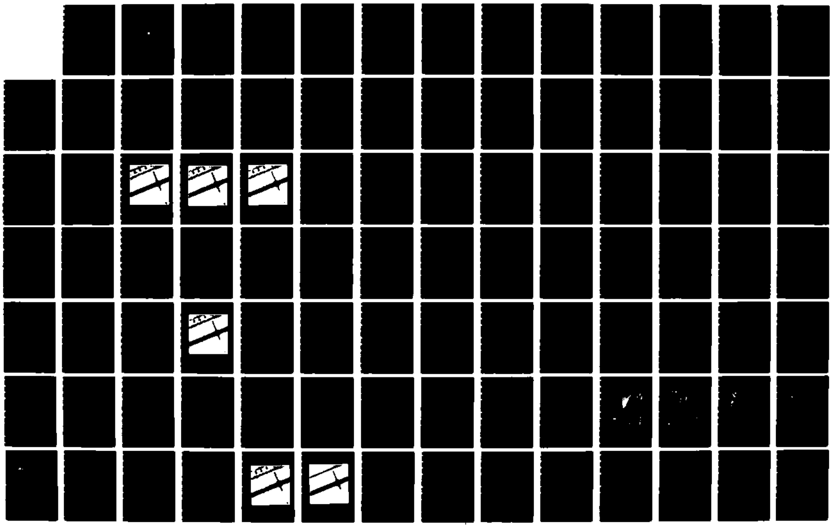
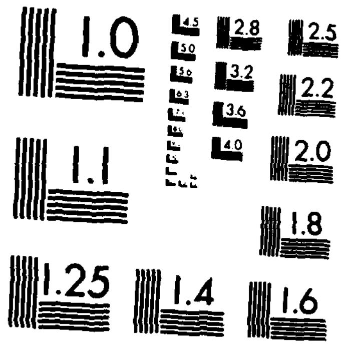
AD-A172 323   INTERACTIVE RECONSTRUCTION OF COMPRESSED IMAGES(U)   1/2
NAVAL POSTGRADUATE SCHOOL MONTEREY CA A LEDESMA JUN 86

UNCLASSIFIED   F/G 9/2   NL




1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

2

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

INTERACTIVE RECONSTRUCTION OF COMPRESSED IMAGES

by

Alfred Ledesma

June 1986

Thesis Advisor Chin-Hwa Lee

AD-A172 323

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School — 6b OFFICE SYMBOL (If applicable) 62	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	7b. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION — 8b. OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
INTERACTIVE RECONSTRUCTION OF COMPRESSED IMAGES

12 PERSONAL AUTHOR(S)
Alfred Ledesma

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM ____ TO ____	86 June 20	125

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Image Processing; Data Compression; Image Reconstruction

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

By using irregular sampling, an image may be digitized by concentrating sampling in areas of interest and reducing sampling in other areas. With this approach storage required for the digital image is more efficiently used and is usually reduced.

This thesis explores an interactive method for reconstructing a compressed image. An interactive method allows human intuition to be combined with the speed and versatility of a computer. The user is able to guide the reconstruction through difficult cases of loop connectivity, branch linkup, and triangulation. MOSAIC, an element of MOVIE.BYU, is used for this reconstruction.

This study shows that an interactive method has both good and bad points. Better reconstruction results when working with large, well-formed contours. When working with small, irregular shaped contours a significant loss of detail occurs.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL Prof Chin-Hwa Lee	22b TELEPHONE (Include Area Code) (408)646-2190 — 22c OFFICE SYMBOL 62Le

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

Interactive Reconstruction of Compressed Images

by

Alfred Ledesma
Lieutenant, United States Navy
B.S.E.E. University of Texas, August 1979

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June, 1986

Author: [illegible]
Alfred Ledesma

Approved by: [illegible]
Chin-Hwa Lee, Thesis Advisor

Mitchell L. Cotton
Mitchell L. Cotton, Second Reader

[illegible]
Harriett B. Rigas
Chairman, Department of Electrical
and Computer Engineering

J.N. Dyer
John N. Dyer
Dean of Science and Engineering

ABSTRACT

By using irregular sampling, an image may be digitized by concentrating sampling in areas of interest and reducing sampling in other areas. With this approach, storage required for the digital image is more efficiently used and is usually reduced.

This thesis explores an interactive method for reconstructing a compressed image. An interactive method allows human intuition to be combined with the speed and versatility of a computer. The user is able to guide the reconstruction through difficult cases of loop connectivity, branch linkup, and triangulation. MOSAIC, an element of MOVIE.BYU, is used for this reconstruction.

This study shows that an interactive method has both good and bad points. Better reconstruction results when working with large, well-formed contours. When working with small, irregular shaped contours a significant loss of detail occurs.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I wish to gratefully acknowledge my thesis advisor, Professor Chin-Hwa Lee, who provided invaluable assistance in the completion of this thesis.

I would also like to express by gratitude to Professor Mitchell L. Cotton for his assistance.

I. INTRODUCTION

A. BACKGROUND

Image processing can be described as the alteration and analysis of a picture for such purposes as enhancement and recognition. Although improvements in processing methods for transmitted digital pictures have continued over the last four decades, it was not until the advent of large-scale digital computers that many of these methods became practical.

Digital image processing techniques are playing a key role in a variety of problems. Any area that has a need for methods capable of enhancing pictorial information for human interpretation and analysis can benefit from image processing. The space program, an early innovator in image processing, enhanced and restored images from such programs as the Surveyor missions to the moon, the Mariner missions to Mars, and the Apollo manned flights to the moon. The improvement of the processed pictures over the unprocessed pictures was truly amazing [Ref. 1]. In the medical field, physicians are assisted by computer procedures that enhance the contrast or code the intensity levels into color for easier interpretation of xrays and other biomedical images. Other successful applications of image processing concepts can be found in archeology, astronomy, biology, law enforcement, and defense applications.

A basic, general-purpose digital image processing system is shown in Figure 1.1 [Ref. 2]. The operation of the system may be divided into three principle categories:

Digitization

Processing

Display

B. DIGITAL IMAGES

A digital image is an image which has been discretized both in spatial coordinates and in brightness. The spatial resolution is the number of pixels into which an image is divided, indicating the precision and accuracy horizontally and vertically. The brightness or gray level may be viewed as a third axis.

In digital image processing systems one usually deals with arrays of numbers obtained by spatially sampling points of an optical image. After processing, another array of numbers is produced, and these numbers are then used to reconstruct a continuous image for viewing. A typical representation of a digital image is a matrix. The row and column indices identify a point in the image and the corresponding matrix element value identifies the gray level at that point.

