td><strong>Chapter</strong></td>
<td></td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Importance of Edges</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Literature Review</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Definition of the Research Objective</td>
<td>10</td>
</tr>
<tr>
<td>1.4 Research Contribution</td>
<td>11</td>
</tr>
<tr>
<td>1.5 Preview of the Thesis</td>
<td>12</td>
</tr>
<tr>
<td>2. EDGE DETECTION ALGORITHM</td>
<td>15</td>
</tr>
<tr>
<td>2.1 The Edge Model</td>
<td>15</td>
</tr>
<tr>
<td>2.2 The Detection of Edges in One-Dimensional Data</td>
<td>18</td>
</tr>
<tr>
<td>2.3 The Detection of Edges in Two-Dimensional Data</td>
<td>22</td>
</tr>
<tr>
<td>3. THE ADJACENCY CLASSIFIER</td>
<td>28</td>
</tr>
<tr>
<td>3.1 The Adjacency Method of Statistic Generation for Fields</td>
<td>29</td>
</tr>
<tr>
<td>3.2 Statistical t-Test to Determine Field Classes</td>
<td>34</td>
</tr>
<tr>
<td>4. APPLICATIONS OF THE EDGE DETECTION AND ADJACENCY CLASSIFIER ALGORITHMS</td>
<td>38</td>
</tr>
</tbody>
</table>
4.1 Analysis of Land-Lake Data Collected by the LANDSAT-1 Satellite .............................................. 39

4.2 Detection of Edges in LANDSAT-1 Satellite Data Associated with Field Patterns in an Agricultural Area. 48

4.3 Other Applications of Edge Detection ................................................................. 51

5. SUMMARY AND RECOMMENDATIONS .............................................................. 52

5.1 Summary ................................................................. 52

5.2 Recommendations ................................................................. 53

APPENDIX

A. TWO EDGE DETECTION FORTRAN PROGRAMS .................................................. 57

A.1 Subroutine Edge ................................................................. 57

A.2 FORTRAN Listing of Subroutine Edge ................................................................. 60

A.3 Edge Detection Version II ................................................................. 63

A.4 FORTRAN Listing of Edge Detection Version II .................................................. 67

A.5 Choice of the Gradient Threshold ................................................................. 71

B. THE ADJACENCY METHOD COMPUTER PROGRAM ........................................ 74

B.1 The Adjacency Method of Statistic Generation .................................................. 76

B.2 A FORTRAN Listing of the Adjacency Method .................................................. 77

B.3 The Choice of the Gradient Threshold ................................................................. 84

C. DISPLAY OF DIGITAL DATA WITH A LINE PRINTER ........................................... 86

C.1 Limitations of Program SYMOVER ................................................................. 86

C.2 Use of SYMOVER Parameter Cards ................................................................. 91

C.2a SYMOVER Parameter Card One ................................................................. 91

C.2b SYMOVER Parameter Card Two ................................................................. 94

C.2c SYMOVER Parameter Card Three ................................................................. 94
C.2d SYMOVER Symbol Parameter Card Type One.............. 97
C.2e SYMOVER Symbol Parameter Card Type Two.............. 98

D. USE OF THE EDGE DETECTION AND THE ADJACENCY METHOD PROGRAMS... 101
  D.1 Use of Subroutine Edge........................................ 101
  D.2 Edge Detection Version II JCL................................ 104
  D.3 SYMOVER JCL..................................................... 105
  D.4 Adjacency Method JCL.......................................... 106
  D.5 Example of Run Deck for Symbol Maps of Edges............. 107
  D.6 Example of Run Deck for Symbol Maps of Fields............. 108

BIBLIOGRAPHY.................................................................. 110
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>An Ideal Edge and the Gradient Magnitude in One Dimensional Data</td>
<td>17</td>
</tr>
<tr>
<td>2.2</td>
<td>An Example of the Realistic One-Dimensional Edge Model</td>
<td>21</td>
</tr>
<tr>
<td>2.3</td>
<td>A Flow Chart of a Two-Dimensional Edge Detection Algorithm</td>
<td>23</td>
</tr>
<tr>
<td>2.4</td>
<td>The Importance of the Gradient Vector Direction to a Two-Dimensional Edge Detector</td>
<td>26</td>
</tr>
<tr>
<td>3.1</td>
<td>A Flow Chart of the Adjacency Method of Statistical Generation for Fields of Points Surrounded by Edges</td>
<td>30</td>
</tr>
<tr>
<td>3.2</td>
<td>A Flow Chart of the Statistical Test for Significantly Different or Similar Fields</td>
<td>35</td>
</tr>
<tr>
<td>4.1</td>
<td>(a) LANDSAT-1 Band 7 Imagery of Lakes in Codington County, South Dakota and (b) Simultaneous Aircraft Coverage</td>
<td>41</td>
</tr>
<tr>
<td>4.2</td>
<td>(a) Line-Printer Gray-Tone Map of Lakes and (b) Edges Located with the Edge Detection Algorithm</td>
<td>42</td>
</tr>
<tr>
<td>4.3</td>
<td>(a) Line-Printer Gray-Tone Map of Lakes and (b) Lakes Classified with the Adjacency Method</td>
<td>47</td>
</tr>
<tr>
<td>4.4</td>
<td>(a) LANDSAT-1 Band 5 Imagery of Agriculture and (b) Simultaneous Aircraft Imagery</td>
<td>49</td>
</tr>
<tr>
<td>4.5</td>
<td>(a) Line-Printer Gray-Tone Map of Digitized LANDSAT-1 Imagery of Agriculture and (b) Edges Located in the Digitized Agricultural Image</td>
<td>50</td>
</tr>
<tr>
<td>C.1</td>
<td>Example of SYMOVER Parameter Deck</td>
<td>92</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>LAKE STATISTICS GENERATED BY THE ADJACENCY METHOD</td>
</tr>
<tr>
<td>4.2</td>
<td>LAKES CLASSIFIED WITH THE ADJACENCY CLASSIFIER</td>
</tr>
<tr>
<td>A.1</td>
<td>USE OF SUBROUTINE EDGE</td>
</tr>
<tr>
<td>A.2</td>
<td>USE OF EDGE DETECTION VERSION II</td>
</tr>
<tr>
<td>B.1</td>
<td>LIMITATIONS OF THE ADJACENCY METHOD COMPUTER PROGRAM</td>
</tr>
<tr>
<td>C.1</td>
<td>A FOUR CHARACTER OVERSTRIKE SEQUENCE</td>
</tr>
<tr>
<td>C.2</td>
<td>SYMOVER PARAMETER CARD ONE</td>
</tr>
<tr>
<td>C.3</td>
<td>SYMOVER PARAMETER CARD TWO</td>
</tr>
<tr>
<td>C.4</td>
<td>SYMOVER PARAMETER CARD THREE</td>
</tr>
<tr>
<td>C.5</td>
<td>USE OF PARAMETER NDISP</td>
</tr>
<tr>
<td>C.6</td>
<td>SYMOVER SYMBOL PARAMETER CARD TYPE ONE</td>
</tr>
<tr>
<td>C.7</td>
<td>SYMOVER SYMBOL PARAMETER CARD TYPE TWO</td>
</tr>
</tbody>
</table>
CHAPTER 1.
INTRODUCTION

Digital images of the entire earth are gathered once every eighteen days with a multispectral scanner mounted on the LANDSAT-1 satellite. The sensors used in the multispectral scanner sense the earth's radiation in four different wavelength intervals. These data are transmitted to earth, stored on digital tapes and converted into four black and white images, one for each spectral region. The repetitive coverage of the earth is necessary for an evaluation of dynamic features on the earth, for example: surface water area measurements, soil moisture programs such as irrigation scheduling, crop yield estimates, etc. Automatic digital processing algorithms are necessary because of the large amount of digital data collected by satellites and the value of monitoring real-time information. The edge between spectrally different ground targets is one type of information required to aid in utilizing these data.

1.1 Importance of Edges

One important image analysis tool is a computer algorithm to locate edges in images. Edges are located at the boundaries between adjacent homogeneous data regions. Prewitt (1) cited that, "Edge enhancement and detection is one of the oldest goals of analog and digital picture processing". Several reasons for the importance of edge detection in digital picture processing follow:
1. Regions of different spectral reflectance characteristics are separated by edges in imagery.

2. The shape, area or perimeter of regions can be calculated if edge locations are known.

3. The statistics of the data in the regions of data can be used to train an M-class classifier.

4. Several images of the same scene can be registered by superimposing edges detected in the images.

Quantitative information is required from the data for certain applications. The length, width, area or perimeter around a field or lake cannot be computed without precise location of edges. Statistics describing spectral information such as the average, the standard deviation and the number of data values in a data region surrounded by an edge are important. The edges around a data region must be detected before statistics of the data region can be computed.

Images from repetitive coverage of the same scene require data registration. A method of registration of several images of the same scene was illustrated by Anuta (2). Anuta registered images by enhancing edges around landmarks in one image and superimposing them on the same edges in the other images.

An M-class classifier requires that training sets of data be chosen from the image for all M classes of data. The user of an M class classifier is required to mentally decide where the edges surrounding different classes are in order to choose training sets.
A computer algorithm which locates edges simplifies the process of choosing training sets in digital data. A computer algorithm which automatically extracts regions of data that are surrounded by edges could be used unsupervised to extract training sets of data for an M-class classifier. Edge detection and region extraction applied to an entire image would allow the use of all regions of data that are surrounded by edges as training sets for an M-class classifier.

1.2 Literature Review

A major part of the literature reviewed in this chapter includes edge detection techniques based on the characteristics of the two dimensional gradient vector. Articles which suggest the psychological importance of edges and a basic thesis of picture processing are included to support the search for an edge detection algorithm. After edge detection techniques which use information about the gradient vector are surveyed, several other more complex edge detectors are mentioned.

The importance of edges to psychophysics was suggested by Prewitt (1):

There are numerous psychological and neurophysiological experiments which suggest that boundaries contain a very significant proportion of the total useful pictorial information. Psychopictorics is a subfield of psychophysics which uses images of natural scenes as the pictorial stimuli. Because of the enormous complexity of even one image, individual images are analyzed only in terms of specific objects selected by an observer.
Lipkin (3) surveyed three psychophysical variables used to define objects: contrast and border, shape and geometry, and texture. She suggested that a basic thesis in psychopictorics and also picture processing is that properties obtained from the array of luminances that compose an image can be used to define a specific object in the image.

The data gradient vector contains important properties which are calculated from the array of digitized luminances that compose an image. The gradient vector can be used to locate edges around individual objects in the data. Objects in a typical scene of the earth might be cities, forests, lakes or agricultural fields. Edges are enhanced with differential operators because of three important characteristics of the gradient vector:

1. High gradient vector magnitudes are inherent at edges between objects.
2. Low gradient vector magnitudes are inherent within objects.
3. The gradient vector direction is perpendicular to the edge direction.

Several references from the psychophysical world about the various gradients in images were cited by Roberts (4). Roberts used three criteria to judge the choice of a differential operator:

1. The edges produced should be as sharp as possible.
2. The background should produce as little noise as possible.
3. The intensity of lines produced at edges should correspond to a human's ability to perceive the edge in the original image.
He also determined that background noise was reduced when two orthogonal differential operators were used to calculate a gradient vector magnitude rather than a single differential operator.

According to psychological theory, equally apparent edges have equal magnitudes when the square roots of the intensities are used rather than the intensities. The square root operation also gives extra line sharpness and reduces background noise.

Roberts (4) selected a four-by-four square of gradient magnitude values as a feature point of an edge if the square contained gradients above a certain threshold. Correlations of lines with slopes of 0, 1, infinity, and -1 with the original data around selected feature points of an edge was used by Roberts to detect edge locations. A heuristic approach was used to eliminate multiple interconnections of edges, to delete short sections of non-closed edges, and to add sections of missing lines. The heuristic method proposed by Roberts uses assumptions to best locate edges in images of three-dimensional solids. Because four-by-four squares of gradient magnitudes were used to select feature points of an edge the minimum distance between edges is four elements. This implies that edges closer than four elements apart can not be distinguished.

Graham (5) used a simple gradient-based detector in a transmission system, which decreased the bandwidth required to transmit an image. The bandwidth was decreased because a low resolution version of the low pass image and the coded edges in the image were transmitted. The edge coding system used a threshold on gradient magnitudes to select starting points on an edge contour and a local operator to follow along the edge.
The local operator compared the eight nearest neighbors to a starting point and followed the edge by choosing the largest of the eight neighbors as the next point on the edge.

Prewitt (1) described many approaches to enhance and detect edges. Edge enhancement increases the ability to visualize the edge while edge detection spatially locates the edge. She discussed enhancement by the following methods: the gradient magnitude, the Laplacian and a modified form of the Laplacian. Several methods of calculation of orthogonal partial derivatives for approximation of the gradient vector magnitude were presented. The gradient vector magnitude was also approximated as the largest of the partial derivatives in the eight directions of adjacent neighbors to a point. A modified form of the Laplacian, which smoothed data along gradient contours and enhanced only the gradients normal to contours, was described. Prewitt also described several methods of edge detection from data processed with edge enhancement methods. Thick contours were normally obtained from edge-enhanced data by applying a threshold to the gradient magnitudes. Gaps in the gradient contours were filled and the contours were reduced to a single line thickness to locate the position of edges. Prewitt surveyed several algorithms called thinning algorithms which were designed to reduce thick lines to thin lines. The success of these algorithms was not presented.

Prewitt (1) described a sequential tracking algorithm which uses both gradient vector magnitudes and directions to track edges. The algorithm first computes the gradient threshold value from a histogram
of gradient values. When a sharp change in the gray scale is located, a fine scan mode is initiated by the tracking algorithm. A local search is made along the direction of the gradient vector for larger gradient magnitudes until a maximum gradient is found. Next the search moves orthogonal to the gradient, to the next candidate edge point. In this manner continuous edges are located. A routine to bridge gaps of a certain length or less can be used to close gaps in edges.

Anuta (2) utilized gradient vector magnitudes to enhance edges in several different images of the same scene to register the images. He used a gradient threshold on the gradient vector magnitudes to obtain thick lines at high gradient edges. Thick lines which were located at similar positions in the scenes in separate images were superimposed to register the images together. Better registration results could have been obtained by Anuta if the width of the thick lines at edges could have been reduced to a single element.

Rosenfeld, Thurston and Lee (6, 7, 8) said that the size of the neighborhood used in enhancing edges determines the widths of the edges detected. When small neighborhoods are used, microedges can be located, but edges between textured regions require neighborhoods large enough to average out the detail in the texture. They enhanced edges with two passes through the data; one pass for vertical edge enhancement and a second pass for horizontal edge enhancement. Differences of the data over multisized neighborhoods were used for edge enhancement. The maximal differences (in the direction across an edge) were chosen as the edge locations. In their procedure, there was no provision to assure that maximums found with the vertical and horizontal edge detectors were
found while crossing perpendicularly across edges. Special detectors were described for spots, curves, and streaks in images. The curve detector could be used to find collinear maximal points located by the vertical and horizontal edge detectors. If the maximal points are not collinear, this detector eliminates the points from the set of edges. Several iterations with the curve detector are necessary to eliminate most of the noncollinear points. Noncollinear points that are close to collinear edges can not be eliminated with this approach.

Example outputs of several basic edge enhancement techniques were illustrated by Carton, et al (9). He showed examples where the difference between different-sized neighborhoods was used to approximate directional derivatives. This method used the difference which was the largest as an approximation to the gradient. Differences between averages of large blocks of points was suggested for an approximation of the gradient which is less sensitive to minor edges or noise. The idea of suppressing non-maximums in the direction across an edge was presented for providing an output of a thinner edge. Edge detectors which used the Laplacian, high-pass filtering and gray level contouring were also described. Carton showed comparative results of all of these methods applied to the same data.

Davis (10) surveyed in detail many of the edge detection techniques available before November 1973. High frequency filtering, directional differencing and the gradient proposed by Roberts were several classical methods described for enhancing edges. Davis pointed out that these operations do not detect thin edges because points close to edges are
also enhanced. However, there were many procedures in the literature at that time which could be used to reduce thick edges to thin edges.

Two edge detection methods developed by Rosenfeld, Thurston and Lee (7), use products of the differences between average gray levels of pairs of neighborhoods of different sizes. These were described by Davis. The product of differences and averaging aspects of Davis' procedure reduced the effects of noise.

Davis (10) described both the advantages and disadvantages of the following edge detection approaches: Griffith's edge detector for the block world, Hueckel's operator which determines the edge element that will best fit a given region, Chow's variable threshold method, Martelli's heuristic search method, Montanari's dynamic programming method of detecting a system of curves, Kelly's edge detection by planning and Shihai's system which "understands" the block world. These approaches include some of the more exotic edge detection algorithms in the literature before November 1973. Davis (10) includes a very good summary of each of these.

Griffith (11, 12) discussed an optimal way of using properties of the block world to detect edges. He developed a mathematical model for the different regions of intensities that occur in the block world. Using Bayesian probability he wrote a closed form for an edge centered in a rectangular region of data. Many of his assumptions pertain only to the block world. He ignored regions, which had lines or edges not centered in a rectangular region, because of computational complexities.