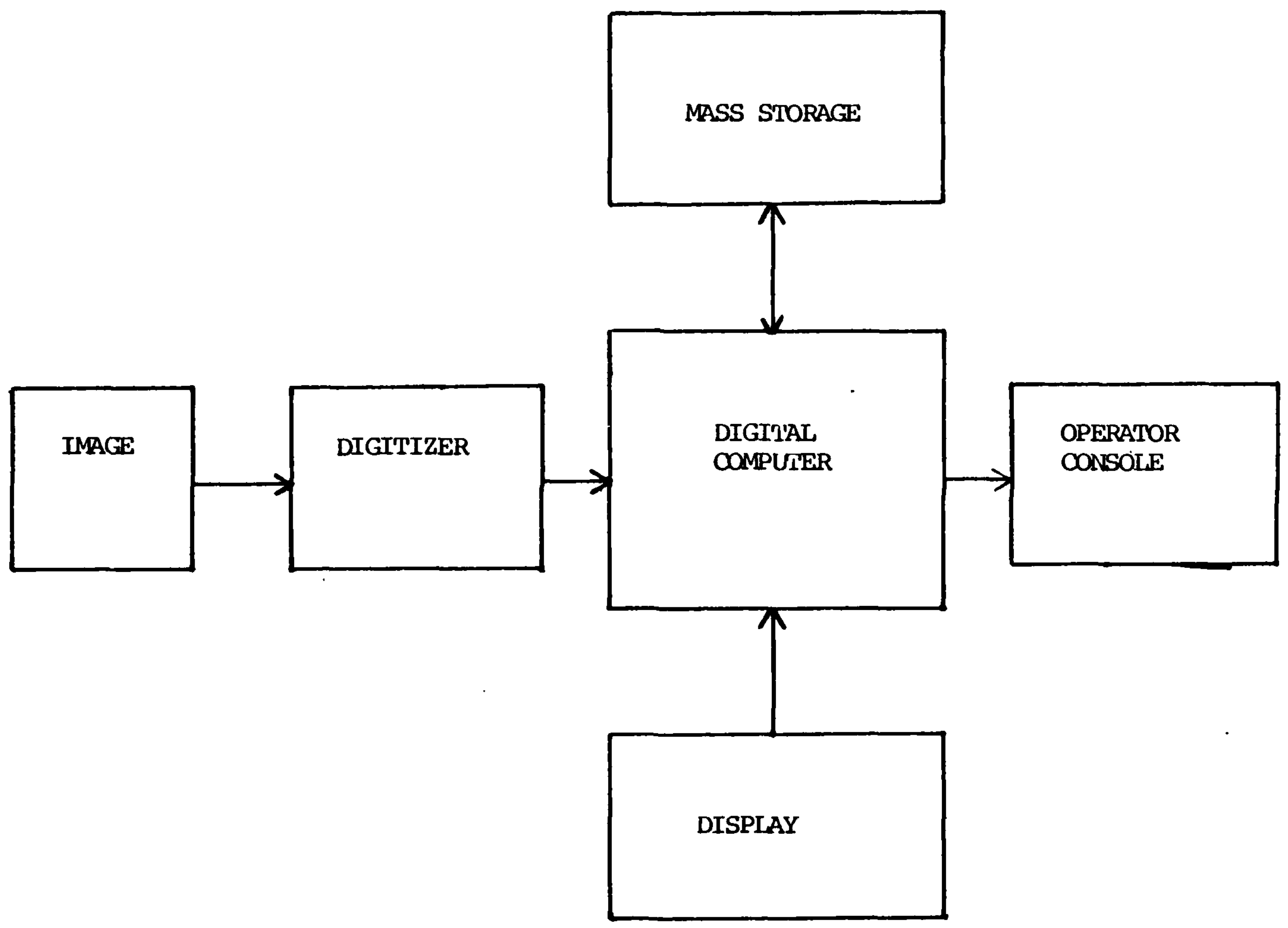


Figure 1.1 Elements of a Digital Image Processing System

The number of gray levels and dimensions of the matrix may vary from application to application however there are advantages to using a number of gray levels which are integer powers of two and to selecting a square array. As a reference, a typical monochrome TV image is a 512 x 512 array with 128 gray levels. A minimum system for general image processing work should be able to display 256 x 256 pixels with 64 gray levels.

C. IMAGE PROCESSING TECHNIQUES

Techniques for image processing may be divided into four principle categories:

Image Digitization

Image Enhancement and Restoration

Image Coding

Image Segmentation and Representation

Image digitization deals with converting continuous brightness and spatial coordinates into discrete components. Image enhancement and restoration concerns the improvement of a given image for human or machine perception. Image coding is used to reduce the number of bits in a digital image. Image segmentation and representation deals with the decomposition of an image into a set of simplier parts and the organization of these parts in a meaningful descriptive manner. [Ref. 2]

The exact approach to the above depends on which of two broad categories one chooses to work in. Processing techniques in the first category are based on modifying the Fourier transform of an image. The basis for this is the linear system convolution theorem. The spatial domain refers to the image plane itself, and approaches in this category are based on direct manipulation of the pixels in the image plane.

D. PROBLEM DESCRIPTION

Digital representations of images usually require a large number of bits. The logical solution is to consider techniques that can represent an image with fewer bits. When the transmission or storage of a signal requires excessive channel or storage capacity, the requirement can be reduced by more efficient coding. This is referred to as source or image encoding. Applications of image encoding fall into one of three categories:

Image Data Compression

Image Transmission

Feature Extraction

Data compression applications are motivated by the need to reduce storage requirements. In image transmission applications interest lies in techniques which achieve maximum reduction in the quantity of data to be transmitted.

Feature extraction applications are used primarily for pattern recognition by computers.

The resulting problem from the use of the above techniques is how to obtain an image that is acceptable for visual or machine analysis. Alternatively, the problem may be viewed as how to improve an image that has been modified by one of the above applictions into an acceptable image.

This thesis will concentrate on the reconstruction of images that have been compressed. During the reconstruction phase, existing numerical algorithms are not always sufficient. Loop and contour connectivity often causes problems where there are several loops on adjacent levels. This causes features on the reconstructed image to be distorted with sometimes unacceptable results. An interactive method allows human intuition to be combined with the speed and versatility of a computer. This should result in a better reconstructed image. A user can interact with the program to guide it through difficult cases of loop connectivity, branch linkups, and triangulation as well as to form patches.

Programs written by C. T. Miranda will be used to compress digital images. The specific interactive syst used here is Movie.BYU which is distributed by Brigham Young University. Movie.BYU is a general purpose computer graphics software system.

The purpose of this study is to investigate the interactive procedure for image reconstruction. Answers to the following questions will be sought. Can this procedure achieve better results than other types of procedures? What problems are encountered when using an interactive method? What are the nature and characteristics of problems that are encountered? What skill level is required of the interactive user? What hardware demands are needed to implement such a system? And, finally, can this procedure be used in practical situations?

II. IMAGE COMPRESSION

A. INTRODUCTION

Digital representation of images usually requires a very large amount of data. When the number of images that must be retained increases, the storage requirement increases. An example is the storage of X-ray pictures in hospitals. Storage requirements are immense and retrieval of specific images is difficult.

It is important to consider techniques for representing an image, or the information contained in the image, with fewer bits. Image data compression applications are motivated by this need to reduce storage requirements. An additional benefit gained from data compression is a reduction in the quantity of data to be transmitted. With less data, image transmission time and the required bandwidth are reduced. [Ref. 3]

B. IMAGE SURFACE APPROXIMATION WITH IRREGULAR SAMPLES

For most images the objects of interest are localized. Most of the image consists of a background. This background has little or no variation of gray levels. As a result of this, using spatial sampling with regular spacing between the samples wastes storage space for areas with little gray level variation and little interest to the viewer.

If a method could be devised that allows sampling to be concentrated at areas of interest and to be reduced in other areas, the storage required for an image could be more efficiently used and probably reduced. The problem is how to best select these irregularly spaced samples without causing any significant loss of the information in the image.

C. B-SPLINE FUNCTIONS

Two general approaches exist for image coding. One utilizes spatial domain techniques, the other operates in the transform domain of images. The approach used throughout this thesis is a spatial domain technique and is based on the use of B-splines. B-splines are used to approximate contours using variable knots.

A formal definition of a spline function is as follows [Ref. 4]: a function S(x), defined in the range $a \leq x \leq b$, is called a spline function of degree k with knots t_i, $i=1,2,...,n$ if the following conditions are satisfied:

(1) In each interval $[t_{i-1}, t_i]$, $i=1,2,...a+1$ $(a=t, b=t_{n+1})$, S(x) is given by a polynomial of degree k or less.

(2) S(x) and its derivatives of order 1,2,...,k-1 are continuous everywhere in the interval [a,b].

The order of the spline function is defined as

$$m = k + 1 \tag{2.1}$$

where m is the order of spline and k is the degree of the spline.

The piecewise polynomial spline defined above can be characterized as a smooth function that fits at the break points satisfying some ending point requirements. These constraints have the following drawbacks [Ref. 5]:

(1) If we are processing experimental data points, most of the time we are not interested in fitting a curve exactly through this set of points. The reason is that there is an intrinsic error associated with these measurements. It is preferable to fit a curve that presents a tradeoff between closeness of fit and smoothness.

(2) If the number of data points is large the solving of the tridiagonal system associated with the spline function is time consuming and requires a lot of storage.

(3) The lack of the ability to manipulate the curve fitting is a poor characteristic for a curve fitting algorithm.

B-spline functions which are based on linear space theory overcome the above constraints and are well suited for use in image processing. Fundamentals of B-spline functions are discussed by De Boor [Ref. 6]. Andrew [Ref. 7] discusses some early and important applications of B-splines for image processing.

Some important properties of B-spline functions in the area of image processing include the following [Ref. 5]:

(1) Local Basis: Only k+1 B-splines have non zero value at any particular interval $[t_i, t_{i+1}]$. This property suits very well the nonglobal characteristics of the pixels in an image plane.

(2) Non-Negative Basis: The B-spline is non-negative which is an asset to image processing since the signals represented are usually light intensities and are always positive quantities.

(3) Differentiation and Integration: The B-spline can be evaluated from a recursive relation which allows the numeric differentiation and the integration to be carried out efficiently by just subtracting or adding the coefficients respectively.

D. APPLICATION OF B-SPLINE FUNCTIONS IN KNOT SELECTION

Miranda [Ref. 5] uses the B-spline basis as "fundamental entities for a data compression model." Miranda's method can be broken up into two steps.

First the image contours are generated from the data over a rectangular and uniform grid representing the image. The information contained in the image signal and the number of contour levels or layers chosen determine the total number of contour lines.

Next, from the image contours generated above, knots are selected so that the image is represented as data over a set of uneven knots. It contains less elements than the original data set, but at the same time all important features of the original signal are maintained.

In order to extract important knots from contours, the contours must be stored in such a manner that is convenient for the knot selection algorithm.

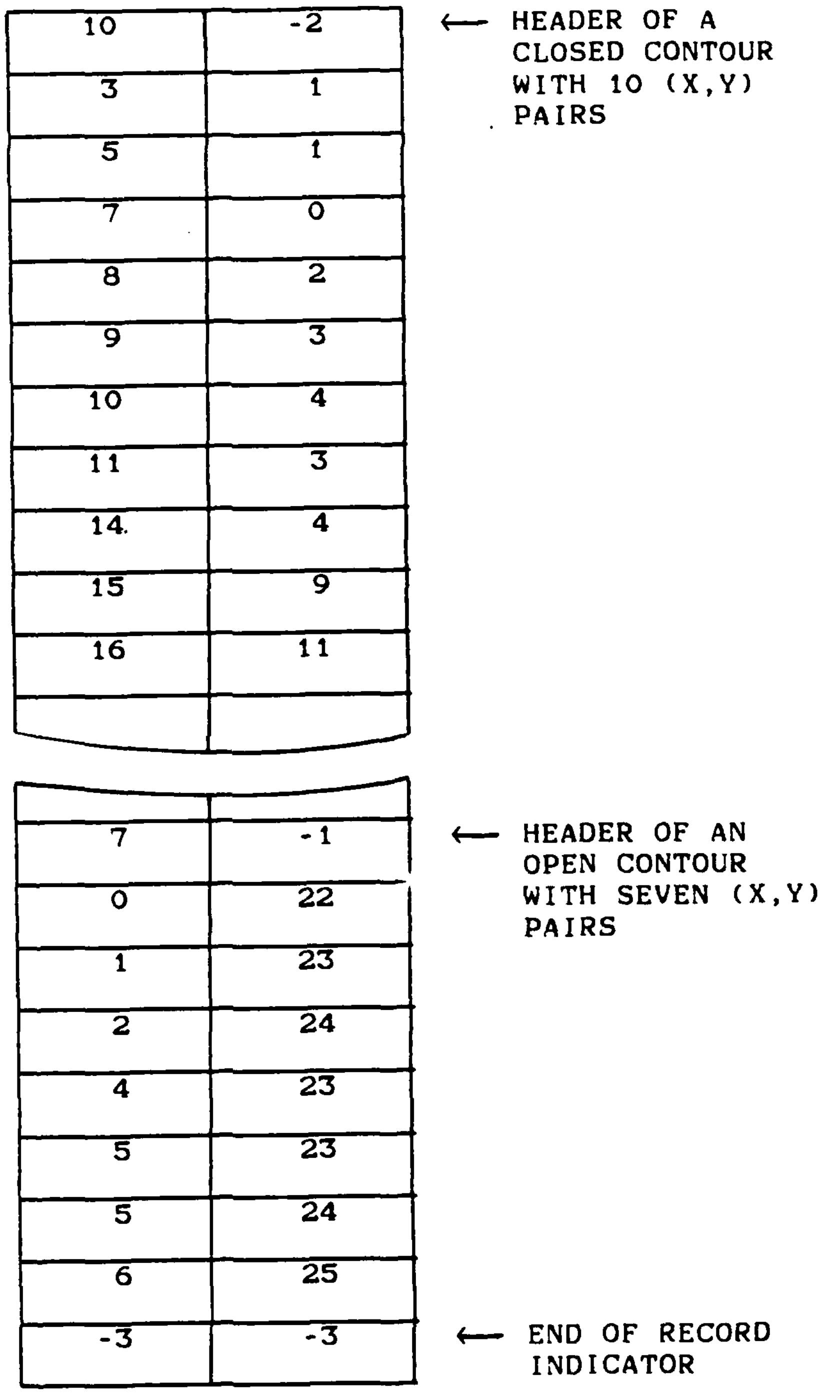
Miranda developed a series of programs that take as inputs an image represented as a uniform grid and user controlled parameters that indicate the number of levels and a threshold that eliminates contours with less points than this threshold. The threshold level has a smoothing effect on noisy images. The final output of this series of programs are three files.

File 1 is a contour record. It contains the (X,Y) pairs that make up the coordinates of each contour. The contours are concatenated together with the first two entries of each contour indicating the number of points on the contour and whether the contour is open (-1) or closed (-2). An end marker (-3, -3) is used to indicate the end of data. Figure 2.1 is an example of a contour record.

File 2 is a character matrix with the same dimension as the given image. It contains the projections of all contours of the image surface. Each layer is coded by a character starting from character A (lowest contour level) through character Z (highest contour level). Figure 2.2 is an example of a character matrix.

File 3 is a layer record indicates the gray level of each layer and how many contours there are at each gray level. Figure 2.3 is an example of a layer record.

After the contour information has been stored in a convenient format it is time to fit a smooth spline function to each contour. The goal is to define a measurement for the smoothness of the approximating splines and another measurement for their closeness of fitting so that these two properties, usually contradictory, can be controlled by a single parameter.