The approach proposed by Hueckel (13, 14) to edge detection was to determine the edge element which best fits the intensities in a
circular disc region of points. He used a series expansion and truncation of the series in the frequency domain. The complexity of this approach limits its usefulness.

Chow (15) used a variable thresholding technique on raw data to locate edges. He assumed that, given a histogram of two classes around an edge, the histogram was bimodal. A threshold was chosen from bimodal histograms of subregions of the data to locate edges. A danger in this approach occurs when the region size is not small enough and thus a mixture of several classes is contained in the histogram. The method works on images with relatively few objects, but is not directly applicable to images with many different sized objects.

1.3 Definition of the Research Objective

Many of the classical approaches to edge enhancement, which use differentiation, were surveyed in the literature review. Of these approaches, the Roberts gradient which uses derivatives in two orthogonal directions was shown to work better than differentiation in a single direction. Edges are enhanced when the gradient vector is applied because of three important characteristics:

1. High gradient vector magnitudes are inherent at step edges between objects.
2. The gradient vector direction is perpendicular to the edge direction.
3. Low gradient vector magnitudes are inherent within objects.

One objective of this study was to utilize all three characteristics of the gradient vector for the analysis of a two-dimensional
digitized image. Both Prewitt (1), and Rosenfeld and Thurston (6) suggested that edges are detected in gradient magnitudes by suppressing non-maximums which are located while crossing edges. However, the procedure to locate non-maximums while crossing perpendicular to edges is not documented in the literature. Another objective was to develop, document and show examples of an edge detection algorithm which uses the information available from the gradient vector. None of the edge detection algorithms in the literature, which use the gradient vector to locate edges, are concerned with the calculation of statistics of regions of data surrounded by edges and the classification of these regions. The final objective was to develop, document, and show examples of an algorithm which computes statistics of data surrounded by edges.

1.4 Research Contribution

The seven main research contributions of this thesis are:

1. A unique edge detection algorithm for locating edges between homogeneous data regions was developed.
2. A tutorial explanation of the flow of the entire edge detection process when applied to Griffith's (11,12) mathematical model of an edge was presented.
3. Two FORTRAN computer programs for use as edge detectors were written, documented and applied to digitized images.
4. Results obtained from digitized LANDSAT-1 satellite images of lakes and agricultural scenes were included for support of the usefulness of the algorithm when applied to real data.
5. A unique region extraction and classification technique called the adjacency classifier was developed from the characteristics of the gradient vector.

6. Computer programs to implement the adjacency classifier were written, documented, and applied to digitized images.

7. An efficient computer program, SYMOVER, for displaying digital data with a computer line printer was written.

Greve of the SDSU computer wrote several IBM 370 assembly subroutines for SYMOVER. Greve's contribution is listed in Appendix D. The contribution made by the author was a FORTRAN main program which allows the user of SYMOVER to easily use the available assembly subroutines. SYMOVER was ten times more efficient than the line display program in operational use at the Remote Sensing Institute at South Dakota State University before the conception of SYMOVER. SYMOVER allows for overstrikes of symbols in order that the line printer map is a gray-tone map. Results of the use of SYMOVER are included in the Figures of Chapter 4.

1.5 Preview of the Thesis

In Chapter 2 an edge model is defined. This edge model starts with the assumption that an ideal edge is a step function. A realistic edge is derived from an ideal edge by addition of distortion and noise. The characteristics of the gradient vector direction and magnitude in a one-dimensional model of edges are used to develop an edge detection algorithm. The one-dimensional edge detection algorithm is modified to a two-dimensional edge detector. The flow chart of the two-dimensional
edge detection algorithm is presented along with two FORTRAN computer programs for implementation in Appendix A.

In Chapter 3 an approach for the generation of statistics of homogeneous regions of data, that are surrounded by edges, is presented. This approach is called the adjacency method because each group of points that are adjacent and are surrounded by edges are labeled by a different number called the field number. A statistical test which divides the field statistics into statistically different or similar fields is described. The adjacency method combined with this statistical test is called the adjacency classifier algorithm. The FORTRAN program for the adjacency method is presented in Appendix B.

Two examples of results obtained from the application of the edge detection algorithm are illustrated in Chapter 4. The examples of edge detection which are presented include:

1. The detection of edges between land and water in LANDSAT-1 band 7 (0.8-1.1µm) data. The location is Coddington County in South Dakota.

2. The detection of edges between agricultural fields of corn and fallow in LANDSAT-1 satellite band 5 (0.6-0.7µm) data. The location is in Kansas.

Simultaneous aircraft coverage of both of these areas and the LANDSAT-1 imagery are included for comparisons. These data are also displayed as gray-tone line printer maps. Overstriking of line printer characters produces gray-tone maps. This procedure is implemented by a computer program called SYMOVER. These procedures are discussed in Appendix C. The LANDSAT-1 band 7 data of lakes in Coddington County, South Dakota,
which are analyzed with the edge detection algorithm, are also analyzed with the adjacency classifier. Other applications of the edge detection algorithm are referenced and briefly described.

A summary of the thesis and recommendations for further studies of edge detection and the adjacency classifier are included in Chapter 5. Modifications are recommended for other possible uses of the gray tone line printer program, SYMOVER. An algorithm which combines both the adjacency classifier method of field classification and edge detection is also recommended.
CHAPTER 2.

EDGE DETECTION ALGORITHM

In this chapter Griffith's (11,12) mathematical model of an edge is used to create realistic edges from ideal edges. An ideal edge is assumed to be located where a step change in data occurs. First an algorithm is developed to locate an ideal edge in one-dimensional data. This algorithm is modified to locate edges from Griffith's model of a more realistic edge. The one-dimensional edge detection algorithm is modified to locate edges in realistic two-dimensional data. The edge detection algorithm is different from other edge detection approaches that use a differential operator because both the direction and the magnitude of the data gradient-vector of the two-dimensional surface are used to locate edges. A FORTRAN computer program of the edge detection algorithm is listed in Appendix A with procedures for its use. Two examples of results obtained from the application of the two-dimensional edge detection algorithm for lake and agricultural scenes obtained by the LANDSAT-1 satellite data are included in Chapter 4. These results indicate that the edge detection theory, which is proposed and implemented as computer programs, is useful for image analysis.

2.1 The Edge Model

An ideal edge between two homogeneous data clusters is located where a step change in the data occurs. Data in which no edges exist have no data gradients. An ideal, continuous and one-dimensional edge between two classes is illustrated with solid lines in Figure 2.1a.
The first derivative of an ideal edge (Figure 2.1b) is an impulse function (infinite in magnitude) at the location of the edge. The mathematical model of a realistic edge proposed by Griffith creates a realistic edge by adding distortion and noise to an ideal step edge.

Distortion of the edge is produced by the convolution of a point spread function with a step edge. This distortion is a natural effect called blurring which is produced by imaging systems or scanners. A multispectral scanner (MSS) collects data from the LANDSAT-1 satellite in four wavelength regions. Originally the MSS data are stored on computer compatible tapes. Digital LANDSAT-1 data can either be obtained from the original digital tapes or they can be acquired from transparencies which are digitized with a scanner. The point spread function for the MSS scanner produces blurring which distorts ideal edges. The digital LANDSAT-1 data used in Chapter 4 were obtained from a digitized transparency which was produced by assigning light levels to the data on the original tapes and exposing a film. The digital data were obtained by digitizing the transparency with a scanner which is part of the SADE (Signal Analysis and Dissemination Equipment) at the Remote Sensing Institute in Brookings, South Dakota. Therefore, the LANDSAT-1 data which were used for examples were affected by several point spread functions. The point spread functions for the MSS scanner, the digital MSS to image process and the digitization by SADE all produce a certain amount of blurring which distorts ideal edges.
Figure 2.1. An Ideal Edge and the Gradient Magnitude in One-Dimensional Data. The line in (a) and arrow in (b) represent a continuous ideal edge and its first derivative. The dots are a digitized blurred edge and its first derivative.
The other quantity that is modeled in Griffith's realistic edge model is noise. There is a noise component created by the LANDSAT-1 MSS, the production of an image from MSS computer tapes, and the digitization of an image with SAGE. Noise, in a general sense, is any data fluctuation which results in the misclassification of an edge. This can be electronic noise, noise added by the various data conversions, or actual minor data fluctuations. The distribution of the noise is not investigated by this thesis. However, the fact that noise exists in realistic data requires a provision in the edge detection algorithm to eliminate false edges.

An algorithm to locate edges in continuous data with ideal edges can be developed with minimal effort. To emphasize the reasoning for steps in a general edge detection algorithm for realistic two-dimensional surfaces, a simple algorithm is first developed for ideal edges in one-dimensional data. This algorithm is gradually modified and applied to one-dimensional data that are first digitized, then distorted with a point spread function and finally noise is added. The edge detection algorithm for realistic one-dimensional data is modified and applied to a realistic two-dimensional surface. By using this approach, reasoning for individual steps in the edge detection algorithm are understood more easily.

2.2 The Detection of Edges in One-Dimensional Data

In this section algorithms are developed to locate edges in the proposed realistic model of a one-dimensional edge. An algorithm is developed first for ideal edges, secondly it is modified for ideal edges
which are blurred, and finally it is modified for blurred ideal edges with noise added.

The magnitude and direction of the gradient vector are used to locate edges. In one-dimension, the gradient vector magnitude is the absolute value of the first derivative. The sign of the first derivative is the direction of the gradient vector. A positive first derivative implies that the data gradient is toward the right and a negative first derivative implies that it is to the left.

An algorithm to locate edges in continuous data with ideal edges can be developed using simple logic. Edges are located where the absolute value of the gradient is infinite. All locations with zero gradient magnitudes are located within homogeneous data classes. An edge in continuous ideal data and the gradient (first derivative) are shown with continuous lines in Figure 2.1.

Continuous data are digitized before a computer algorithm can be used to locate edges. An approximation for the gradient magnitude at the midpoint between two digitized points is the difference between the two data magnitudes. The gradient magnitude at an edge is approximately equal to the difference between the average data values on either side of the edge and it is zero elsewhere. Therefore, an algorithm for locating ideal edges is to locate all nonzero gradient magnitudes.

The first step in converting an ideal edge to a realistic edge model is to convolve the data with a point spread function. Digitized data of a blurred edge and the data gradients are illustrated with dots in Figure 2.1. The gradient magnitude increases when an edge is
approached, reaches a maximum value at the location of the edge and decreases after the edge is crossed. When three adjacent gradient magnitudes are compared and the middle one is larger than the other two, an edge is located. Using this method to locate edges requires that all three adjacent points in the data are searched for maximums. A gradient threshold which limits this search for maximums to those locations where gradient magnitudes are greater than zero reduces the number of points searched to only areas near edges, and therefore saves many calculations.

The final step in converting an ideal edge to a realistic edge model is to add noise to ideal data which are first blurred with a point spread function. Examples of these data and the gradients are shown in Figure 2.2. Edges are located at maximum gradient values when three adjacent gradient magnitudes are compared. However, not all maximums are located at edges. If the magnitude of the noise is smaller than the gradient magnitudes at edges, a gradient threshold can be used to eliminate those maximums that have magnitudes less than or equal to the largest noise level. Only data gradients above a certain gradient threshold are searched for maximums. The locations of these maximums are "true" edge locations if the noise level produces smaller gradients than the gradient threshold. A gradient threshold which is too large eliminates some gradient maximums which are located at edges. A gradient threshold which is too small allows gradient maximums which are located in the noise to be misclassified as edges. A gradient threshold is shown in Figure 2.2b as a dashed line.
Figure 2.2 An Example of the Realistic One-Dimensional Edge Model. The first derivative of these data and a gradient threshold are shown. The solid line represents an ideal edge distorted by blurring. The dots represent a blurred edge with noise added. The dashed lines in (b) represent the gradient thresholds. There are two gradient threshold dashed lines because the threshold is applied to the absolute value of the first derivative. An edge is located at point Max 3.
2.3 The Detection of Edges in Two-Dimensional Data

An algorithm was developed in the last section to locate edges in a realistic one-dimensional data model. The steps in a two-dimensional edge detection algorithm are deeply rooted in the following ideas which were apparent when locating edges in one-dimensional data. High gradient magnitudes are located within homogeneous classes. To locate edges when ideal edges are blurred, it is necessary to compare all groups of three adjacent gradients and to locate those with midpoints larger than the adjacent points. A gradient threshold was necessary to eliminate false edges or maximums that were located because of noise added to the model. Noise was defined as any fluctuation in the data that created a data gradient that was larger than those in the homogeneous data and that could be mistaken as an edge.

In this section the one-dimensional edge detection algorithm is modified to work on a two-dimensional data surface. The edge detection algorithm developed here is programmed in FORTRAN and presented in Appendix A. An explanation is included on the use of the FORTRAN program. The results of the algorithm applied to digitized LANDSAT-1 images are included in Chapter 4. The flow chart of this edge detection algorithm is presented in Figure 2.3 and described next.

For a continuous two-dimensional surface \( F(x, y) \) the magnitude, \(|\nabla F(x, y)|\), and the direction, \( \theta \), of the data gradient vector are defined as:

\[
|\nabla F(x, y)| = \left( \left( \frac{\partial F}{\partial x} \right)^2 + \left( \frac{\partial F}{\partial y} \right)^2 \right)^{\frac{1}{2}}
\]

\[
\theta = \arctan \left( \frac{\partial F}{\partial y} / \frac{\partial F}{\partial x} \right)
\]
CALCULATE APPROXIMATE GRADIENT MAGNITUDES AT THE MIDPOINTS, $x_i$, OF ALL FOUR ADJACENT DATA POINTS

$$|\text{GRAD}| = |vF(x)| = (\partial F/\partial y)^2 + (\partial F/\partial x)^2$$

CALCULATE A QUANTIZED GRADIENT VECTOR DIRECTION, $\theta_j$, FROM $\text{TAN}(\theta_j)$

$$\text{TAN}(\theta_j) = \frac{\partial F/\partial y}{\partial F/\partial x}$$

QUANTIZED $\theta_j = (1, 2, 3, 4)$

$$\text{TAN}(\theta_j) = \frac{a - d}{b - c}$$

Figure 2.3. A Flow Chart of a Two-Dimensional Edge Detection Algorithm. The algorithm is explained on pages 22-24.
The partial derivatives, \( \frac{\partial F}{\partial x} \) and \( \frac{\partial F}{\partial y} \), in this expression are taken in two orthogonal directions. The gradient vector at a point on a two dimensional surface is defined as the maximum slope of the surface at that point. The direction of the gradient vector at a point is the direction on the surface of the greatest slope at that point.

The magnitude, \( |\nabla F(x,y)| \), of the gradient vector at the midpoint of four points and the direction, \( \tan \theta \), are approximated as:

1) \[ |\nabla F(z)| = \left( (a-d)^2 + (b-c)^2 \right)^{\frac{1}{2}} \]

or

2) \[ |\nabla F(z)| = |a-d| + |b-c| \]

3) \[ \tan \theta = \frac{a-d}{b-c} \]

The characters a, b, c and d represent digitized data values and the gradient magnitude and direction are approximated at the midpoint \( z \).

The two orthogonal directions X and Y are used to calculate the gradient vector direction and magnitude are diagonals in the data.

The difference in data values \( (a-d) \) approximates \( \left( \frac{\partial F}{\partial y} \right) \), the partial derivatives of the data with respect to the Y direction at point \( z \).

The difference \( (b-c) \) approximates \( \left( \frac{\partial F}{\partial x} \right) \), the partial derivative of the data with respect to the X direction at point \( z \). Equation one above produces a more accurate approximation to the gradient vector magnitude than equation two, but it takes more steps to implement equation one on a computer. Because of this fact and the fact that both equations produce gradient magnitudes which work equally well to locate edges, equation two is the better of these two. Prewitt (1) has several other approximations for the gradient vector magnitudes which are more precise, but require considerably more calculations.
Only four gradient vector directions are used by the edge detection algorithm. These four directions point towards the eight adjacent neighbors of a point. The four quantized directions are used by the edge detection algorithm to route the algorithm toward four different edge detectors. These quantized directions, $\theta_i$, are computed directly from the $\tan \theta$, which is equal to $(a-d)/(b-c)$. For example the direction number one is the gradient direction when the ratio $(a-d)/(b-c)$ is between the values of the $\tan (22.5^\circ)$ and $\tan (-22.5^\circ)$ or between $\tan (157.5^\circ)$ and $\tan (202.5^\circ)$. A similar procedure is used to compute the other three directions.

The gradient vector direction is not used to locate edges in one dimensional data. However, the gradient vector direction is necessary to locate edges in two-dimensional data. Edges are located with the one-dimensional edge detector by searching for maximums among gradient magnitudes larger than the gradient threshold. An edge is located when the midpoint of three adjacent gradient values are numerically larger than both adjacent points. An edge is not located at point E, in Figure 2.4 if the gradients at points C and D are compared to the gradient at point E because no maximum is detected. However, an edge is correctly detected when the gradients at points A and B are compared to the gradient at point E. The edge was correctly detected only where the three gradients are compared in a direction which is perpendicular to the edge direction. Gradient vector directions are always perpendicular to the edge direction.
Figure 2.4 The Importance of the Gradient Vector Direction to a Two-Dimensional Edge Detector. An edge is located at point E by comparing gradient magnitudes adjacent along the gradient vector direction and finding a maximum. A maximum is not found by comparing adjacent gradients in the edge direction.
In the flow chart in Figure 2.3, points a and c are adjacent points to a midpoint at b. When the gradient magnitude at point b is greater than or equal to the magnitude at points a and c, an edge has been located. The equal condition is required because it is possible to have two adjacent points which have equal gradient magnitudes that are larger than the gradient threshold. These points would not be detected as edges with only the greater-than-condition which is proposed for continuous data. There are four separate detectors that detect edges in this manner, one for each of the four possible gradient directions. A gradient threshold is used to eliminate false edges created by noise in the data.
CHAPTER 3. THE ADJACENCY CLASSIFIER

In this chapter an algorithm is described which computes several statistics for each homogeneous data area surrounded by an edge. The algorithm, called the adjacency method, was developed from several important observations of data gradient magnitudes. These observations are that homogeneous data areas consist of low gradient magnitudes and that higher gradients are located at edges. The edge detection algorithm described previously locates positions of edges in data regions where the gradient is larger than a gradient threshold. The adjacency method divides the data into separate regions of adjacent points which are surrounded by high gradient edges. Regions of homogeneous data points are called fields. Each point in a region of adjacent points is labeled with the same field number. The number of data points, the sum of the data values, and the sum of the squared data values are calculated for all different numbered fields. A statistical t-test is applied to the average data values of the fields found by the adjacency method to classify the fields. The t-test performs classification by separating fields which are statistically different and combining fields which are statistically similar at a specified confidence level.

The adjacency classifier consists of two algorithms. One algorithm is called the adjacency method, and the other is a statistical t-test which is used to determine if two fields are the same or different. The adjacency method, which obtains statistics for all fields or homogeneous
data regions, is described in Section 3.1. The statistical t-test is described in Section 3.2.

An example of results obtained from the adjacency classifier is included in Chapter 4. A FORTRAN listing of a computer program which implements the adjacency method is included in Appendix B along with a description of its use. For the purpose of this thesis the statistical test was programmed on an HP-25 calculator. A general FORTRAN program which implements the statistical analysis is available at the Remote Sensing Institute at SDSU.

3.1 The Adjacency Method of Statistic Generation for Fields

A flow chart of the adjacency method, the portion of the adjacency classifier algorithm that obtains statistics for homogeneous data regions, is presented in Figure 3.1. Field numbers are assigned to homogeneous data regions and statistics are calculated on a temporary basis as individual data points are searched. Temporary field numbers are assigned because it is possible to assign different field numbers to appendages of the same field if the field is complex in shape. The algorithm combines into one field all points that are adjacent and surrounded by a high gradient edge.

If several different homogeneous data regions, which are assigned different field numbers, have adjacent points the adjacency method changes this discrepancy. This discrepancy is handled in the program by storing in the computer a table of the fields that are adjacent. This table is used to convert the temporary field numbers into permanent field numbers. The adjacent field table records all
Figure 3.1 A Flow Chart of the Adjacency Method of Statistic Generation for Fields of Points Surrounded by Edges.
temporary fields that are adjacent. The temporary fields are the
locations in this table while the numbers assigned to the locations are
the composite lowest field numbers of the temporary fields that are
adjacent. That is, if temporary fields 5, 10 and 15 are adjacent:
then the 5th, 10th and 15th slots in the adjacent field table would
have a 5 in them. The IBM 370 assembly translate command uses a
table defined in exactly the same manner as an adjacent field table
which is limited to 255 fields. The table defines a one-to-one
mapping of temporary fields into permanent fields. The translate
command efficiently converts the temporary field numbers, which
are assigned to homogeneous regions of data, into permanent field
numbers.

There are three paths in the adjacency method flow chart of
Figure 3.1. They are surrounded by dashed lines. A point, denoted
as point (b) in the flow chart, is assigned either a field or an
edge number in path 1 or path 2. These two paths are separated by
a comparison of the gradient magnitude at point (b) with the gradient
threshold.

When the gradient magnitude at point (b) is less than the
gradient threshold, point (b) is assumed to be located within some
field by the algorithm. Points which are adjacent to point (b),
like point (a) which is to the left or like point (c) which is above,
are assigned either field numbers or edge numbers before point (b).
If the point to the left, point (a), is not a field point, a new
table of the temporary field statistics is started. The table
includes the number of points, the sum of the data values and the sum of the squared data values. If the point to the left, point (a) is already a field point, the statistics at point (b) are added to the temporary field statistics that were previously started. This continues until an edge is reached to the right of point (b). The point above (b) is checked for a previously assigned field number that is adjacent or an edge number. The field numbers for the different adjacent fields and the number of adjacent fields are stored when Path 1 in Figure 3.1 of the algorithm is followed. This information is used in Path 2 by the algorithm to assign a temporary field number to the string of adjacent points to the left of the edge.

Path 2 in the algorithm is followed when the gradient at point (b) is greater than or equal to the gradient threshold. When this occurs point (b) is assumed to be located in an edge region. If the previous point, point (a), is also an edge point, an edge is coded in the temporary field data set and the algorithm goes to the next point. If the previous point, point (a), belongs to a field, a temporary field number is chosen. The field number is chosen based on the adjacent field table and the list of adjacent fields compiled while the algorithm evaluated previous points which were members of a field. This field number is coded into a data set, the temporary field data set, and the temporary statistics are stored under the temporary field numbers. All fields that were adjacent are used to search the adjacent field table for the lowest adjacent field number that is adjacent to the previous field points. The lowest adjacent field number for homogeneous regions is used to update the adjacent field
table for all temporary field numbers that are adjacent. This field number also codes the field number in the field points to the left of the edge. If the string of adjacent field points is not adjacent to any fields a new temporary field number is chosen to code the field points. A comparison is made of the new field number and 256. The number of temporary field numbers is restricted to 256 by the assembly translate command. If the field number is less than 256, the edge point is coded and the algorithm goes to the final section. This section, called the translation routine, converts temporary field numbers into permanent field numbers.

The algorithm branches to the translation routine when all data have been assigned either edge numbers or temporary field numbers, or when 256 field numbers have been used. In this section permanent field numbers are assigned to the temporary fields, and the temporary statistics are combined and stored as permanent statistics. Spaces in the sequence of permanent field numbers used in the adjacent field table are eliminated. These spaces are the result of eliminating the temporary field numbers which are adjacent to other temporary fields. There are unused permanent field numbers or space for every temporary field number which was eliminated. If the program ends before going through all the data, because the temporary field number is 256, this does not mean that there will be 256 permanent fields in the data. There will actually be 256 minus the number of temporary fields lost because they are adjacent to another field minus five (four numbers are reserved for designating edge points and one for insignificant fields).
After the adjacent field table is transformed into a table containing the lowest number of adjacent fields, the statistics are moved into a vector accessible with the new permanent field numbers. The translate command uses the adjacent field tables as a translation table and converts the temporary field data set into permanent fields and edges. This permanent field set is written onto a tape for permanent storage. The mean and standard deviation are calculated for each field. The field number, the sum of the squares, the mean and the standard deviation are listed as the output of the program. The gray-tone line printer program SYMOVER which is described in Appendix C can be used to assign symbols to the fields stored on the permanent data set tape and to provide the gray-tone line printer maps.

3.2 Statistical t-Test to Determine Field Classes

The algorithm used to statistically classify the homogeneous fields separated from the data with the adjacency method is described in this section. This algorithm was programmed for a HP-25 calculator. The HP-25 program, which implements the statistical t-test, is not included. Results of the adjacency classification algorithm, when applied to LANDSAT-1 band 7 (0.8-1.1 \( \mu \)m) data of lakes in Codington County, South Dakota, are included in Chapter 4.

The flow chart of the statistical t-test used for classification is presented in Figure 3.2. First the field means are ranked from highest to lowest. The algorithm starts at the top of this list and
RANK FIELD MEANS FROM HIGHEST TO LOWEST

COMPARE LARGEST MEAN WITH NEXT LOWEST MEAN

TEST SIGNIFICANCE
$t$-test
0.05 level

YES

ASSIGN A CLASS TO THE LARGER MEAN

COMPARE THE SMALLER MEAN WITH THE NEXT LOWEST MEAN

NO

COMPUTE NEW STATISTICS BY COMBINING MEANS AND VARIANCE TO RETAIN ORTHOGONALITY

COMPARE THE NEW MEAN WITH THE NEXT LOWEST MEAN

Figure 3.2. A Flow Chart of the Statistical Test for Significantly Different or Similar Fields.
compares the largest mean value with the next largest mean value. The statistical t-test for two independent samples that are assumed to have the same variance but have different means $M_1$, $M_2$ is used as a significance test of the null hypothesis. The null hypothesis is $H_0: M_1 = M_2$ and the alternate hypothesis is $H_1: M_1 \neq M_2$. Given two independent populations $\{x_1, x_2, x_3 \ldots x_{n_1}\}$ and $\{y_1, y_2, y_3 \ldots y_{n_2}\}$ the t statistic is calculated as follows:

$$M_1 = \frac{1}{N_1} \sum_{i=1}^{N_1} (x_i) \quad M_2 = \frac{1}{N_2} \sum_{i=1}^{N_2} (y_i)$$

$$t_{\text{value}} = \frac{M_1 - M_2}{\sqrt{\frac{1}{N_1} \left( \sum_{i=1}^{N_1} (x_i)^2 - (N_1)(M_1)^2 \right) + \frac{1}{N_2} \left( \sum_{i=1}^{N_2} (y_i)^2 - (N_2)(M_2)^2 \right)}}{\sqrt{\frac{1}{N_1} + \frac{1}{N_2}} \left( \frac{N_1 + N_2 - 2}{N_1 + N_2 - 2} \right)}$$

The null hypothesis is tested with a two-tailed t-test at the .05 probability level.

The null hypothesis is accepted when the calculated t-value is less than a t-value at the given number of degrees of freedom in a table of t-values at the .05 level. Acceptance of the null hypothesis implies that the two populations do not have significantly different mean values. Therefore, the statistics of the two compared populations are combined and a new composite statistic is calculated. This process continues by comparing this new mean with the next lowest average on the list.

The null hypothesis fails if the computed t-value is greater than the t-value from the statistical tables. This implies that the two
compared populations have significantly different mean values at the .05 confidence level. When this happens the larger mean compared in this test is declared to be a significantly different class of data and the other mean is compared with the next lowest mean in the list. The mean which is classified as a significantly different class is either one field or a composite of similar fields which were determined to not be significantly different. The composite mean $M_j$ and standard deviation $S_j$ are calculated for each significantly different class of data determined by the adjacency classifier.

\[
M_j = \frac{1}{N_1} \sum_{i=1}^{N_1} x_i
\]

\[
S_j = \sqrt{\frac{\sum_{i=1}^{N_1} (x_i - \bar{x})^2 / N_1}{N_1 - 1}}
\]

Results obtained from the entire adjacency classification algorithm are included in the next chapter.
APPLICATIONS OF THE EDGE DETECTION AND ADJACENCY CLASSIFIER ALGORITHMS

Several applications of the edge detection algorithm and the adjacency classifier are included in this chapter to illustrate the usefulness of these two approaches during the analysis of real two-dimensional data. Results obtained from the analysis of a land-lake scene and an agricultural scene sensed by the multispectral scanner aboard the LANDSAT-1 satellite are presented. Photographs of the LANDSAT-1 images and some images collected from aircraft on the same day are included for comparison. The digitized LANDSAT-1 images and results obtained from both the edge detection algorithm and the adjacency classifier are illustrated as gray-tone maps, which were produced by a computer line printer. Procedures for using the edge detection algorithm, the adjacency classifier and the line printer display program are included in Appendices A, B, and C, respectively. A description is presented in Appendix D of the entire procedure necessary for production of final edge maps and maps of data classes calculated by the adjacency classifier.

Example output products from the edge detection algorithm and the adjacency classifier algorithm for land-lake edges located on digitized LANDSAT-1 satellite images are included in the first section of this chapter. The accuracy of the edge detection algorithm is evaluated for land-lake edges located in these data. The adjacency classifier is used to classify the lakes and land.
In the next section, results of the edge detection algorithm, when applied to LANDSAT-1 data of fallow fields and corn fields, are illustrated. In the last section, other applications of the edge detection algorithm are referenced and briefly described.

Digitized LANDSAT-1 data are affected by many variables. Digitization was accomplished with an image dissector which is part of SADE at the Remote Sensing Institute in Brookings, South Dakota. The light setting, the vertical and horizontal scale factors and the log or linear processing are variable factors in SADE, which were not evaluated. The transparencies digitized were assumed to be at an original scale of 1:1,000,000. Each digitized LANDSAT-1 data point represents 27.7 meters by 27.7 meters on the ground for the system geometry chosen to digitize these transparencies. The accuracy of the edge detection algorithm is a function of all of the variables which affect the data. Spatial accuracy of the edge detection algorithm is presented.

Edge detection was applied to the following data:

1. Digitized images of the earth taken from aircraft
2. Digitized S190A images taken from SKYLAB
3. SKYLAB conical S192 multispectral scanner digital data
4. Digitized images of diseased leaves
5. Digitized LANDSAT-1 satellite images

4.1 Analysis of Land-Lake Data Collected by the LANDSAT-1 Satellite

Results obtained from the application of the edge detection algorithm and the adjacency classifier to LANDSAT-1 band 7 data of
a land-lake scene are presented next. Band 7 data are from a multi-
spectral scanner which has a sensor which is sensitive in the reflective
infrared (0.8-1.1µm). This spectral window was selected because it
generally results in good film density contrasts between water and
upland vegetation. Original digital LANDSAT-1 data are converted from
digital tapes to black and white transparencies at the Goddard Space
Center in Maryland. The 1:1,000,000 scale transparency was converted to
digital data at the Remote Sensing Institute at South Dakota State
University by their Signal Analysis and Dissemination System or SADE.
LANDSAT-1 digital data acquired in this manner are nearly spatially
correct in contrast to data on the original tape product which are not
spatially correct. SADE digitizes with a Dicomed scanner. Data obtained
from SADE for this example conform very well to the data model described
in the edge detection chapter.

The aircraft and LANDSAT-1 land-lake images are illustrated in
Figure 4.1. The aircraft imagery is a black and white print of color
infrared film collected by the Remote Sensing Institute at South Dakota
State University. It was collected on July 4, 1974, under USGS contract
number 14-08-001-13576.

Gray-tone maps of the digitized LANDSAT-1 data and the edges located
with the edge detection algorithm are illustrated in Figure 4.2. The
computer program SYMOVER, which is described in Appendix C, is used to
generate gray-tone line printer maps. The darker tones are created by
overstriking several line printer characters.

The accuracies obtained in locating edges with the edge detection
algorithm are evaluated for the LANDSAT-1 band 7 data, which are
Figure 4.1 (a) LANDSAT-1 Band 7 Imagery of Lakes in Codington County, South Dakota and (b) Simultaneous Aircraft Coverage.
Figure 4.2 (a) Line Printer Gray Tone Map of Lakes and (b) Edges Located With the Edge Detection Algorithm. The scale of each picture element is approximately 91 feet by 91 feet. In (a) the light is lake and the dark is upland vegetation.
illustrated in Figure 4.1a. Areas of surface water bodies in the
LANDSAT-1 data are computed from symbol maps of edges which are located
by the edge detection algorithm by counting the number of data points
located within the surface water edges. Only half of the area of points
at edges is counted into the area of the surface water body. These
surface water areas are compared to areas calculated with a planimeter
from scaled aircraft images (Figure 4.1b) that were collected on the same
day. A planimeter is a mechanical device which calculates the area of a
lake when the planimeter is traced around the shoreline of the lake. The
sun glint that is present in the aircraft image of Figure 4.1b had no
effect on the surface water area calculations.

The variables in the digitization process need to be analyzed
before the actual accuracy limits can be calculated. However, results
obtained from several different digitizations of land-lake edges from
LANDSAT-1 band 7 data have been evaluated. These accuracies are
documented by Russell et al (16). Edges were detected within one
resolution cell of the actual boundaries or within 27.7 meters.

The ground coverage of each picture element from the LANDSAT-1
1:1,000,000 transparencies digitized by SADE was 27.7 meters by 27.7
meters. Lakes with surface areas greater than 3 hectares were calculated with less than 10% error. The error obtained while locating edges
and measuring areas of vegetated lakes is larger because there is less
spectral contrast between the vegetated cover and water than there is
between land and water.
The adjacency classifier was also applied to these LANDSAT-1 land lake data. The land and lakes were separated and their statistics generated with the adjacency method. The statistics of the land and lake areas are displayed in Table 4.1. The land-lake statistics were classified with the t-test discussed in Chapter 3. The combination of the adjacency method and this t-test classifier is called the adjacency classifier. Lakes with similar characteristics in these data classes are displayed in Table 4.2 along with the composite average data value and the standard deviation of each class. The field numbers in this table are the same as those in Table 4.1. The position of these fields are also denoted in Figure 4.3 by the same numbers. The gray-tone maps were generated by a line printer with the computer program SYMOVER.