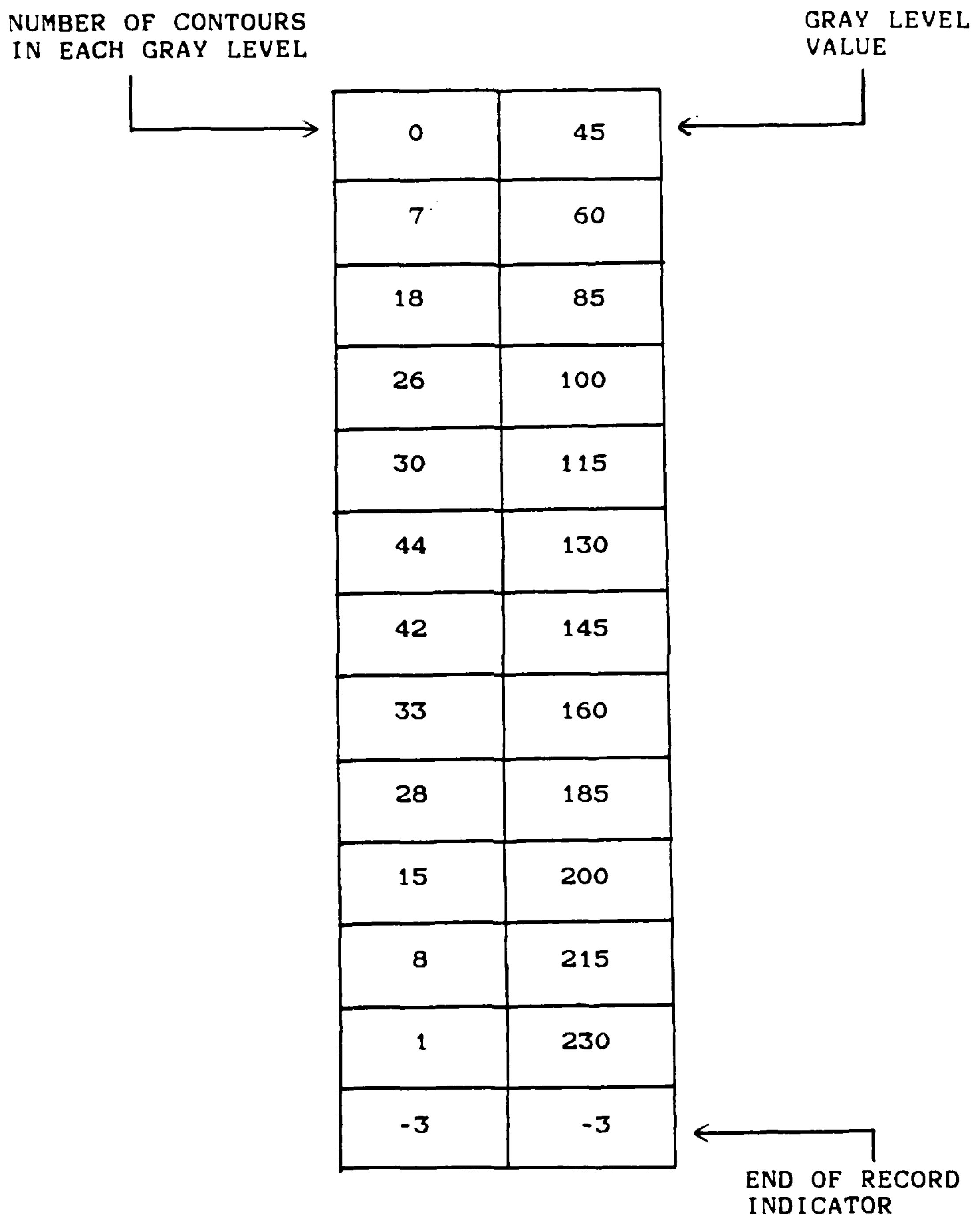


10	-2	← HEADER OF A CLOSED CONTOUR WITH 10 (X,Y) PAIRS
3	1	
5	1	
7	0	
8	2	
9	3	
10	4	
11	3	
14.	4	
15	9	
16	11	
7	-1	← HEADER OF AN OPEN CONTOUR WITH SEVEN (X,Y) PAIRS
0	22	
1	23	
2	24	
4	23	
5	23	
5	24	
6	25	
-3	-3	← END OF RECORD INDICATOR

Figure 2.1 Contour Record Format

```
                          C                 D
                  B       C                 D D D
        A A A A   B       C C C       C C C     D
  A A A A     A   B           C C C C C   C   D D
  A           A   B B B B B               C   D
A A           A           B B B B B       C   D D
A             A A A A A A A       B       C     D D
A A           B B B B     A A A   B B     C
  A A         B     B         A     B     C C C C
    A A A     B     B         A     B           C
        A     B B   B         A A   B         C C
      A A       B B B           A   B         C
      A                         A   B         C C
      A A A A                 A A   B           C
            A A A             A     B   C C C C C
                A A A   A A A A     B   C
B B B               A A         B B     C
    B B B B                   B B       C
          B   B B B B B B B B   B       C
          B B B             B B B       C
                                        C
                    C C C C C C C C C   C C C
                    C               C       C
C C C C C C C C C C C               C C C C C
```

Figure 2.2   Character Matrix Format



0	45
7	60
18	85
26	100
30	115
44	130
42	145
33	160
28	185
15	200
8	215
1	230
-3	-3

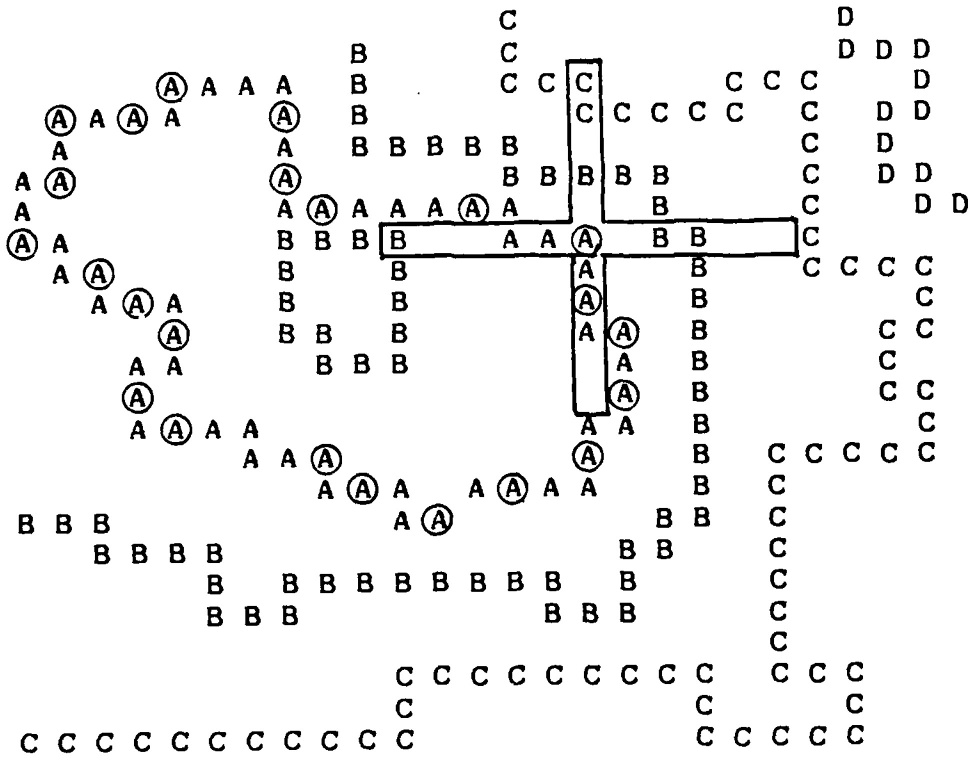
Figure 2.3 Layer Record Format

Miranda develops and implements an algorithm that allows the user to automate the determination of the knots. Since the number of selected knots is much lower than the initial contour points, high compression rates can be achieved.

The following example gives an overview of this algorithm. For further details see Chapter 3 of Miranda's work [Ref. 5].

Figure 2.4 is a superimposition of the contour record, Figure 2.1, and the character matrix, Figure 3.2. The character matrix is composed of fully connected contours represented in letters, while the contour record consists of the contours represented by unevenly spaced knots marked with a circle. The contour density measurement is accomplished by moving the isometric cross along each contour so that is is centered on every contour record point (circled character in Figure 2.4). Information about the distribution of contours in that specific neighborhood can be validated by checking in four directions spaced 90 degrees apart. The length of each arm of the isometric cross is set by a parameter which is input by the user. A value of five for this parameter was found by Miranda to be a good choice.

The other user supplied parameter to this algorithm is the smoothing factor. This parameter controls the smoothing and fitting of the contours. A small value gives a function that fits closer but is less smooth due to the large number



LEGEND

○	contour point
✚	sliding window

Figure 2.4 Density Measurement Scheme

of selected knots. A large value gives a smoother function that does not fit as close. The final choice on a value for the smoothing factor depends on the problem at hand and the desired results.