One output of the adjacency method program is a computer tape which labels all fields and edges in the data with the same field numbers used to label the statistics. All edge points are coded with numbers from 1 to 5 and all points with fields are coded with the appropriate field number. The program SYMOVER converts the field numbers into different symbols for a hard copy print of the fields.

The results obtained by the adjacency classifier for this land lake set of data are helpful in analyzing the types of surface water bodies that are surrounded by edges by relating reflectance to water properties. The most that the edge detection procedure can be expected to do is to give an accurate measurement of the surface water of each body of water in an image. The adjacency classifier helps to classify the various types of data that edges surround.
### TABLE 4.1. LAKE STATISTICS GENERATED BY THE ADJACENCY METHOD

<table>
<thead>
<tr>
<th>Field Number</th>
<th>Number of Field Elements</th>
<th>Average Data Values</th>
<th>Standard Deviation</th>
<th>Sum of Data Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25937</td>
<td>207</td>
<td>21.33</td>
<td>5392169</td>
</tr>
<tr>
<td>2</td>
<td>261</td>
<td>132</td>
<td>16.73</td>
<td>34710</td>
</tr>
<tr>
<td>3</td>
<td>28</td>
<td>125</td>
<td>12.80</td>
<td>3518</td>
</tr>
<tr>
<td>4</td>
<td>606</td>
<td>116</td>
<td>10.86</td>
<td>70572</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>176</td>
<td>17.49</td>
<td>1238</td>
</tr>
<tr>
<td>6</td>
<td>15</td>
<td>163</td>
<td>9.89</td>
<td>2449</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>145</td>
<td>1.00</td>
<td>725</td>
</tr>
<tr>
<td>8</td>
<td>237</td>
<td>120</td>
<td>15.13</td>
<td>28654</td>
</tr>
<tr>
<td>9</td>
<td>62</td>
<td>127</td>
<td>9.43</td>
<td>7887</td>
</tr>
<tr>
<td>10</td>
<td>1493</td>
<td>163</td>
<td>12.40</td>
<td>1471</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>112</td>
<td>13.78</td>
<td>168364</td>
</tr>
<tr>
<td>12</td>
<td>24</td>
<td>142</td>
<td>8.94</td>
<td>1138</td>
</tr>
<tr>
<td>13</td>
<td>6</td>
<td>177</td>
<td>18.68</td>
<td>4271</td>
</tr>
<tr>
<td>14</td>
<td>6</td>
<td>174</td>
<td>15.55</td>
<td>1048</td>
</tr>
<tr>
<td>15</td>
<td>8</td>
<td>187</td>
<td>2.44</td>
<td>1122</td>
</tr>
<tr>
<td>16</td>
<td>1045</td>
<td>186</td>
<td>15.49</td>
<td>1493</td>
</tr>
<tr>
<td>17</td>
<td>39</td>
<td>121</td>
<td>11.13</td>
<td>136871</td>
</tr>
<tr>
<td>18</td>
<td>6</td>
<td>146</td>
<td>10.72</td>
<td>5707</td>
</tr>
<tr>
<td>19</td>
<td>7</td>
<td>175</td>
<td>8.71</td>
<td>1051</td>
</tr>
<tr>
<td>20</td>
<td>7</td>
<td>146</td>
<td>13.89</td>
<td>1026</td>
</tr>
<tr>
<td>21</td>
<td>7</td>
<td>157</td>
<td>9.84</td>
<td>1101</td>
</tr>
<tr>
<td>22</td>
<td>20</td>
<td>150</td>
<td>10.29</td>
<td>3005</td>
</tr>
</tbody>
</table>
TABLE 4.2. LAKES CLASSIFIED WITH THE ADJACENCY CLASSIFIER.

<table>
<thead>
<tr>
<th>Class Number</th>
<th>Field Number</th>
<th>Composite Average Intensity Value</th>
<th>Composite Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>207.00</td>
<td>21.33</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>186.78</td>
<td>2.75</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>176.90</td>
<td>3.35</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>163.33</td>
<td>3.29</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>21</td>
<td>157.00</td>
<td>9.84</td>
</tr>
<tr>
<td>6</td>
<td>22</td>
<td>150.00</td>
<td>10.29</td>
</tr>
<tr>
<td>7</td>
<td>20</td>
<td>146.23</td>
<td>4.25</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>142.00</td>
<td>3.19</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>132.00</td>
<td>4.35</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>126.72</td>
<td>5.14</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>17</td>
<td>121.31</td>
<td>4.78</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>116.00</td>
<td>3.50</td>
</tr>
<tr>
<td>13</td>
<td>11</td>
<td>112.00</td>
<td>4.15</td>
</tr>
</tbody>
</table>
Figure 4.3 (a) Line-Printer Gray-Tone Map of Lakes and (b) Lakes Classified with the Adjacency Method. The scale is approximately 91 feet by 91 feet for each pixel. The numbers in (b) correspond to the field numbers in Table 4.1.
4.2 Detection of Edges in LANDSAT-1 Satellite Data Associated With Field Patterns in an Agricultural Area.

An example of the quality of edges detected between agricultural fields from LANDSAT-1 satellite data is presented in this section. A black and white photograph of a color infrared film taken on the same day as the satellite image is included. This aircraft image and the LANDSAT-1 image are illustrated in Figure 4.4. The lighter agricultural areas on the aircraft image are fallow (plowed) fields while the dark areas are corn. The aircraft image was taken from an altitude of 1,500 meters above the ground level with a Hasselblad camera by the Remote Sensing Institute at South Dakota State University under a United States Bureau of Reclamation contract, number 14-06-700-7466. The city in the lower left part of the image is Courtland, Kansas, which is located in the north-central part of Kansas.

LANDSAT-1 band 5 (0.6-0.7µm) data which were digitized with SADE, were used. This spectral region normally yields contrast between fallow fields and corn fields. Therefore, band 5 is attractive for the detection of edges between vegetated and non-vegetated fields.

The edges detected on these data of an agricultural region are shown in Figure 4.5b. The darker symbols used in this edge display correspond to larger gradient magnitudes. A large gradient implies that a major edge is present. A larger gradient threshold could be used to eliminate the lighter symbols or minor edges in this image. The edge values are coded with the gradient magnitudes located beneath them. The gradient threshold can be chosen in one of two methods, either while the program is detecting edges or when SYMOVER is used to display the
Figure 4.4 (a) LANDSAT-1 Band 5 Imagery of Agriculture and (b) Simultaneous Aircraft Imagery. In (a) dark tones in general are fallow fields while light tones are corn. The opposite is true in (b). The aircraft image is a black and white image produced from color infrared film. The scale is approximately one inch to the mile.
Figure 4.5 (a) Line-Printer Gray-Tone Map of Digitized LANDSAT-1 Imagery of Agriculture and (b) Edges Located in the Digitized Agricultural Image. Dark Tones in general are corn fields and light tones are fallow fields. The scale is approximately 91 feet by 91 feet for each pixel.
edges detected. For example, assigning blank symbols to the first ten symbols with SYMOVER is equivalent to a gradient threshold of nine. When the first 20 symbols are assigned blanks by SYMOVER the equivalent gradient threshold is 19.

4.3 Other Applications of Edge Detection

During the developmental phase of the edge detection algorithm, many applications were tried. The edge detection algorithm was developed to locate edges between fields in digitized S190A SKYLAB images and SKYLAB S192 multispectral digital data. The location of ground test sites are located easier after the edge detection algorithm delineated the field boundaries. The edges detected in SKYLAB S192 scanner data, digitized RB-57 thermal scanner images are in a paper by Moore et al (17). These applications of edge detection and an example of the adjacency classifier results applied to digitized RB-57 scanner data are included in another paper by Moore et al (18).

Results obtained from edge detection of digitized leaves which have diseased portions are illustrated by Wehde et al (19). Edge detection was used to locate edges between the leaf and the background, and edges at the diseased portions of leaves. If pursued, this technique could be modified to automatically and accurately determine the percent of diseased area of the leaves.
CHAPTER 5
SUMMARY AND RECOMMENDATIONS

5.1 Summary

It was stated in Chapter 1 that it is important to be able to detect edges in digital images because:

1. Regions of different spectral reflectance characteristics are separated by edges in imagery.
2. The shape, area or perimeter of the regions can be calculated if edge locations are known.
3. The statistics of the data in the regions can be used to train an M-Class classifier.
4. Several images of the same scene can be registered by superimposing edges detected in the images.

In this thesis two unique two-dimensional analysis techniques were developed and implemented with software. Results of the edge detection and adjacency classifier computer programs when applied to digitized LANDSAT-1 data are illustrated. Other applications of the two algorithms which are reported elsewhere are referenced. The accuracy of the edge detection algorithm was evaluated for surface water edges. The adjacency method can be used to determine several statistics of data regions that are surrounded by edges. These statistics were not used to train an M-class classifier, but they were used to define a unique adjacency classifier algorithm. Results of the use of the adjacency classifier for land-lake LANDSAT-1 satellite data were reported. No effort was applied to the registration of images by detecting and enhancing edges.
Edge detection was applied to the following data:

1. Digitized images of the earth taken from aircraft
2. Digitized images taken from SKYLAB
3. SKYLAB conical S192 multispectral scanner digital data
4. Digitized images of diseased leaves
5. Digitized LANDSAT-1 satellite images

Another useful computer program called SYMOVER, which produces a gray-tone line printer output, was developed and successfully used for displaying edges and the output of the adjacency classifier. SYMOVER produces dark symbols by overprinting characters.

In this thesis the successful use of three important characteristics of the gradient vector are incorporated into the software required to detect edges. The three important characteristics of the gradient vector are:

1. High gradient vector magnitudes are inherent at step edges between objects.
2. Low gradient vector magnitudes are inherent within homogeneous objects.
3. The gradient vector direction is perpendicular to the edge direction.

5.2 Recommendations

1. The variables that enter into the digitization process should be analyzed. The accuracy of the edge detection algorithm, when applied to ideal step edges, is a good starting point for this. The edge detection algorithm should be evaluated for
different point spread functions and noise distribution that are applied to the ideal data.

2. A method to fill gaps in edges which are obtained from the edge detection algorithm might be developed.

3. A procedure for automatic gradient threshold value assignment would be a useful addition to either the edge detection algorithm or the adjacency method.

4. The adjacency classifier extracts regions of data surrounded by "thick" high gradient edge regions. An ideal adjacency classifier computer program should have a single element edge between data classes similar to the edges detected with the edge detection algorithm. The advantages of both the adjacency classifier and edge detector should be combined in one computer program.

5. The utility of SYMOVER as an analysis tool rather than simply a display mechanism could be achieved with minor modifications.
   a. It would be an advantage to have a histogram of the number of times that each symbol is used in a display generated by SYMOVER.
   b. Another useful addition would be the capability to choose any rectangular set of data and display only the points within this set. (The rectangle might be oblique to the data line.) This would allow a user to obtain histograms of the data over only the specified rectangular area.
c. The symbols used by SYMOVER can be made from as many as five different simple characters. It would be possible with some modifications to read five different registered data sets and obtain a map of unions and intersections of the various data ranges in all five data sets.
APPENDIX A.

TWO EDGE DETECTION FORTRAN PROGRAMS
APPENDIX A.

TWO EDGE DETECTION FORTRAN PROGRAMS

The edge detection algorithm which is described in Chapter two and applied to real two-dimensional LANDSAT-1 satellite data in Chapter four, is implemented here as two FORTRAN computer programs. These computer programs are called Edge Detection Version II and Subroutine Edge. FORTRAN listings of both of these programs and the following questions are answered in this appendix:

1. Which program should be used for a given application?
2. Which data are acceptable inputs to these programs?
3. How are the programs used?
4. Which parameters are required to be chosen?
5. Which subroutines or functions are called by the programs?

A.1 Subroutine Edge

Subroutine Edge can be called from any FORTRAN computer program to detect edges in a matrix of two-dimensional data. The major difference between Subroutine Edge and Edge Detection Version II is the memory requirement that is necessary for the edge detection algorithm. The memory size in bytes required by Subroutine Edge is approximately equal to 24 times the number of data points. For example, a 128 x 128 matrix of data values requires approximately 400 kilobytes of memory to store enough information to locate edges. Edge Detection Version II needs only 7.0 kilobytes of memory to analyze a 128 x 128 matrix of data values. Subroutine Edge requires that the input data is a two-dimensional array which is created in the calling program.
Subroutine Edge is useful for the following applications:

1. Locating edges in small regions of data. (See TABLE A.1 for memory size requirements.)

2. Testing modifications to the basic edge detection algorithm.
   a. The code in Subroutine Edge is straightforward to read and easily modified.
   b. Arrays of gradient vector directions and magnitudes are available outputs of Subroutine Edge. (These matrices are helpful when the effects of modifications are evaluated.)

3. Subroutine Edge accepts a matrix of data values as input, while Edge Detection Version II is required to read input data from a tape.
   a. The data model proposed for a realistic edge is easily simulated and stored in matrix form.
   b. The point spread function, plus noise functions with various distributions can be used to convert ideal edges which are stored in matrix form into the proposed realistic data model. (See Section 2.1) Subroutine Edge can be used directly to detect edges in these data because they are stored in a matrix.

The call procedure for Subroutine Edge, the parameters in the call list, and the common area with input data, edges, gradient magnitude and gradient direction, are listed in Table A.1. Subroutine Edge and Function IDIRX(x,y), which computes the gradient vector direction, are listed in the next section.
TABLE A.1 USE OF SUBROUTINE EDGE

Call Procedure:

CALL EDGE (K,NR,LRBGR)

Parameter List:

K = the number of columns in the data matrix.
NR = the number of rows in the data matrix.
LRBGR = the gradient threshold

These statements are required in the calling program.

INTEGER*2 MA,MB,MC,MD

Matrices MA, MB, MC, and MD are filled with two byte words.

COMMON MA(256,128), MD(256,128), MB(256,128), MC(256,128)

MA(I,J) = input data matrix.
MB(I,J) = calculated gradient vector magnitudes matrix.
MC(I,J) = calculated gradient vector directions matrix.
MD(I,J) = detected edge matrix.

Data Memory Size Required:

Memory required = NR(K)(24) = (number of data points)(24)bytes

Example: NR = 100, K = 100

Memory required = 240 kilobytes

1 The method of choosing the gradient threshold is included with the description of Edge Detection Version II.
An example of a main program which reads data from tape, creates a matrix of data, calls Subroutine Edge and writes the edge output to another tape is included in Section D.1. Procedures to choose the gradient threshold parameter for edge detection are provided in Section A.5.

The only outputs to Subroutine Edge are three matrices which include the following information: the gradient vector directions, the gradient vector magnitudes, and a matrix with zeros stored at non-edge locations and gradient magnitudes at edge locations. Subroutine Edge only calculates values for these three matrices and returns them to the calling program. The program SYMOVER, which is described in Appendix C, can not directly convert the matrix of values located by Subroutine Edge into symbols for display. Data displayed with program SYMOVER is required to be read from a tape or disc.

A main program, which calls Subroutine Edge to locate edges, is included in Section D.1. This example program is handy because it allows the user to do almost everything with Subroutine Edge that can be done with Edge Detection Version II. (Memory size limitation remains the same) Data is read from tape or disc with this main program and an edge output tape is created, which can be displayed as a symbol map by application of program SYMOVER.

A.2 FORTRAN Listing of Subroutine Edge

Subroutine Edge is coded in FORTRAN and catalogued in a file maintained by the Remote Sensing Institute at South Dakota State University. The code was written in a very straightforward but inefficient manner. It is helpful to look at the edge detection flowchart, Figure 2.3,
when reading this code. The first section in this appendix explains several applications where Subroutine Edge is more suited for use than Edge Detection Version II.

************************************************************************************

SUBROUTINE EDGE

************************************************************************************

SUBROUTINE EDGE(K,NR,LRBGR)
INTEGER*2 MA,MD,MA,MC
COMMON MA(256,128),MD(256,128),MB(256,128),MC(256,128)
KD=K-1
N2=NR-1
K3=KD-1

DESCRIPTION OF THE THREL PARAMETERS
K IS THE NUMBER OF ELEMENTS IN A LINE OF DATA
NR IS THE NUMBER OF LINES OF DATA ANALYZED BY EDGE
LRBGR IS THE GRADIENT THRESHOLD

MA IS THE MATRIX OF DATA VALUES THAT ENTER SUBROUTINE EDGE
MB, MC, MD ARE MATRICES COMPUTED BY SUBROUTINE EDGE

MB IS A MATRIX OF GRADIENT MAGNITUDES
MD IS A MATIX OF EDGE VALUES
MC IS A MTRIX OF GRADIENT VECTOR DIRECTIONS

CALCULATE THE GRADIENT VECTOR DIRECTIONS
AND GRADIENT VECTOR MAGNITUDES FOR THREE LINES
OF DATA.