Figures 2.5, 2.6, and 2.7 demonstrate the effects of this algorithm on an image. Figure 2.5 is the original image that is to be processed using the programs written by Miranda. Using a compression ratio of 30, Figure 2.6 is the resulting reconstructed image. Figure 2.7 is the reconstructed image after applying a compression ratio of 40.

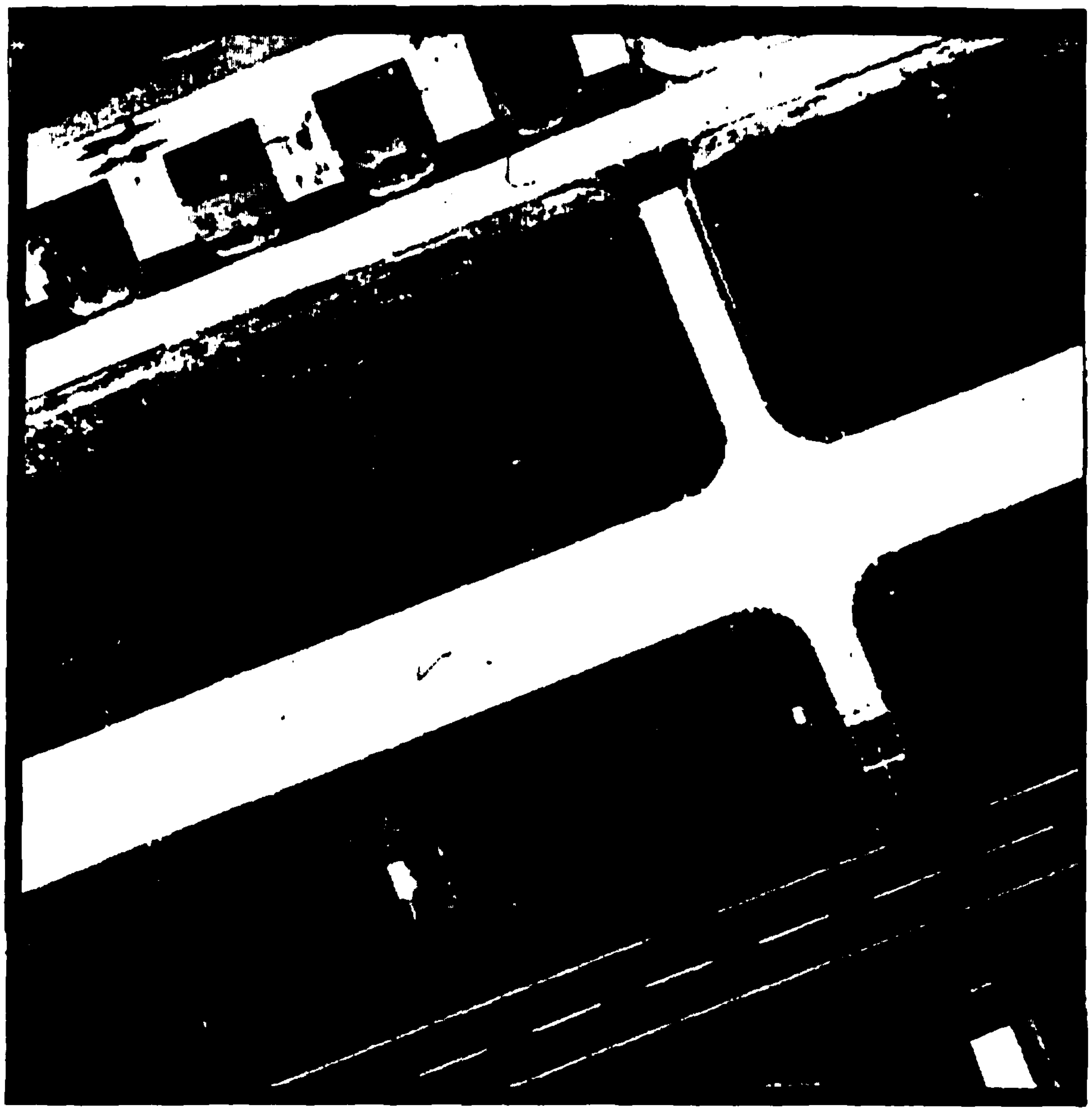


Figure 2.3 Original Image

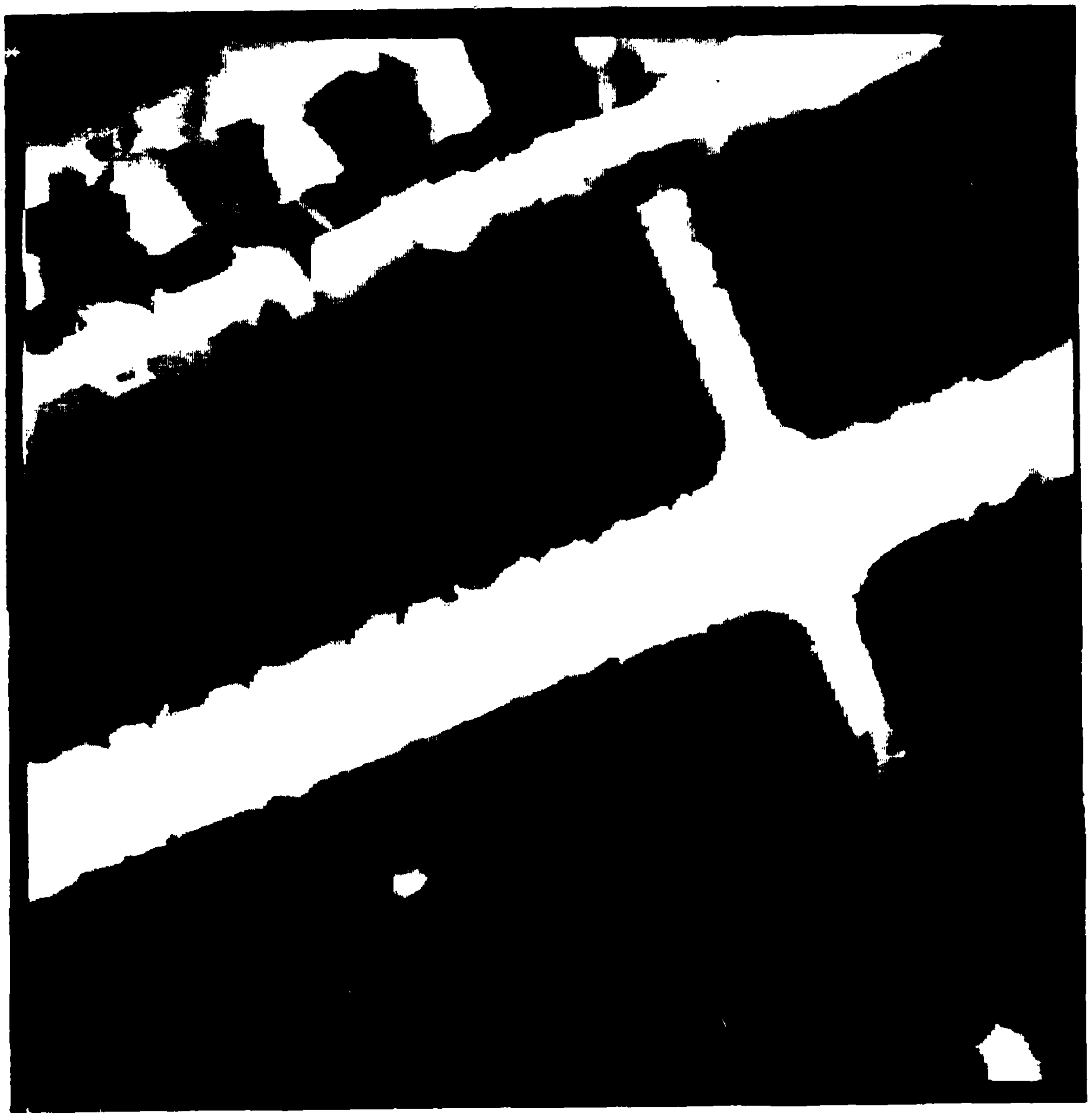


Figure 2.6 Reconstructed Image
Compression Ratio = 30

Figure 2.7 Reconstructed Image
Compression Ratio = 40

3. IMAGE RECONSTRUCTION

A. BACKGROUND

The true test of any image coding algorithm must be based on a comparison of the original image with a reconstructed image. The criteria for this comparison may vary from application to application. Depending on the intended use of a reconstructed image there may be acceptable differences between the original and reconstructed images.

There is no single optimum method for image reconstruction. There is a "best" restoring method relative to the type of a priori *information* regarding the object and noise one might have. Other, more peripheral factors, such as the amount of data to be processed, and permissable computer cost, also must enter into the choice. These factors must often be balanced against an "optimum" choice based purely on accuracy. [Ref. 1]

A major issue in image reconstruction is the intended "evaluator" of the reconstructed image. A person may pass favorable judgement on a reconstructed image while a machine may reject the reconstructed image. A person basically makes a qualitative judgement between the original and reconstructed images. On the other hand a computer usually makes a quantitative evaluation of the two images. This quantitative evaluation is based on mathematical relationships.

B. QUALITATIVE FIDELITY CRITERIA

When the output images are to be viewed by people who usually use a subjective fidelity criteria corresponding to how good the images appears to human observers, it is best to use qualitative criteria.

Since digital images are displayed as a discrete set of brightness points, the ability of the eye to discriminate between different brightness levels is an important consideration in presenting image processing results. The human visual system can adapt to a wide range of light intensities. This range is on the order of 10^{10} from the scoptic threshold to the glare limit.

A key point however is that the visual system can not operate over this entire range simultaneously. It accomplishes this large variation by sliding the center of its sensitivity range, a phenomenon known as brightness adaptation. The human eye is only able to differentiate between about 30 gray levels and about 120 different colors. [Ref. 5]

Another feature of the human eye is its tendency to filter out any abrupt changes in brightness levels. This can be compared to a smoothing effect.

There are no mathematical relationships that can indicate the qualitative fidelity of a reconstructed image. It is entirely a subjective matter. A commonly used scale for reporting the qualitative fidelity of an image was developed by Panel 6 of the Television Study Organization [Ref. 8]. This scale ranges from excellent to unusable with the following definitions:

(1) Excellent: The image is of extremely high quality, as good as you could desire.

(2) Fine: The image is of high quality providing enjoyable viewing. Interference is not objectionable.

(3) Marginal: The image is of acceptable quality. Interference is not objectionable.

(4) Inferior: The image is of poor quality but you could watch it. Objectionable interference is definitely present.

(5) Unusable: The image is so bad that you can not watch it.

C. QUANTITATIVE FIDELITY CRITERIA

Some image transmission systems can tolerate errors in the reconstructed image. When this is the case, a fidelity criteria can be used as a measure of system quality. The basic measurement of the error between an input pixel and the corresponding output pixel is the difference in gray levels between these two pixels or expressed mathematically

$$e(x,y) = g(x,y) - f(x,y) \qquad (3.1)$$

where g(x,y) is the output image signal and (x,y) is the spatial coordinates. [Ref. 2]

One can apply this error function, $e(x,y)$ in several ways to come up with a measurement of quantitative fidelity criteria. Some examples of this are the root-mean-square (rms) error between the input image and output image, and the rms signal-to-noise ratio of the output image. The root-mean-square error is defined as the square root of the squared error averaged over the entire image. The signal-to-noise ratio can be defined as the square root of the peak value of $g(x,y)$ squared and divided by the root-mean-square error. Gonzales [Ref. 2:pp. 229-231] has a complete development of the specific equations.

D. SURFACE RECONSTRUCTION

Now that some guidelines for comparing a reconstruction image with the original image have been laid, a method must be found to reconstruct the set of irregular samples given by (x_k, y_k, f_k), $k=1,2,\ldots,N$. It is necessary to find a smooth function $F(x,y)$ so that $F(x_k, y_k) = f_k$, $k=1,2,\ldots N$. Many interpolation methods exist that are based on either a global interpolation method or a local interpolation method.

To simplify the reconstruction process and keep the computational time required reasonable, local interpolation methods will be used.

In local interpolation methods, $F(x,y)$ depends only on a limited neighborhood of f_k's. The justification for using these methods is that the correlation between two pixels in an image usually decreases rapidly with the distance between them. The influence a sample exerts on another sample is inversely proportional to the distance away from it. Two commonly used local interpolation methods are the triangle element method and the local thin plate spline method.

The method to be used in this paper is based on the triangle element method. In the triangle element method, the first step involves building a triangular network covering the convex hull of the samples. The (x_k, y_k) points become the vertices of the triangular cells. The second step involves the evaluation of an arbitrary point (x,y) in the output array. This is done by linearly interpolating the point from the values at the vertices of the triangle to which the point belongs. A step by step procedure to construct a uniform grid over a set of scattered points (x,y) is given by Lawson in a comprehensive paper [Ref. 9]. Figure 3.1 is an example of a triangulation grid.

Unfortunately any partition of a convex hull into triangular cells is not unique. Some triangularizations will be superior to others in the sense that less long thin triangles are generated and a better reconstruction of the original image results.

The algorithm discussed by Lawson is well suited for automatically generating these triangles by computational methods. A brief summary of the method follows.