DO 11 J2=2,4
J=J2-1
DO 10 I=1,KD
I2=I+1
KY=MA(J,1)-MA(J2,12)
MX=MA(J2,1)-MA(J,12)
MB(J,1)=IABS(MX)+IABS(MY)
10 MC(J,1)=IDIRX(MX,MY)
11 CONTINUE

CALCULATE EDGES FOR ALL LINES OF DATA
DO 12 J2=5,N2
J3=J2-1
JPLUS=J2-2
J=J2-3
JMINS=J2-4
J8=J0+1
DO 25 I=2,K3
ICX=0
IMINS=I-1
MY=MA(J3,IMINS)-MA(J2,1)
MX=MA(J2,IMINS)-MA(J3,1)
MB(J3,IMINS)=IABS(MX)+IABS(MY)
25 CONTINUE

DETAILED DESCRIPTION OF THE CODE

K IS THE NUMBER OF ELEMENTS IN A LINE OF DATA
NR IS THE NUMBER OF LINES OF DATA ANALYZED BY EDGE
LRBGR IS THE GRADIENT THRESHOLD

MA IS THE MATRIX OF DATA VALUES THAT ENTER SUBROUTINE EDGE
MB, MC, MD ARE MATRICES COMPUTED BY SUBROUTINE EDGE

MB IS A MATRIX OF GRADIENT MAGNITUDES
MD IS A MATRIX OF EDGE VALUES
MC IS A MATRIX OF GRADIENT VECTOR DIRECTIONS

CALCULATE THE GRADIENT VECTOR DIRECTIONS
AND GRADIENT VECTOR MAGNITUDES FOR THREE LINES
OF DATA.
MC(J3,IMINS) = IDIRX(MX,MY)
IK6M=Mb(J,1)
IF(I chubby.LT.LRBGP)GO TO 25
IDX=MC(J,1)
IPLUS=I+1
GO TOU(3,1,2,3,24),1UX
GO TO 25
31 IRBl=Mb(JPLUS,1MINS)
IRB2=Mb(1MINS, IPLUS)
IF(IRBl.GT.IRBM.OR.IRB2.GT.IRBM)GO TO 28
GO TO 21
22 IRBl=Mb(J,1MINS)
IRB2=Mb(J,1PLUS)
IF(IRBl.GT.IKBM.OR.IRB2.GT.IKBM)GO TO 28
GO TO 21
23 IRBl=Mb(IMINS,1MINS)
IRB2=Mb(JPLUS, IPLUS)
IF(IRBl.GT.IKBM.OR.IRB2.GT.IKBM)GO TO 28
GO TO 21
24 IRBl=Mb(IMINS,1)
IRB2=Mb(JPLUS,1)
IF(IRBl.GT.IKBM.OR.IRB2.GT.IKBM)GO TO 28
21 ICX=IDX
28 CONTINUE
25 MD(IMINS,IMINS)=ICX
DO 26 1=K3,KD
1P=1+1
MY=MA(J3,1)-MA(J2,1P)
MX=MA(J2,1)-MA(J3,1P)
IMINS=IMINS+1
MB(J3,IMINS)=IABS(MX)+IABS(MY)
26 MC(J3,IMINS)=IDIRX(MX,MY)
12 CONTINUE
RETURN
END

FUNCTION IDIRX(MX,MY) CALCULATES GRADIENT VECTORS DIRECTIONS

FUNCTION IDIRX(MX,MY)
X=MX
Y=MY
IF(.NOT.(X.EQ.0.AND.Y.EQ.0))GO TO 5
IDIRX=0
RETURN
5 IF(X.EQ.0)X=.001
RAT=Y/X
IF(.NOT.(RAT.GT.-4.142.AND.RAT.LT.2.4142))GO TO 1
IDIRX=2
RETURN
1 IF(.NOT.(RAT.LT.-4.142.AND.RAT.GT.-2.4142))GO TO 2
IDIRX=4
RETURN
2 RAT=ABS(RAT)
IF(RAT.LT.2.4142)GO TO 3
IDIRX=3
RETURN
3 IDIRX=1
RETURN
END
A.3 Edge Detection Version II

Edge Detection Version II is a stand alone computer program written to detect edges within a packed data set. Data in a packed data set are assumed to have a range of values from zero to 255 and are stored in eight bit bytes a line at a time in a sequential data storage area, like a tape or disc. Edge Detection Version II does the following:

1. Reads subsets of a packed sequential data set from tape or disc. (The parameters which are chosen to do this are illustrated in Table A.2)
2. Detects edges within the data subset.
3. Saves the edge data on a tape in the correct format for display of edges with SYMOVER. (Zeros are stored at non-edges and gradient magnitudes at edges.)
4. Computes a histogram of the gradient magnitudes stored at edge locations. (If histogram option is specified)

The major differences between Subroutine Edge and Edge Detection Version II are presented first. Next, a brief description of the changes made in Edge Detection Version II, to reduce memory requirements, are presented. A summary of the following requirements of the program are illustrated in Table A.2 and briefly described in this section:

1. FORTRAN unit numbers used for input and output.
2. Parameters for parameter cards.
3. Subroutines and Functions called.
4. Formula to compute approximate memory size.
TABLE A.2 USE OF EDGE DETECTION VERSION II

Required Storage Devices: (Example JCL for tapes in Appendix D)

1. Input data read from FORTRAN unit number 14.
2. Output edge data saved on FORTRAN unit number 15.

Parameter Cards:

1. Card one: Title used at top of output FORMAT(20A4).

2. Card two: MXI,MXF,MYI,MYF,NFRQ,LRBGR FORMAT(6I5)

   MXI = starting byte number.
   MXF = stopping byte number.
   MYI = starting line number.
   MYF = stopping line number.
   NFRQ = histogram option.

1 and 2 ... Cumulative distribution graph plotted.
3 ... Cumulative and Normalized Distribution
   in Table form.
4 ... Both above histograms forms.

   LRBGR = gradient threshold

Subroutines and Functions called:

1. MASMOV, SYMFREQ: Catalogued in RONS.PLOTDEKS (S.D.S.U. computer
   center, Ron Greve's Subroutines)

2. HISPLT, Edge Detection Version II, FUNCTION IDIRX: Catalogued
   in SKYLBLIB (R.S.I. maintains these routines)

Data Memory Size Required:

   Memory = K(56)bytes = (number of elements in a line of data)(56)bytes
A listing of the computer program, which implements Edge Detection Version II, is included in the next section. Job Control cards for this program are listed in Appendix D along with a program which detects edges and displays them as symbol maps with program SYMOVER. Both SYMOVER and Edge Detection Version II are catalogued at the South Dakota State University Computer Center and maintained by the Remote Sensing Institute. A method is described in the last section of this appendix for choosing the best gradient threshold. This same gradient threshold is also the best choice of gradient threshold for the Adjacency Method.

The major difference in this program and Subroutine Edge is the reduced memory requirements necessary. The memory requirement, in bytes, is equal to 56 times the number of elements in one line of data. For example, only approximately seven kilobytes of memory are necessary to analyze a data matrix with 128 x 128 points or a data matrix with 128 x 2048 points. Subroutine Edge requires 100 kilobytes and 1.6 megabytes, respectively, for the above data matrices. This reduced memory requirement is a very practical benefit for an edge detector program.

Because of the huge reduction in required computer memory, Edge Detection Version II is useful for the following specific cases. Very large data sets can be handled by Edge Detection Version II. The major limitations are the width of the data set and the computer memory size. For example, given that the maximum memory size is 256 kilobytes, the width of the data set can be as large as 18,200 points. A minicomputer or microprocessor with a 15 kilobyte memory can detect edges in a data set which is 700 points wide. (This calculation allows five kilobytes for an assembly equivalent to this program.)
The following method was used to reduce memory requirements for Edge Detection Version II. Only three lines of gradient magnitudes and directions are necessary to calculate a line of edge values. Therefore, after one line of edges are calculated from three lines of gradient vector information, the first line of gradient information can be eliminated and a new line of gradient information is calculated to take its place. A line of edge values can now be detected in the present three lines of gradient information. This continues until all lines of data are analyzed. The implementation of these steps complicates the code but makes the edge detection algorithm useful for analysis of large matrices of data, or permits edge detection to be done with a minicomputer or microcomputer.

The edges detected by Edge Detection Version II are stored on a tape or disc with a packed format. Zeros are stored where there are no edges and gradient magnitudes are stored at edge locations. The edge data are stored in words that are a byte long and thus limited to a range of zero to 255. This format is compatible with the input tape format of program SYMOVER. The line-printer edge maps displayed in Figure 4.2 and 4.5 were created by SYMOVER.

Two forms of histograms of the gradient magnitudes which are located at edges, are available. A parameter, NFRQ, is available to choose between a histogram table, a graph of the histogram or no histogram at all. The histogram option is helpful when a symbol map of the edges stored on a tape are printed with SYMOVER. The histogram shows the entire range of gradient magnitudes that are located at edges.
A.4 FORTRAN Listing of Edge Detection Version II

A FORTRAN program called Edge Detection Version II is listed here. A summary of the requirements of this program are illustrated in Table A.2. Job control cards, JCL, are included in Appendix D. A computer program which detects edges from data stored on a tape, creates a tape of edge with gradients stored at edges, and prints a symbol map of the edges is also included in Appendix D. The choice of parameter cards for SYMOVER is presented in Appendix C.

Several Subroutines, which are used by Edge Detection Version II, are not listed here but are available at the South Dakota State University computer center. A short description of the purpose of these routines is included here:

1. MASMOV - Stored in RONS.PLOTDEKS, maintained at computer center by Ron Greve, Purpose: Assembly routine to move data from one vector to another.

2. SYMFRQ - Stored in RONS.PLOTDEKS, maintained at computer center by Ron Greve, Purpose: Save frequency of gradient magnitude values stored at edges.

3. HISPLT - Stored in SKYLBLIB, maintained by the Remote Sensing Institute, Purpose: Display gradient magnitude frequencies saved by SYMFRQ.
EDGE DETECTION VERSION II

DIMENSION IRB(2)
 INTEGER*4 LINEPK(512), IOFREQ(256), IGFREQ(256)
 INTEGER*4 ALABEL(20), MEDG4(512)
 INTEGER*2 M2
 LOGICAL*1 LINE,LIN3(2)
 LOGICAL*1 LIN(2048), LIN2(2048), MEDG1(2048)
 EQUIVALENCE (MEDGE(11), MEDG1(11), MEDG4(11))
 INTEGER*2 MLIN1(1024), MLIN2(1024), MLIN3(11)
 EQUIVALENCE (MLIN4(11), LIN3(11))
 DIMENSION NUM1(16), NUM2(16), NUM3(16)
 INTEGER*2 IDIR(1024,10), MRBGK(1024,10), MEDGE(1024)
 INTEGER*2 NSUM2(1024)
 INTEGER*2 NSUM1(1024)
 COMMON /KLENG/ INE/KCALL/ICALL
 COMMON /INPUT/LINE(2048)
 EQUIVALENCE (MLIN1(11), LIN(11))
 EQUIVALENCE (MLIN2(11), LIN2(11))
 EQUIVALENCE (LINE(11), LINEPK(11))
 DATA NUM1/1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1/
 DATA NUM2/2,3,4,5,1,2,3,4,5,1,2,3,4,5,1,2/
 DATA NUM3/3,4,5,1,2,3,4,5,1,2,3,4,5,1,2,3/
 DATA IUFREQ/256*0/, IGFREQ/256*0/
 ICR=11
 LP=12

C MAGIN IS THE INPUT DEViCE NUMBER = UNIT 14.
C MAGOU IS THE OUTPUT DEVICE NUMBER = UNIT 15.
MAGIN=14
MAGOU=15

C READ PARAMETER CARD
READ (ICK,1000)MXI,MXF,MYI,MYF,NFRQ,LRBGR
C
C........MXI........STARTING HORIZONTAL DATA POSITION NUMBER.
C........MXF........STOPPING HORIZONTAL DATA POSITION NUMBER.
C........MYI........STARTING VERTICAL LINE NUMBER.
C........MYF........STOPPING VERTICAL LINE NUMBER.
C........NFRQ.......CONTROL PARAM FOR THE HISTOGRAM OPTION.
C........1........AND 2 GRAPH OF CUMULATIVE DISTRIBUTION.
C........3........TABLE OR THE CUMULATIVE AND
C........NORMALIZED DISTRIBUTION FUNCTIONS.
C........4........BOTH ABOVE HISTOGRAMS
C........LIEBGR....GRADIENT THRESHOLD.
MXF=MXF/4
MXF=MXF+4
NR=MYF-MYI
GO TO(108), MYI
MYI=MYI-1
DO 100 II=1, MYI
100 READ(MAGIN,1001)
108 ILEN=MXF-MXI+1
KM=ILEN-1
KML=ILEN-2
NNN=31
ICALL=1
IY=MYI+2
MX=MXI+1
NS=MXI-1
ILEN2=ILEN*2  
K4=KM/4  
III=0  
DO 104 I=1, NR  
III=III+1  
READ(MAIN,1001)(LINEPK(J),J=1,MX4)  
call MASMOV(ILEN,LINEX(PX1),LINEX(1))  
LIN3(2)=LINEX(1)  
1  
RESUMESUMSPICTOTGETACROSSTHELINE.  
DO 51 IT2=4, ILEN2,2  
IT=IT2-2  
ITT=IT/2  
ITB=ITI+1  
M2=MLIN3(1)  
LIN3(2)=LINE(1ITB)  
51 NSUM2(ITT)=M2+MLIN3(1)  
GO TO(14,15,16,10,10,10,12,12,12,12,12,12,12,12,13), III  
13 III=8  
C  
SECTIONFORLOCATINGEDGES  
12 ITND=III-1  
11=NUM1(ITND)  
12=NUM2(ITND)  
13=NUM3(ITND)  
1YY=1YY+1  
C........CHECKEACHEXPONTFORLOCALMAXIMUMSACROSSTHELINE.  
DO 22 IT=2,KML  
ICX=0  
ITPLUS= IT+1  
ITMIN=IT-1  
ITX=ITIP(IT,12)  
IRAM=MRBGR2(IT,12)  
IF(IRBM.LT.LRBM)GO TO 25  
GO TO(24,28,26,23,24,28,26,23), ITX  
23 IRB1=IRBM-MRBGR2(ITPLUS,11)  
IRB2=IRBM-MRBGR2(ITMIN,13)  
IF(IRB1)22,41,41  
41 IF(IRB2)22,60,60  
60 CONTINUE  
GO TO 21  
24 IRB1=IRBM-MRBGR2(IT,11)  
IRB2=IRBM-MRBGR2(IT,13)  
IF(IRB1)22,43,43  
43 IF(IRB2)22,62,62  
62 CONTINUE  
GO TO 21  
28 IRB1=IRBM-MRBGR2(ITMIN,11)  
IRB2=IRBM-MRBGR2(ITMIN,13)  
IF(IRB1)22,45,45  
45 IF(IRB2)22,63,63  
63 CONTINUE  
GO TO 21  
26 IRB1=IRBM-MRBGR2(ITPLUS,12)  
IRB2=IRBM-MRBGR2(ITPLUS,12)  
IF(IRB1)22,47,47  
47 IF(IRB2)22,65,65  
64 CONTINUE  
21 ICX=IRBM  
IF(ICX.GT.255)ICX=255  
25 CONTINUE  
2 CONTINUE
22 MEDGE(ITMIN)=ICX
DO 73 IP=2,1LEN2,2
73 MEDGE(IP/2)=MEDGE(IP)
MX=1
NNN=31
CALL SYMFDRO(0,ITMIN,MEDGE(1),IGFREQ)
C
GRADIENT SECTION
10 IT=NUM1(II-3)
DO 52 IT=1,KM
IT2=IT+1
X=MLIN1(IT2)-MLIN2(IT)
Y=MLIN1(IT)-MLIN2(IT2)
IGTR(IT,11)=IDIRX(X,Y)
GGRAD=ABS(X)+ANS(Y)
52 MRBUR2(1T,11)=GGRAD
16 CALL MASMOV(ILEN2,LIN2(1),L1N(1))
15 DO 74 IT=1,ILEN
74 MLIN2(IT1)=(NSUM2(IT)+NSUM1(IT))/4
14 CALL MASMOV(ILEN2,NSUM2(1),NSUM1(1))
104 CONTINUE
WRITE(LP,1009)ITMIN,1YY
IGFREQ(1)=0
CALL HISPLOT(IGFREQ,ALABEL,NFRQ)
1000 FORMAT(6I5)
1001 FORMAT(200(12A4))
1002 FORMAT(' ','32I4')
1003 FORMAT(12A4)
1004 FORMAT('11,U1X,2CA4')
1005 FORMAT('15,5X,9ELEMENT '15,'LINE '15,73X,'ELEMENT '15,'LINE '15)
1006 FORMAT(200(10A1))
1009 FORMAT('14,'ELEMENTS USED, AND '14,' LINES')
END
FUNCTION IDIRX(X,Y)
IF(.NOT.(X.EQ.0.AND.Y.EQ.0))GO TO 5
IDIRX=0
RETURN
5 IF(X.EQ.0)X=.001
RAT=Y/X
IF(.NOT.(RAT.GT..4142.AND.RAT.LT.2.4142))GO TO 1
IDIRX=2
IF(Y.LT.0)IDIRX=6
RETURN
1 IF(.NOT.(RAT.LT.-.4142.AND.RAT.GT.-2.4142))GO TO 2
IGRX=4
IF(X.GT.0)IDIRX=8
RETURN
2 RAT=ABS(RAT)
IF(RAT.LT.2.4142)GO TO 3
IDIRX=3
IF(Y.LT.0)IDIRX=7
RETURN
3 IDIRX=1
IF(X.LT.0)IDIRX=5
RETURN
END
A.5 Choice of the Gradient Threshold

Noisy edges that are located by the edge detection algorithm can be eliminated by increasing the gradient threshold. Increasing the gradient threshold can be achieved with two methods:

1. Rerun the edge detection algorithm with a larger gradient threshold.
2. Use SYMOVER to assign blanks to all edges which are coded with gradient magnitudes that are less than the new larger gradient threshold. (Gradient magnitudes are stored at edge locations and zeros are stored elsewhere.) Symbols are assigned to edges coded with gradient magnitudes greater than the gradient threshold.