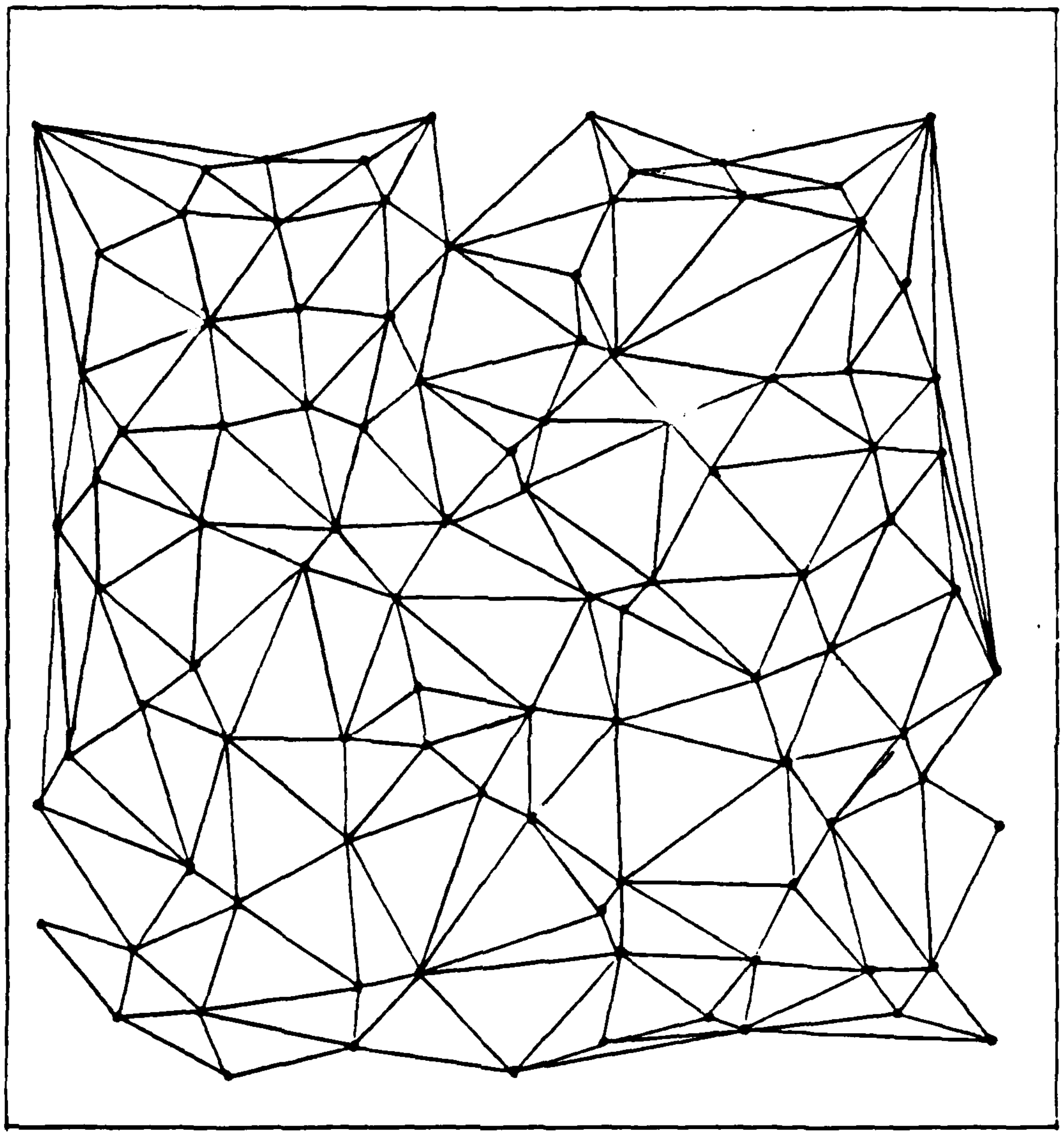


Figure 3.1 Triangulation Grid

Using the three closest pairs of points, the first triangle is constructed. A point is added in a sequential manner in the ascending order of the distance from the midpoint of the existing edge visible to it. The added triangle and the neighboring triangle together form a quadrilateral. The algorithm next tests whether the alternative dissection can maximize the minimum interior angles of the resultant two triangles. Whenever necessary, an exchange of diagonals in the quadrilateral is conducted.

Unfortunately automatic numerical algorithms are not always sufficient in the reconstruction phase. Several loops on adjacent contour levels often causes a problem where triangulation does not follow the contour lines. This causes features on the reconstructed image to be distorted with sometimes unacceptable jagged contours. By being able to interactively guide the triangulation process through high gradient areas, the reconstructed image will hopefully show an improvement over the image reconstructed by a method similar to Lawson's.

The remainder of this paper discusses an interactive method for triangulation of a compressed image and compares results with the triangulate method.

4. MOVIE.BYU

A. OVERVIEW

In order to interactively reconstruct an image one must have a set of software tools. This is not a simple matter. The complexity required of such a program is large. This is especially true when dealing with images that include both man-made objects and natural objects. Man-made objects usually have well-defined shapes that are easily discernable when they are viewed in contour lines. The opposite is true for natural objects. These contour lines do not always give a true indication of the features they represent. In the presence of noise their irregular shapes usually cause problems in reconstructing the image for display. Any program tools used to interactively construct an image must have a way to allow the user to deal with this kind of bad contours.

B. INTRODUCTION TO MOVIE.BYU

The particular set of programs selected for use in the image reconstruction process is MOVIE.BYU Version (5). MOVIE.BYU is a general purpose computer graphics software system distributed by Brigham Young University. The basis for the remaining discussion on MOVIE.BYU is the MOVIE.BYU

Training Text [Ref. 10] and the MOVIE.BYU User's Manual [Ref. 11].

The elements of the MOVIE system are FORTRAN programs for the display and manipulation of data representing mathematical models. The geometry of these mathematical models may be described in terms of polygonal elements, polyhedral solid elements, or contour line definitions.

All input to the MOVIE system is read as alphanumeric characters with each character interpreted individually. This circumvents all of the restrictions and conventions of FORTRAN input formats and makes the code virtually "bomb proof." The key words, integers, and real numbers are then reconstructed in software and the commands are interpreted from the resulting list of key words and numbers. The current edition of MOVIE.BYU has also been enhanced with capabilities to generate and access command files. This simplifies repetitive input of the same series of commands.

The hardware requirement for MOVIE.BYU is a time sharing digital computer with a word length of at least 32 bits. Software interfaces for most major graphic terminals are available including a special software interface for the Tektronix 4027 which significantly extends the color possibilities by the use of patterns which simulate additional intensities for each color gun.

C. ELEMENTS OF MOVIE.BYU

The MOVIE system is composed of six elements - DISPLAY, UTILITY, SECTION, TITLE, MOSAIC, and COMPOSE. "The six programs in MOVIE work in harmony to provide displays of the data in line drawing or continuous tone image format, to clip and cap three dimensional systems to expose internal surfaces, to modify geometry, displacement, and/or scalar function files by way of correction, appendage, or symmetry operations, to generate new models or representations, to convert complex contour line definitions into polygonal element mosaics, or to view multiple models simultaneously." [Ref. 11:pp. 1.1]. A brief description of each element follows.

DISPLAY is the "heart" of the system. It is an interactive program for the display and animation of any model composed of polygons. DISPLAY allows the user to manipulate the model (rotate, translate, etc.), specify colors for the background and the different element parts, and select the display device.

UTILITY is a data generation and editing program which allows the user to produce and/or edit models of two or three dimensional polygonal systems. The FORTRAN data files created are in a format which is compatible with the other programs in the MOVIE system.

SECTION is a program used to modify solid data representations so that they are compatible with DISPLAY.

TITLE is a program to generate two and three dimensional characters whose data format is the same as that used by the other programs. The user enters a line of text which is converted into characters composed of polygons according to specifications given by the user.

MOSAIC is an interactive program that converts complex surfaces defined by contour lines into mosaics of triangular and quadrilateral elements.

COMPOSE is a program that allows the retrieval of data generated by DISPLAY (with the RECOrd option) to compose drawings with multiple images and text.

Of the six modules listed above only two are of interest in this work. MOSAIC is the program that allows a user to interactively control the triangulation of a muliple level contour representation of an image. DISPLAY is of interest in that it allows the user to get a three dimensional perspective of the connectivity of the layers after the triangulation is completed by MOSAIC. The user can then go back to MOSAIC to resolve any inconsistencies that showed up.

D. MOSAIC

1. Introduction

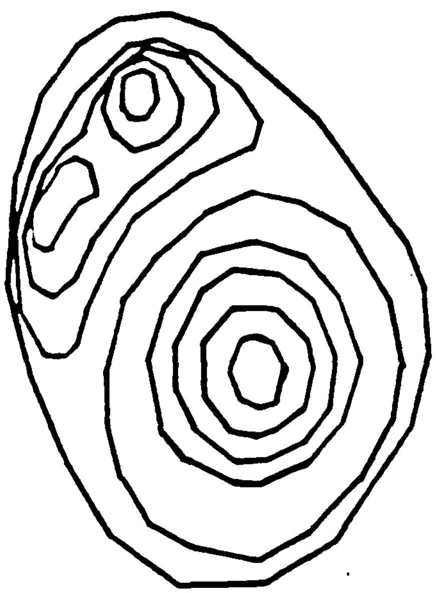
MOSAIC implements an algorithm for processing complex contour arrangements into polygonal element mosaics which are suitable for line drawing and continuous tone displays. The program does this by mapping adjacent contours onto the same unit square and, subject to ordering limitations, connecting nodes of one contour to their neighbors in the other contour so that the total length of the connecting lines is minimized. A user is able to interact and resolve highly ambiguous situations. Other key features of MOSAIC are node thinning and selective reduction of triangular pairs to quadrilaterals.

Figure 4.1 illustrates a surface which has been reconstructed using MOSAIC. Figures 4.1a and 4.1c show the original contour loops prior to processing, Figures 4.1b and 4.1d show the triangulation performed by MOSAIC.