SYMOVER can be used to display several different gradient thresholds at the same time by assigning different symbols over the range of gradient magnitudes that are stored at edges. (Make sure zero gradient magnitudes in the edge data set are assigned blanks because these can not be edges.) Symbols which are assigned to lower gradient ranges, and which make the edge map look noisy are eliminated by assigning blanks on a future run of SYMOVER to eliminate these noisy edges.

The following procedure is suggested for choosing the gradient threshold. With this procedure the edge detection algorithm, Edge Detection Version II or the main program in Appendix D, which calls Subroutine Edge, is used only once with an arbitrary choice of the threshold. (A threshold of 5 will work for most data.) Larger gradient thresholds can be used when the user has a feel for the best threshold.
Program SYMOVER is used twice in this procedure. First the edge detection algorithm is applied to the data with a relatively small gradient threshold. Next SYMOVER is used as described above to display several different gradient thresholds. The final edge map is produced by using SYMOVER to assign blanks to noisy edges which are less than a new larger gradient threshold chosen from results displayed on the first pass.

1. Prepare parameters for Edge Detection.
   a. Specify input tape and output tape JCL.
   b. Choose subset of area to be analyzed.
   c. Choose correct histograming option. (Edge Detection Version II only.)
   d. Run Edge Detection Program.

2. Prepare general SYMOVER deck to display edge data stored on tape.
   a. Specify input tape.
   b. Choose a symbol sequence, which will be assigned to the numbers, so several different gradient thresholds are displayed. (Histogram output available from Edge Detection Version II is helpful because it calculates the distribution of the gradient magnitudes at edges.)

3. Use SYMOVER to simulate edge detection with a gradient threshold large enough to assign blanks to edges which are noisy.
APPENDIX B.

THE ADJACENCY METHOD

COMPUTER PROGRAM
APPENDIX B.
THE ADJACENCY METHOD COMPUTER PROGRAM

A computer program which implements the Adjacency Method, which is described in Chapter three, is described in this appendix. The Adjacency Method calculates statistics for fields. A field is a group of points that are adjacent and surrounded by an edge. The Adjacency Method is one of two programs in the Adjacency Classifier. This program is implemented in FORTRAN. The other program in the Adjacency Classifier is a statistical test which predicts whether fields located by the Adjacency Method are significantly different or similar. This statistical test was implemented with a program written for an HP-25 calculator. The test could easily be written in FORTRAN.

The limitations of the Adjacency Method are summarized in Table B.1 and briefly described in the next section. The necessary storage areas, the format of input tape, the parameters chosen, the subroutines called, and the outputs which are available are presented. A FORTRAN listing of the Adjacency Method is presented in Section B.2 and a brief description of the choice of the gradient threshold is presented in Section B.3. It is best to run Edge Detection Version II before the Adjacency Method. The use of program SYMOVER to display field data with a line printer is discussed in Appendix C. Example Job Control Language cards and computer run decks which start from data tapes and end with symbol maps with different characters assigned to different fields, are included in Appendix D.
TABLE B.1 LIMITATIONS OF THE ADJACENCY METHOD COMPUTER PROGRAM

Required Storage Devices: DD Necessary

1. Input data tape read by assembly subroutine REDAPE. (//GO.TAPIN)
2. Temporary field data disc storage. (//FT.08F001)
3. Final permanent field data tape. (//FT.10F001)

Parameter Card:

MXI, MXF, MYI, MYF, LRBGR, IAVE: FORMAT (615)

1. MXI, MXF, MYI, MYF are illustrated and described in Table A.1.
2. LRBGR is the gradient threshold. (Section B.3 describes how to choose the gradient threshold.)
3. IAVE allows the input data to be averaged over (IAVE*IAVE) neighborhoods to reduce high frequency noise.

Subroutines and Functions Called: (Briefly described in Section B.2)

Catalogued in RONS.PLOTDEKS (Ron Greve's Subroutine)

MASMOV, MODTAB, TRNTAB

Catalogued in SKYLBLIB (Remote Sensing Institute)

REDAPE, CLSAPE, Adjacency Method main program, IDIRX, TRANS.

Function IDIRX (X,Y) is listed in Section A.2.

Subroutine TRANS is listed with the Adjacency Method in Section B.2.
B.1 The Adjacency Method of Statistic Generation

The Adjacency Method is written in FORTRAN for an IBM 370/145 computer with an OS/VS operating system. The input data is assumed to be stored on a computer tape and limited to a data range between zero and 255. These data are stored in a packed format sequentially a line at a time, where the packed format implies that each data value is stored in a single eight bit byte.

A temporary data set on a disc is used to store temporary field and edge information. The final data storage area for the permanent fields is stored on a tape. These two data sets are stored in the same packed format that the input data is stored. Edge information is coded with values zero through five while fields are coded with values 6 to 255. The permanent field data set and a list of statistics for each permanent field are the only output products of this program. The statistics listed are labeled under the same field numbers that are used in the permanent field data. The number of field points, the average data values, the sum of the data values squared, and the standard deviation of the data are listed for all permanent fields.

Program SYMOVER (See Appendix C) can be used to convert the field numbers into symbols and display them with a line printer.

The parameter cards necessary for the adjacency method are illustrated in Table B.1. The parameters MXI, MXF, MYI, and MYI have the same use in this program as they do in Edge Detection Version II. They specify a data sub-area for analysis with the program. Not all sub-routines and functions used in the adjacency method are listed in this appendix. Brief descriptions are included in the next section.
B.2 A FORTRAN Listing of the Adjacency Method

A FORTRAN program called the Adjacency Method is listed here. A summary of the requirements of this program are illustrated in Table B.1. JCL (Job Control Language) cards and a run deck are included in Appendix D. Program SYMOVER can convert the field numbers stored on a tape into a symbol map. The choice of parameter cards which produces the map of field data illustrated in Figure 4.3 is included in a run deck in appendix D. This deck combines the Adjacency Method and SYMOVER.

Several subroutines and functions are used by the Adjacency Method and short descriptions of these routines are included here. They are available at the South Dakota State University Computer Center or Remote Sensing Institute.

1. REDAPE - Stored in SKYLBLIB, maintained by the Remote Sensing Institute. Purpose: Assembly routine to read a line of packed data and move the data to an INTEGER*2 vector.

2. CLSAPE - Entry in REDAPE. Purpose: Closes data file opened in REDAPE.

3. TRANS - Listed in this section, stored in SKYLBLIB. Purpose: Converts temporary field data set to permanent field data tape.


5. MASMOV - Stored in RONS PLOTDEKS, maintained at the computer center by Ron Greve. Purpose: Assembly routine to move data from one place to another.

6. MODTAB - Stored in RONS.PLOTDEKS, maintained at the computer center by Ron Greve. Purpose: Creates a translation table from an adjacent field table. (This table is used to convert temporary field numbers into permanent field numbers.)
7. TRNTAB - Stored in RONS.PLOTDEKS, maintained at the computer center by Ron Greve. Purpose: Translate temporary field numbers into permanent field numbers with the assembly translate command.

C**********************************************************************
C          PROGRAM NAME.........ADJACENCY CLASSIFIER
C       OCTOBER 2, 1975
C       WRITTEN BY MIKE RUSSL
C**********************************************************************
C
C CONTROL SECTION
DIMENSION MAJ(512)
DIMENSION LINSUM(1024,3)
INTEGER LS(256)
INTEGER*4 NUM, SUM, SUM2
INTEGER LINE, LINI(1024), LA, DEC, DECL(1024), LINDUP(1024)
LOGICAL LC, LA(512)
COMMON NUM(256,2), SUM(256,2), SUM2(256,2), IBG, ICO, IL, LA(2048)
COMMON DEC(1024), LC(256)
COMMON /INPUT/LINE(1024)
EQUIVALENCE (DECL(1), LC(1))
EQUIVALENCE (LA(1), A(1))
EQUIVALENCE (LINDUP(1), LINE(1))
C
C INITIALIZE CONSTANTS AND SET LIMITS
C**********************************************************************
C
C OFFSET INTO TRANSLATION TABLE
ITAB=0
C
ICD IS THE STARTING CLASS
CLASSES 1-4 ARE RESERVED FOR EDGES
CLASS 5 IS RESERVED FOR INSIGNIFICANT CLASSES
ICD=5
IBG=ICO

ICD & IDO ARE VARIABLES TO CONTROL SEARCHING TECHNIQUE(IFLG)
IDO=1
IDO=2

ICR=NUMBER OF CARD READER
LP = NUMBER OF WRITER
MDIN=NUMBER OF TEMPORARY FIELD NUMBER STORAGE AREA

MODOUT=NUMBER OF PERMANENT FIELD NUMBER STORAGE AREA.
ICR=5
LP=6
MDIN=8
MODOUT=10
C
IERR=1 IMPLIES TRANSLATION TABLE IS FULL AFTER TRANSLATION
IERR=2
C
 INITIALIZE TRANSLATION
LA(1)=1
LA(2)=2
LA(3)=3
LA(4)=4
LA(5)=5
1004 FORMAT('U', LINE='/3214)
1005 FORMAT('U', IFLG='110', NUTS='110)
C READ INITIAL DATA BLOCK LIMITS (615)
READ (ICR,1000)MXI,MXF,MYI,MYF,LRBGR,IAVE
C MXI,MXF IS THE LOCATION OF THE INITIAL POINT
C MYI,MYF IS THE LOCATION OF THE FINAL POINT
C LRBGR IS THE GRADIENT THRESHOLD
C NREC IS THE NUMBER OF ROWS MINUS ONE FROM THE DATA
C IAVE IS THE NUMBER OF DATA LINES AVERAGED
NREC=MYF-MYI
DO 200 I=1,256
NUM(I,1)=0
SUM(I,1)=0
200 SUM2(I,1)=0
C
C START THE LOOP OF READING DATA
C******************************************************************************
C SKIP MYI DATA RECORDS AND READ ONE RECORD INTO COMMON/INPUT/
DO 100 I=1,MYI
100 CALL READEP(MXI,MXF,KM)
K=KM
MZ=2*K
IELEM=IAVE+IAVE
II=0
C... AVERAGE IAVE BY IAVE NEIGHBORHOODS FOR FIRST LINE......................
GO TO(52),IAVE
IIAV=IAVE-1
K=KM-IAVE
MZ=2*K
DO 51 I=1,IAVE
DO 50 J=1,K
JMAX=J+IIAV
DO 50 KK=J,JMAX
50 LINSUM(J,I)=LINSUM(J,I)+LINE(KK)
51 CALL READEP(MXI,MXF,KM)
DO 55 J=1,K
SUS=0
DO 54 I=1,IAVE
54 SUS=SUS+LINSUM(J,I)
LIni(J)=SUS/IELEM
55 CONTINUE
GO TO 53
52 CALL MASMV(MZ,LINDUP,LIN1)
53 KD=K-1
KP=K-1
NZ=2*KD
II=0
C... WHEN DONE WITH DATA-------------------------------------------------------
DO 101 III=IAVE,NREC,IAVE
DO 101 KKB=1,IAVE
II=II+1
CALL READEP(MXI,MXF,KM)
C... AVERAGE NEXT LINE--------------------------------------------------------
C... OR BRANCH AROUND LINE AVERAGE SECTION
GO TO(773),IAVE
DO 57 JJ=1,K
LINSUM(JJ,KKB)=0
JMAX=JJ+IIAV
DO 58 KK=JJ,JMAX
58 LINSUM(JJ,KKB)=LINSUM(JJ,KKB)+LINE(KK)
SUS=0
DO 59 KB=1,IAVE
59 SUS=SUS+LINSUM(JJ,KB)
57 LINE(JJ)=SUS/IIELEM
773 IFLG=ID
 NADJ=4
 IFL= 1
1GNNUM=2
C*** ...PROCESS A LINE OF DATA FOR ADJACENT CLASSES..................
 DO 102 J=1,KD
  J2=J+1
  N=LINE(J)
  Y=LINE(J2)
  X=LINE(J2)-LINE(J)
 C*** ...APPROXIMATE MAGNITUDE OF GRADIENT VECTOR .....................
  Z=(X*X+Y*Y)**.5
  IRBGRI=Z
 C*** ...COMPARE GRADIENT MAGNITUDE TO THRESHOLD ......................
  IF(IRBGRI=LRBGRI)110,111,111
 C*** ...IF GRADIENT IS LESS THAN THRESHOLD .......................110
  GO TO(113,114,115,115),IFLG
 C*** ...SEARCH POSITION A AND C FOR FIELDS A, B, C.................
 C*** ...FIELD NUMBERS ADJACENT TO B ................................
 C  SECTION IFLG=3 OR 4
115 IB=DEC(J)
116 C*** ...CHECK POSITION C FOR INSIGNIFICANT FIELD NUMBERS ...........
  GO TO(509,509,509,509,509,509,509),IB
  YES THERE IS AN ADJ CLASS (CHECK IF SAME AS BEFORE)
  GO TO(506),IFL
  C  CONTINUE (NO NEW ADJ)
   GO TO 114
 516 GO TO(519,519,519,519,519,519,519),NADJ
  GO TO 114
 519 NADJ=5
   GO TO 114
 C*** A NEW ADJ CLASS
 C*** ...RECORD A NEW ADJACENT FIELD NUMBER..........................
 506 GO TO(517,517,517,517,517,517,517),NADJ
  NADJ=NADJ+1
   GO TO 518
 517 NADJ=6
 518 IFL=2
   MADJ(NADJ)=IB
   GO TO 114
 C*** ...RESET ADJCLASS IFL=1
 509 IFL=1
 C  SECTION 2
C*** ...START COMPILING TEMPORARY STATISTICS........................
114 GO TO(510),1GNNUM
113 NUMER=N
   NUMER=1
   NUMER2=N*N
 1GNNUM=1
   JSL=J
   GO TO 102
 C*** ...CONTINUE COMPILING TEMPORARY STATISTICS.......................
 510 SUMER=SUMER+N
   NUMER=NUMER+1
   SUMER2=SUMER2+N*N


GO TO 102
C... ...IF GRADIENT IS GREATER THAN OR EQUAL TO THE THRESHOLD...........
C... ...CHECK IF PREVIOUS POINT A WAS A FIELD ..........................
111  GO TO(511,514),IGRNUM
     GO TO 102
C... ...YES PREVIOUS POINT WAS A FIELD ..............................
511  K8=0
     LOW=256
     LREAL=256
     GO TO(221,221,221,221,131),NADJ
     GO TO 223
221  GO TO(222,222),NUMER
     GO TO 131
222  LREAL=5
     GO TO 144
223  DO 502 I=W=6,NADJ
515  L=MADJ(1W)
C... ...COMPARE LOWEST ADJACENT FIELD NUMBER..........................
504  IF(L<LOW)150,503,503
150  K8=K8+1
     LOW=L
     LS(K8)=LOW
     L=LAI(LOW)
     GO TO504
503  IF(L<LOW.AT.LREAL)LREAL=LOW
     LOW=256
502  CONTINUE
C... ...UPDATE ADJACENT FIELD (TRANSLATION) TABLE WITH LOWEST ......
C... ...ADJACENT FIELD.................................................
DO 128  K9=1,K8
     KI=LS(K9)
128  LAI(KI)=LREAL
     GO TO 512
131  ICD=ICD+1
132  LREAL=ICD
C... ...COMPARE MAXIMUM FIELD NUMBER TO 256...........................
IF(ICD.NE.256)GO TO 144
     MLAS=J-1
     CALL TRANSFERR(K,MLAS,MDIN,MDOUT)
     GO TO 141
144  CONTINUE
     LAI(LREAL)=LREAL
512  JEND=J-1
C... ...CODE FIELD STRINGS IN THE DATA..............................
DO 129  K9=JSL,JEND
129  DEC(K9)=LREAL
     NADJ=4
     IFL=1
     IGRNUM=2
     NUM(LREAL,1)=NUM(LREAL,1)+NUMER
     SUM(LREAL,1)=SUM(LREAL,1)+SUMER
     SUM2(LREAL,1)=SUM2(LREAL,1)+SUMER2
     514  DEC(J)=IU1RX(X,Y)
102  IFLG=1DD
C... ...NO PREVIOUS POINT A WAS NOT A FIELD...........................
C... ...A CLASS AT LINE END........................................
GO TO(511),IGRNUM
     ID=3
     IDD=4
     CALL MASMOV(MZ,LINE,LINI)
CALL MASM0V(NZ,DEC,DEC1)
WRITE(MUN,1010)(DEC1(I),I=1,KD)
101 CONTINUE
MLAS=J-1
C... ...GO TO TRANSLATION SUBROUTINE.................................
CALL TRANS(TERR,K,MLAS,MIN,MOOUT)
141 WRITE(LP,1001)
   DO 130 I=6,ICO
   NUM=NUM(I,1)
   IF(NUM.EQ.0)GO TO 1777
   AVE=SUM(I,1)/NUM
   D(1)=1
   D(2)=2
   D(3)=3
   D(4)=4
   D(5)=5
   RWIND MDIN
   ICR=5
   LP=6
   WRITE(LP,1006)ICO,II,MLAS
   NGRT=4
1006 FORMAT('O',ICO,'I10, 'LINE',I5,' ELEMENT',I5)
   WRITE(LP,1004)(LA(I),I=5,ICO)
   1004 FORMAT('O',ICO,' TABLE LA(I) BEFORE TRANSLATION'/('O',32I4))
   IUFFST=IBG-1
   DO 256 I=5,ICO
   NUM(I,2)=0
   SUM(I,2)=0
   256 SUM2(I,2)=0
   IFX=ICO-1
C... ...REDUCE OVERALL ADJACENCY TABLE TO LOWEST ADJACENT FIELDS....
   DO 250 I=6,IFX
   M=I
   301 L=LA(M)
   IF(L.EQ.M)GO TO 300
   M=LA(L)
   GO TO 301
   300 IF(L-NGRT)251,251,252
   251 MBG=D(LA(L))
   GO TO 240
   252 IBG=IBG+1
   NGRT=L
   MBG=IBG
C... ...REDUCE TEMPORARY STATISTICS INTO..............................
C... ...PERMANENT ADJACENT FIELD STATISTICS............................
   240 NUM(MBG,2)=NUM(MBG,2)+NUM(I,1)
   SUM(MBG,2)=SUM(MBG,2)+SUM(I,1)
   SUM2(MBG,2)=SUM2(MBG,2)+SUM2(I,1)
   NUM(I,1)=0
   SUM(I,1)=0
   SUM2(I,1)=0
   D(I)=MBG
   250 CONTINUE
   IEND=II-1
   DO 257 I=5,IBG
   NUM(I,1)=NUM(I,2)
   SUM(I,1)=SUM(I,2)
   257 SUM2(I,1)=SUM2(I,2)
   CALL MASM0V(MBG,IBG,LA)
   UD 254 I=2,512,2
FUNCTION FOR COMPUTATION OF DIRECTION OF GRADIENT VECTOR

FUNCTION IDIRX(X,Y)
  IF (.NOT. (X.EQ.0 .AND. Y.EQ.0)) GO TO 5
  IDIRX=0
  RETURN
  5   IF (X.LT.0) IDIRX=1
      IF (Y.LT.0) IDIRX=4
      IF (Y.GT.0) IDIRX=3
      RETURN
  1   IF (.NOT. (RAT.LT.-.4142 .AND. RAT.GT.2.4142)) GO TO 1
      IDIRX=2
      RETURN
  2   RAT=ABS(RAT)
      IDIRX=3
      IF (RAT.LT.2.4142) GO TO 3
      RETURN
  3   IDIRX=1
      RETURN
END

TRANSLATION ROUTINE

SUBROUTINE TRANS(TERR,K,MLAS,MMIN,MMOUT)

INTEGER*4 NUM,SUM,SUM2
INTEGER*2 DEC(1024),DEC,LA
COMMON NUM(256,2),SUM(256,2),SUM2(256,2),TERR,ICD,LI,LA(2048)
COMMON DEC(1024),LC(256)
LOGICAL*1 C(2048)
EQUIVALENCE (C(1),DEC(1))
B.3 The Choice of the Gradient Threshold

The "best" gradient threshold for use with the Adjacency Method is the same gradient threshold value chosen to display edge data produced by the Edge Detection Algorithm. This gradient threshold eliminates noise and minor edges. See Section A.5 for an explanation of how the gradient threshold is chosen for edge detection. It works best to run edge detection on a small subset of the entire data first to compute the best gradient threshold.
APPENDIX C.

DISPLAY OF DIGITAL DATA WITH A LINE PRINTER
APPENDIX C.
DISPLAY OF DIGITAL DATA WITH A LINE PRINTER

A computer program called SYMOVER, which displays digital data as symbols with a line printer, is described in this appendix. SYMOVER's purpose is to read arrays of numbers, which are limited to a range of zero to 255, convert the numbers to symbols, and print them with a line printer. The symbol set that can be printed on the line printer can be either single character symbols or up to five overprinted character symbols. SYMOVER can be used to display the following arrays of numbers as symbol maps: Arrays of digital data, edges located in digital data, and fields located in digital data with the Adjacency Method. Sample symbol maps for these uses are included in Chapter four.

The limitations of program SYMOVER are explained in the next section. Formats for the parameter cards available with SYMOVER are presented in Section C.2. Job Control Language cards are included in Appendix D. A run deck for SYMOVER, when used in conjunction with Edge Detection or the Adjacency Method computer programs is also included.

C.1 Limitations of Program SYMOVER

SYMOVER converts digital data arrays to symbols with the IBM 370 assembly translate command. The one-to-one correspondence of data values and symbols is stored in a table called a translation table. If data values zero through 100 are to be converted into the symbol A, the first 101 elements in the translation table are A's. When over-striked symbols are used, a translation table is required for each
strike of the printer. A five overstrike symbol requires five translation tables. Translation tables for a four overstrike symbol sequence are shown in Table C.1. The symbol sequence in Table C.1 is defined easily with a single symbol parameter card type 2 for each of the eight symbols.

All of the parameter cards that are used with SYMOVER are described at the end of this Appendix. The following options are all built into SYMOVER.

1. Subareas of a large data set can be printed.
2. The data printed can be enlarged or reduced in either horizontal or vertical directions. It is possible to enlarge the image in the vertical direction and reduce the scale in the horizontal direction at the same time.
3. The mapping of a data value into symbols is defined with translation tables.
   a. Translation tables are specified with symbol parameter cards Type 1 or Type 2. Type 1 works the best for a display of the adjacency methods. While Type 2 works the best for overstrike displays.
   b. Translation tables which are used many times, can be stored and recalled from DISC storage.
   c. Translation tables can be recalled from DISC and modified.
4. Overprinting characters for symbols produces gray-tone maps which cost no more than single character symbol maps.
### TABLE C.1 A FOUR CHARACTER OVERSTRIKE SEQUENCE

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>DATA VALUES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bk*1</td>
<td>0</td>
</tr>
<tr>
<td>-</td>
<td>.</td>
</tr>
<tr>
<td>-</td>
<td>.</td>
</tr>
<tr>
<td>Bk</td>
<td>Bk</td>
</tr>
<tr>
<td></td>
<td>15</td>
</tr>
<tr>
<td>-</td>
<td>.</td>
</tr>
<tr>
<td>-</td>
<td>.</td>
</tr>
<tr>
<td>-</td>
<td>.</td>
</tr>
<tr>
<td>Bk</td>
<td>Bk</td>
</tr>
<tr>
<td></td>
<td>30</td>
</tr>
<tr>
<td>+</td>
<td>.</td>
</tr>
<tr>
<td>+</td>
<td>.</td>
</tr>
<tr>
<td>+</td>
<td>.</td>
</tr>
<tr>
<td>+</td>
<td>.</td>
</tr>
<tr>
<td>+</td>
<td>.</td>
</tr>
<tr>
<td>+</td>
<td>.</td>
</tr>
<tr>
<td>H</td>
<td>.</td>
</tr>
<tr>
<td>E</td>
<td>.</td>
</tr>
<tr>
<td>+</td>
<td>.</td>
</tr>
<tr>
<td>H</td>
<td>.</td>
</tr>
<tr>
<td>$</td>
<td>.</td>
</tr>
<tr>
<td>H</td>
<td>.</td>
</tr>
<tr>
<td>H</td>
<td>.</td>
</tr>
<tr>
<td>H</td>
<td>.</td>
</tr>
<tr>
<td>H</td>
<td>.</td>
</tr>
<tr>
<td>H</td>
<td>.</td>
</tr>
<tr>
<td>H</td>
<td>.</td>
</tr>
<tr>
<td>H</td>
<td>.</td>
</tr>
</tbody>
</table>

*Bk*1 is a blank character
There is a limit to the number of symbols that are printed across one page by a line printer. The limit assumed in the SYMOVER program is 130. SYMOVER automatically opens enough data classes for the entire width of the data printed. A data class is used for each group of 130 characters across the width of the data. The width of a data set is limited to 130 times the number of data sets the user specifies in the Job Control Statements. This is shown in the list of JCL for the catalogued version of SYMOVER. For example, ten SYSOUT classes implies that the width of the data set is a maximum of 1300 characters.

SYM OVER processes a line at a time for symbol output. After a line of data values is read, SYMOVER uses translation tables to translate the data into symbols. A two-character symbol implies that two translation tables are used to translate the data values into two strings of characters. The line of data which is printed is divided into 130 element character groups and stored in as many output classes as necessary. The algorithm then goes to the next line. After all data is converted, the output classes are printed on separate sheets of paper.

The output of the permanent field data and edge regions is efficiently displayed with SYMOVER using SYMBOL parameter card Type one to specify different symbols for different field numbers in the data. (See Section D.6 for a run deck.) In these data, values from zero to six are used to code edges, and values from 7 to 255 are used to code field numbers. By assigning symbols to data values zero to
six and blank characters to the rest of the data, only the edge information is printed. If data values zero to six are assigned blanks and the rest of the data values are assigned individual symbols, only field numbers are translated into symbols. This allows the user to locate the position of fields on a symbol map and to know the statistics of each field. Because there are only 49 characters, the characters must be reused to cover the entire field range of 7 to 255. The correct field number for symbols printed on the map is easily found because of the order in which field numbers are assigned. Field numbers are assigned line-by-line from the upper left area of the data to the lower right area of the data. If the user of this algorithm transverses the output symbol map in this same manner assigning the correct field numbers in the same order that the symbols were assigned, fields are located by number. This procedure was used to label the correct field numbers on Figure 4.3b. The main program of SYMOVER is written in FORTRAN. Many of the subroutines that SYMOVER calls were written in IBM/370 assembly language by Ron Greve of the S.D.S.U. Computer Center. These subroutines were written in September 1974 for an IBM-370-H145 computer with an OS-VS1 operating system. They are documented at the S.D.S.U. Computer Center. The following Subroutines are used:

1. MUPR1005--OPNTR, PRNTR, CPRNTR
2. STTR1005--TRNTAB, MODTAB, PCMTAB, MORTAB, LODTAB, WRITAB
3. ERAN1005--TRNERR, PRTERR, TBPERR
C.2 Use of SYMOVER Parameter Cards

The parameter cards which control SYMOVER are described in this section. There are five types of parameter cards that can be chosen when using SYMOVER. The order of usage of the parameter cards is illustrated in Figure C.1. SYMOVER parameter card one and card two are only used once. They are required to be the first two cards in the deck of parameter cards. SYMOVER parameter card three can be used more than once. Its purpose is to specify the translation tables which are used to define the one-to-one mappings of numbers into symbols. New translation tables are specified with SYMOVER symbol parameter cards type one or type two. Each group of symbol parameter cards is preceded by a SYMOVER parameter card three.

C.2a SYMOVER Parameter Card One

SYMOVER parameter card one is always necessary. This card specifies the area of the data set which is printed, and the scale factors which are applied to the data. Parameters MXI and MXF are the first and last element from the lines of data analyzed by SYMOVER. Parameters MYI and MYF are the first and last lines of data that are analyzed. Parameters IY and IX are scale factors that are applied to the data. A negative scale factor implies that the printed output is reduced in size by the absolute value of the scale factor. A zero or blank scale factor implies that the scale is not changed. A positive scale factor implies that the printed output is enlarged. The coding used for SYMOVER parameter card one is illustrated in Table C.2.
Figure C.1 Example of SYMOVER Parameter Deck
### TABLE C.2 SYMOVER PARAMETER CARD ONE

<table>
<thead>
<tr>
<th>CARD COLUMNS</th>
<th>PARAMETER NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 5</td>
<td>MXI</td>
<td>The position in a data line of the first symbol printed.</td>
</tr>
<tr>
<td>6 - 10</td>
<td>MXF</td>
<td>The position in a data line of the last symbol printed.</td>
</tr>
<tr>
<td>11 - 15</td>
<td>MYI</td>
<td>The first data line which is printed.</td>
</tr>
<tr>
<td>16 - 20</td>
<td>MYF</td>
<td>The last data line which is printed.</td>
</tr>
<tr>
<td>21 - 25</td>
<td>IX</td>
<td>The scale factor applied to data elements within data lines.</td>
</tr>
<tr>
<td>26 - 30</td>
<td>IY</td>
<td>The scale factor applied to lines of data.*</td>
</tr>
</tbody>
</table>

**EXAMPLE TAPE DATA SET**

<table>
<thead>
<tr>
<th>IY</th>
<th>MXI</th>
<th>MXF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**SCALE FACTOR EXAMPLES:**

1. ) IY = -3  Every third line is printed.
2. ) IY = 5  Every line is printed five times.
3. ) IX = 0, 1 or -1  Every element in a line is printed only one time.
4. ) IX = 2  Every element in a line is printed two times.

A negative scale factor implies that the symbol map is reduced in size. A zero scale factor is a default and implies that the map's scale is not changed. A positive scale factor implies that the symbol map is enlarged.
C.2b SYMOVER Parameter Card Two

SYMOVER parameter card two is the second parameter card read by SYMOVER. The number of characters necessary per symbol are specified on this card. A different translation table is necessary for each character that must be overstruck. For example, to print a map which has a maximum of four overstruck symbols four translation tables are necessary. Table C.1 illustrates a four character overstrike sequence and the four translation tables which are necessary. The coding used for SYMOVER parameter card two is illustrated in Table C.3.

<table>
<thead>
<tr>
<th>CARD</th>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLUMNS</td>
<td>NAME</td>
<td></td>
</tr>
<tr>
<td>1 - 5</td>
<td>NUMSTK</td>
<td>The number of overstrikes per symbol printed on the line printer.</td>
</tr>
<tr>
<td>6 - 10</td>
<td>NUMCTB</td>
<td>The number of new translation tables created by this job step.</td>
</tr>
</tbody>
</table>

C.2c SYMOVER Parameter Card Three

Translation tables, which define the one-to-one correspondence between numbers and symbols printed by SYMOVER, can be formed in five different ways. SYMOVER parameter card three specifies how a translation table is formed with parameter NDISP, the disposition of the translation table. The functions of parameter NDISP are outlined in Table C.5. Translation tables that are created during one run of SYMOVER can be saved for further use with the appropriate choice of parameter NDISP.
With NDISP equal to zero an old translation table is recalled from the disc and used. Modifications to old tables can be made by setting NDISP to two, or three. The modifications are specified with symbol parameter cards. The parameter NTYPE specifies the symbol parameter card type which is used to define changes to a translation table. The coding used on SYMOVER parameter card three is illustrated in Table C.4.

<table>
<thead>
<tr>
<th>CARD COLUMNS</th>
<th>PARAMETER NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 8</td>
<td>NAME</td>
<td>The name of the translation table. (No blanks or special characters at beginning or between characters of the name.)</td>
</tr>
<tr>
<td>11 - 15</td>
<td>NDISP</td>
<td>The disposition of the translation table*</td>
</tr>
<tr>
<td>16 - 20</td>
<td>NTYPE</td>
<td>The method used to create a new table. If NTYPE is zero, one table is created using symbol parameter card one. If NTYPE is greater than zero, NTYPE tables are created using symbol parameter card two. (MAX NTYPE=5).</td>
</tr>
<tr>
<td>21 - 25</td>
<td>ICARD</td>
<td>The number of symbol parameter cards for the table (S). Created with this parameter card three.</td>
</tr>
</tbody>
</table>

* NDISP can be any integer 0, 1, 2, 3, or 4.
<table>
<thead>
<tr>
<th>CARD COLUMNS</th>
<th>SYMBOL PARAMETER CARD</th>
<th>OLD TRANSLATION TABLE</th>
<th>STORE TRANSLATION TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NDISP = 0</td>
<td>NONE</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>NDISP = 1</td>
<td>YES. Define symbols for full 0-255 # Range.</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>NDISP = 2</td>
<td>YES. Define for Modified # Range.</td>
<td>IF YES</td>
<td>Replace old table</td>
</tr>
<tr>
<td>NDISP = 3</td>
<td>YES. Define symbols for full 0-255 # Range.</td>
<td>IF YES</td>
<td>Store new table</td>
</tr>
<tr>
<td>NDISP = 4</td>
<td>YES. Define symbols for full 0-255 # Range</td>
<td>NO</td>
<td>Store new table</td>
</tr>
</tbody>
</table>
C.2d SYMOVER Symbol Parameter Card Type One

New translation tables are created or old translation tables are modified with the Symbol parameter cards that follow parameter card three. Symbol parameter card type one is very effective in defining the one-to-one correspondence of numbers into symbols when a lot of symbols are involved. Usage of this parameter card in a SYMOVER parameter card deck is illustrated in Figure 3. The example translation table created with this parameter card deck was used in the adjacency classifier example in Chapter 4 to convert field numbers into symbols and print them. Symbols specified with this card format are required to be assigned for each position in the translation table. A maximum of 70 characters can be specified on one symbol parameter card type one.

<table>
<thead>
<tr>
<th>CARD COLUMNS</th>
<th>PARAMETER NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 5</td>
<td>IOFFST</td>
<td>The first character specified on this card (card column 11) is equivalent to the number IOFFST.</td>
</tr>
<tr>
<td>6 - 10</td>
<td>INUMB</td>
<td>The number of consecutive characters after IOFFST which are defined on this card.</td>
</tr>
<tr>
<td>11 - 80</td>
<td>CHA</td>
<td>The character string defined with this parameter card for all numbers from IOFFST to IOFFST + INUMB - 1.</td>
</tr>
</tbody>
</table>

Example:

IF IOFFST = 10, INUMB = 3 and CHA = ABC

Then data numbers are converted as follows:

10→ A, 11→ B, and 12→ C.
C.2e **SYMOVER Symbol Parameter Card Type Two**

This symbol parameter card is very effective for defining translation tables which produce overstrike character sequences. An overstrike character sequence is shown in Table C.1. A maximum of four characters overstruck for a dark symbol requires four translation tables. A SYMOVER parameter card deck which uses SYMOVER symbol parameter card type two to define these four translation tables is shown in Figure C.2. There are nine different multicharacter symbols specified here. This requires nine type two symbol parameter cards. Characters for all four translation tables are specified on each of the nine necessary symbol parameter cards. A maximum of five characters can be printed. The coding used for this parameter card is illustrated in Table C.7.
### TABLE C.7 SYMOVER SYMBOL PARAMETER CARD TYPE TWO

<table>
<thead>
<tr>
<th>CARD COLUMNS</th>
<th>PARAMETER NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 8</td>
<td>NAME (1)</td>
<td>The name of the first translation table specified with this card. (CHA (1) is added to this table.)</td>
</tr>
<tr>
<td>11 - 18</td>
<td>NAME (2)</td>
<td>The name of a second translation table if created. (CHA (2) is added to this table.)</td>
</tr>
<tr>
<td>21 - 28</td>
<td>NAME (3)</td>
<td>The name of a third translation table if created. (CHA (3) is added to this table.)</td>
</tr>
<tr>
<td>31 - 38</td>
<td>NAME (4)</td>
<td>The name of a fourth translation table if created. (CHA (4) is added to this table.)</td>
</tr>
<tr>
<td>41 - 48</td>
<td>NAME (5)</td>
<td>The name of a fifth translation table if created. (CHA (5) is added to this table.)</td>
</tr>
<tr>
<td>51 - 55</td>
<td>IOFFST</td>
<td>The character CHA in tables 1 through 5 are equivalent to number IOFFST.</td>
</tr>
<tr>
<td>56 - 60</td>
<td>INUMB</td>
<td>The number of times symbol is used consecutively after IOFFST.</td>
</tr>
<tr>
<td>61</td>
<td>CHA (1)</td>
<td>Character added to translation table 1.</td>
</tr>
<tr>
<td>62</td>
<td>CHA (2)</td>
<td>Character added to translation table 2.</td>
</tr>
<tr>
<td>63</td>
<td>CHA (3)</td>
<td>Character added to translation table 3.</td>
</tr>
<tr>
<td>64</td>
<td>CHA (4)</td>
<td>Character added to translation table 4.</td>
</tr>
<tr>
<td>65</td>
<td>CHA (5)</td>
<td>Character added to translation table 5.</td>
</tr>
</tbody>
</table>
APPENDIX D.

USE OF THE EDGE DETECTION
AND THE ADJACENCY
METHOD PROGRAMS
Appendix D.

Use of the Edge Detection and the Adjacency Method Programs

The computer programs described in the previous appendices were all written for an IBM 370/145 computer with an OS/VSI operating system. All of the programs with an exception of Subroutine Edge are catalogued at the South Dakota State University Computer Center. The use of Subroutine Edge is listed in the first section of this Appendix. Next, the JCL (Job Control Language) cards for Edge Detection Version II, SYMOVER, and the Adjacency Method are included. These JCL cards are required to run these three programs. Extraordinary JCL in these lists are briefly described. Computer run decks, which use the catalogued versions of the programs are also included for the following situations:

1. Production of a symbol map of edges located with Edge Detection Version II.
2. Production of a symbol map of fields located with the Adjacency Method.

D.1 Use of Subroutine Edge

The example program listed here reads packed data from a tape on FORTRAN unit number 10 (FT10F001), calls Subroutine Edge to detect edges, and writes a tape of edge data which is packed with FORTRAN unit number 11 (FT11F001). One parameter card is specified for this program. The parameters are described in the computer listing.
**AN EXAMPLE CALL OF SUBROUTINE EDGE**

**TAPE IN - TAPE OUT**

1. DATA IS READ FROM A PACKED TAPE INTO MATRIX MA(I,J).
2. SUBROUTINE EDGE IS CALLED.
3. THE EDGE MATRIX, MD(I,J), IS CONVERTED TO A PACKED FORMAT AND WRITTEN ON A SECOND TAPE, WHICH IS COMPATIBLE WITH THE INPUT TO SYMOVTR.
4. THE NUMBER OF LINES OF DATA, N4, AND THE BLOCK LENGTH, K4, OF THE EDGE DATA TAPE IS WRITTEN.

---

**EDGE DETECTION ALGORITHM**

**SUBROUTINE EDGE CHECK**

```fortran
COMMON MA(256,128), MD(256,128), MB(256,128), MC(256,128)
INTEGER*2 IDUMY2(1)
LOGICAL*1 IDUMY2(2)
INTEGER*2 MA, MD, LINE(256), MB, MC
INTEGER*2 NDAT(256,2)
EQUIVALENCE (IDUMY1(1), IDUMY2(1))
LOGICAL*1 LINE(512)
INTEGER*4 LINEPK(128)
EQUIVALENCE (LINE(1), LINEPK(1), LINE(1))
ICR=5

READ THE PARAMETER CARD
READ(10,1010)MXI,MXF,MYI,MYF,LRBGR
C...........MXI......STARTING HORIZONTAL DATA POSITION NUMBER.
C...........MXF......STOPPING HORIZONTAL DATA POSITION NUMBER.
C...........MYI......STARTING VERTICAL LINE NUMBER.
C...........MYF......STOPPING VERTICAL LINE NUMBER.
C...........LRBGR......GRADIENT THRESHOLD.
DO 16 I=1,MYI
16 K4=X4U(10,1000)
K4=MXF/4
MXF=K4*4
K=MXF-MXI+1
NR=MYF-MYI-2
KD=K-1
DO 10 J=1, NR
READ(10,1000)(LINEPK(I),I=1,K4)
DO 11 I=1,K
11 II=II+1
IDUMY2(I)=LINE1(II)
NDAT(I)=IDUMY2(I)
11 MA(J,I)=IDUMY2(I)
CONTINUE
CALL EDGE(KD,NR,LRBGR)
N2=NR-1
K2=K-2
K2=K-1
K4=K-3
N4=NR-4
K5=K4/4
K4=4*K5
DO 14 J=1,N4
DO 15 I=1,K4
15 IDUMY2(I)=MD(J,I)
14 LINE1(I)=IDUMY2(I)
```
**END**

*** Subroutine EDGE(K, NR, LRBR) ***

*** Function IDIRX(MX, MY) ***

/*
//GO.FT10F001 DO DSN=RS1118,UNIT=2400,VL=SER=RS1118,LABEL=(01,VL),
// DISP=(OLD,KEEP),DCB=(RECFM=F,LRECL=256,BLKSIZ=256,LEN=2)
//GO.FT11F001 DO DSN=RS1159,UNIT=TAPE,VL=SER=RS1159,LABEL=(02,VL),
// DISP=(NEW,PASS),DCB=(RECFM=U,BLKSIZ=256)
//GO.SYSIN UD *

***************

PARAMETER CARD

***************
*/

/*
D.2 Edge Detection Version II JCL

EDGE DETECTION VERSION II

// EXEC FORTGCLG
// FORT.SYSIN DD *

(MAIN PROGRAM FOR EDGE DETECTION VERSION II)

(FUNCTION IDIRX)

/

(LOCATION WHERE CATOLOGUED SUBROUTINES)
(OR FUNCTIONS ARE STORED)

//LKED.SYSLIB DD DSNAME=RONS.PLOTDEKS,DISP=SHR
// DD DSNAME=SKYBLIB,DISP=SHR

(WHERE INPUT DATA IS READ FROM)

//GO.FT14F001 DD UNIT=2400,VOL=SER=RS1244,DSN=RS1244,
// DISP=(OLD,KEEP),LABEL=(01,NL),DCB=(RECFM=U,BLKSIZE=370)

(WHERE EDGE OUTPUT IS STORED)

//GO.FT15F001 DD UNIT=DISK,VOL=SER=XCRTCH,DSN=SYMOWER,
// DISP=(NEW,KEEP),SPACE=(370,(300,50),
// DCB=(RECFM=U,BLKSIZE=370)

(PARAMETER CARDS)

/*
D.3 SYMOVER JCL

// EXEC FORTGCLG
//FORT.SYSIN DD *

(LOCATION WHERE CATALOGUED SUBROUTINES)
(OR FUNCTIONS ARE STORED)

//LKED.SYSLIB DD
// DD DSNAME=CONS.PLOTDEKS,DISP=SHR
// DD DSNAME=SKYLBLIB,DISP=SHR

(THE FOLLOWING NINE OUTPUT CLASSES STORE)
(PRINT OUTPUT. I.E. THE MAXIMUM PRINT WIDTH/PAGE = 9 x 130)

//GO.SYSOUT1 DD SYSOUT=(A,,8LPI)
//GO.SYSOUT2 DD SYSOUT=(A,,8LPI)
//GO.SYSOUT3 DD SYSOUT=(A,,8LPI)
//GO.SYSOUT4 DD SYSOUT=(A,,8LPI)
//GO.SYSOUT5 DD SYSOUT=(A,,8LPI)
//GO.SYSOUT6 DD SYSOUT=(A,,8LPI)
//GO.SYSOUT7 DD SYSOUT=(A,,8LPI)
//GO.SYSOUT8 DD SYSOUT=(A,,8LPI)
//GO.SYSOUT9 DD SYSOUT=(A,,8LPI)

(THIS DD CARD SPECIFIES WHERE THE)
(TRANSLATION TABLES USED ARE PRINTED)

//GO.TABLES DD SYSOUT=A

(TRANSLATION TABLES ARE STORED AND)
(RETRIEVED FROM THIS DISC AREA)

//GO.TABLEPDS DD DSNAME=RSI.TABLES,DCB=BLKSIZE=256,
// UNIT=2314, VOL=SER=VS1301, SPACE=(CYL,(1,1,5)),
// DISP=(OLD,KEEP)

(WHERE INPUT DATA IS READ FROM)

//GO.FT09FO01 DD UNIT=DISK, VOL=SER=XCRTCH,
// DSN=SYMOVER, DISP=(OLD,KEEP), SPACE=(CYL,(1,1,5))

//GO.SYSIN DD *

(PARAMETER CARDS)

/*
// EXEC ENDFORM
/*
D.4 Adjacency Method JCL

THE ADJACENCY METHOD

// EXEC FORTGCLG
// FORTSYSIN DD *

(ADJACENCY METHOD MAIN PROGRAM)

(SUBROUTINE TRANS)

(FUNCTION IDIRX)

/*

,LOCATION WHERE CATALOGUED SUBROUTINES)
(OR FUNCTIONS ARE STORED)

// LKEDSYSLIB DD
// DD DSN=ROSH.PLOTDEKS,DISP=SHR
// DD DSN=SKYLIB,DISP=SHR

// LKEDSYSIN DD *

(OBJECT DECK OF ASSEMBLY SUBROUTINE REDAPE)

/*

(WHERE INPUT DATA IS READ FROM)

// GO.FT14FOO1 DD UNIT=2400, VOL=SER=RSI244, DSN=RSI244, DISP=(OLD,KEEP),
// LABEL=(OL, NL), DCB=(RECFM=U, BLKSIZE=370)

(WHERE TEMPORARY FIELD DATA SET IS STORED)

// GO.FT08FOO1 DD DSN=&STEPFILE,DISP=(NEW,DELETE),UNIT=2314,
// VOL=SER=XCRCH, SPACE=(CYL,(4,1)), DCB=(RECFM=U,BLKSIZE=520)

(WHERE PERMANENT FIELD DATA SET IS SAVED)

// GO.FT10FOO1 DD DSN=RSI159, UNIT=TAPE, VOL=SER=RSI159, LABEL=(1, NL),
// DCB=(RECFM=U, BLKSIZE=520), DISP=(NEW, PASS)

// GO.SYSIN DD *

(PARAMETER CARDS)

/*
D.5 Example Run Deck for Symbol Maps of Edges

The computer run deck which produces edge maps is included here. Both Edge Detection Version II and program SYMOVER were used. During the initial run of SYMOVER, when displaying edges, all 49 characters can be used to display 49 different gradient thresholds. A second run of program SYMOVER is used to assign blanks to those edges mapped during the initial run, which were noisy.

```
   /* DETECT EDGES WITH EDGE DETECTION VERSION II */
   // EXEC PGM=EDGE
   // STEPLIB DD DSN=SKYLBLIB,DISP=SHR
   // GO.FT14F001 DD UNIT=2400, VOL=SER=RSI244, DSN=RSI244,
   // DISP=(OLD,KEEP), LABEL=(1,NL), DCB=(RECFM=U, BLKSIZE=370)
   // GO.FT15F001 DD UNIT=DISK, VOL=SER=XCRTCH, DSN=SYMOVER,
   // DISP=(NEW,KEEP), SPACE=(370,(300,50)), DCB=(RECFM=U, BLKSIZE=370)
   // GO.SYSIN11 DD *

   (PARAMETER CARD ONE)
   THIS TITLE IS PRINTED ON ALL PAGE OUTPUT TOPS OF PROGRAM OUTPUT

   (PARAMETER CARD TWO)
   MXI  MXF  MYI  MYF  NFRQ  LRBGR
   1   370   1   370   1   5

   /*
   
```
D.6 Example Run Deck for Symbol Maps of Fields

The computer run deck, which produced the field map illustrated in Figure 4.3, is included here. It is possible to have unique symbols for each field number (values 6 to 255 indicate fields), if two symbol translation tables are used rather than the one table used here. Two tables are necessary because there are only 49 discretely different symbols.

THE ADJACENCY METHOD AND SYMOVER

(EXAMPLE USE OF CATALOGUED VERSIONS)

// EXEC   PGM=RSTADJ
// STEPLIB   DD DSN=SKILLBLIB,DISP=SHR
// FT05F001 DD DDNAME=SYSIN
// FT06F001 DD SYSOUT=A
// FT14F001 DD UNIT=2400, VOL=SER=RSI244, DISP=(OLD,KEEP),
// LABEL=(1,NL), DCB=(RECFM=UBLKSIZE=370)
// FT08F001 DD DSN=&TEMPFILE, DISP=(NEW,DELETE), UNIT=2314,
// VOL=SER=XCRCH, SPACE=(CYL,(4,1)), DCB=(RECFM=U, BLKSIZE=520)
// FT10F001 DD DSN=RSI159, UNIT=TAPE, VOL=SER=RSI159, LABEL=(1,NL),
// DCB=(RECFM=U, BLKSIZE=520), DISP=(NEW, PASS)
// GO. SYSIN DD *
)*(PARAMETER CARD)
MXI  MXF  MYI  MYF
1 370 1 370

/*

/* DISPLAY FIELDS WITH SYMOV

EXEC PGM=SYMOV
// STEPLIB DD DSN=SKYBLIB, DISP=SHR
// GO.SYSOUT1 DD SYSOUT=(A,,BLPI)
// GO.SYSOUT2 DD SYSOUT=(A,,BLPI)
// GO.SYSOUT9 DD SYSOUT=(A,,BLPI)
// GO.TABLES DD DSNNAME=RSI.TABLES, DCB=BLKSIZE=256, UNIT=2314,
// VOL=SER=VSI301, SPACE=(CYL,(1,1,5)), DISP=(OLD,KEEP)
// GO.FT09F001 DD DSN=RSI159, UNIT=TAPE, VOL=SER=RSI159, LABEL=(1,NL),
// DCB=(RECFM=U, BLKSIZE=520), DISP=(OLD, KEEP)
// GO.SYSIN DD *

(PARAMETER CARD ONE)
MXI  MXF  MYI  MYF  IX  IY
1 365 1 365 1 1

(PARAMETER CARD TWO)
NUMSTK NUMCTB
1 1

(PARAMETER CARD THREE)
NAME  NDISP  NTYPE  ICARD
TABLE1 1 0 5

(SYMBOL PARAMETER CARD TYPE ONE)
IOFFST INUMB  CHA
0  62   GHIJKLmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnop
62  62   621234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnop
124  124  62QRSTUvxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnop
186  186  62GHIJKLmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnop
248  248  77890abc

/*
EXEC ENDFORM
/*
BIBLIOGRAPHY