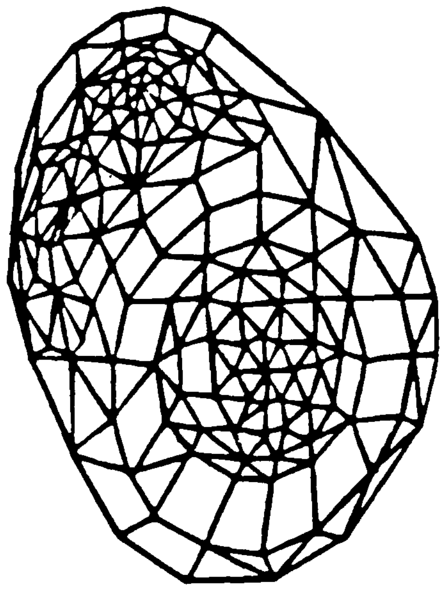
2. Triangulation Algorithm in MOSAIC

The triangulation algorithm implemented by MOSAIC assumes that the optimum arrangement of triangles produces the least sum total length of all connecting diagonals between levels. MOSAIC also requires that the loops to be triangulated are numbered in the same rotational direction and that the first nodes of each loop are closed. This is something, however, that the user need not worry about since MOSAIC automatically performs these preparations.

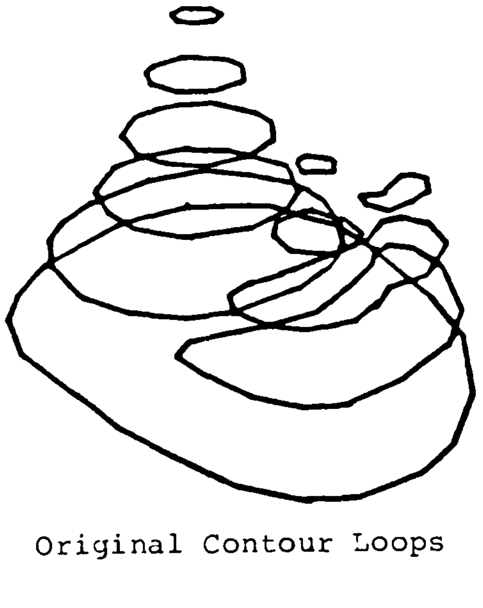
This triangulation algorithm is illustrated in Figures 4.2 and 4.3. Figure 4.2 depicts two loops that are to be triangulated. The graph shown in Figure 4.3a is a representation of these loops. Each intersection of lines can be thought of as defining a diagonal, such that P(i,j)



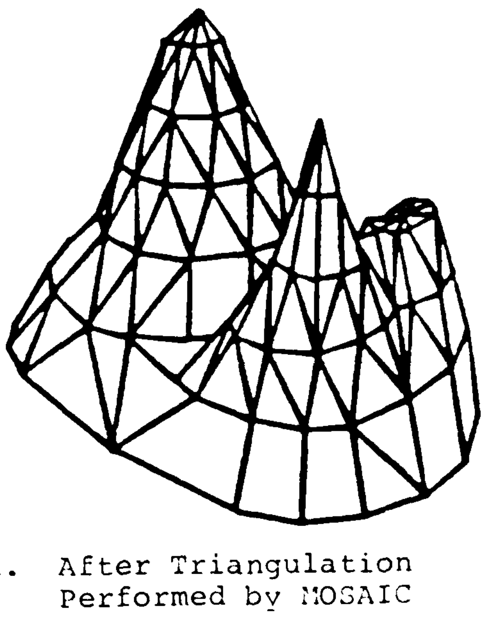
a. Original Contour Loops



b. After Triangulation Performed by MOSAIC

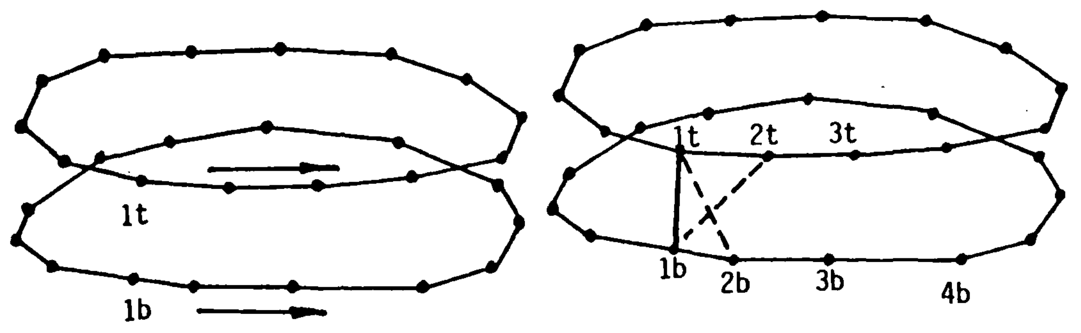


c. Original Contour Loops



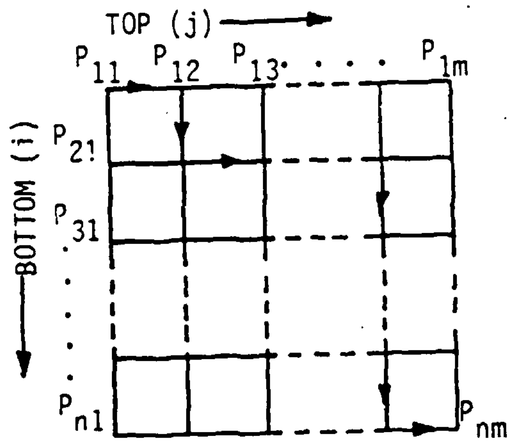
d. After Triangulation Performed by MOSAIC

Figure 4.1



(a) Contour Pair Prior To Processing (b) Beginning Of Triangulation

Figure 4.2



(a) Representation Of Contour Pair

$$\begin{bmatrix} D_{11} & D_{12} & D_{13} \cdots & D_{1m} \\ D_{21} & D_{22} & D_{23} \cdots & D_{2m} \\ D_{31} & D_{32} & D_{33} \cdots & D_{3m} \\ \vdots & \vdots & \vdots & \vdots \\ D_{n1} & D_{n2} & D_{n3} & D_{nm} \end{bmatrix}$$

(b) Total Distance Matrix

$$\begin{bmatrix} M_{11} & M_{12} & M_{13} \cdots & M_{1m} \\ M_{21} & M_{22} & M_{23} \cdots & M_{2m} \\ M_{31} & M_{32} & M_{33} \cdots & M_{3m} \\ \vdots & \vdots & \vdots & \vdots \\ M_{n1} & M_{n2} & M_{n3} & M_{nm} \end{bmatrix}$$

(c) Path Memory Matrix

Figure 4.3

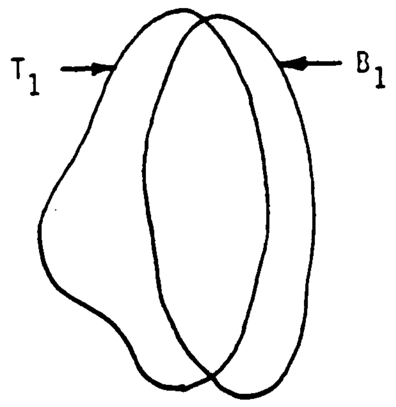
depicts the diagonal connecting node i on the bottom contour to node j on the top. Associated with the graph of Figure 4.3a are the two matrices shown in Figure 4.3b and Figure 4.3c. Both the total distance matrix and the path memory matrix coefficients have a direct correspondence with the points on the graph.

The total distance matrix is the sum of the length of the diagonals included in the shortest path. The path memory matrix is used to indicate the path that was used to attain the least total length in lacing from diagonal P(1,1) to P(i,j). M(i,j) equals one if the path including diagonal ib-jt also includes the diagonal (i-1)b-jt. If the path includes diagonal ib-(j-1)t instead of (i-1)b-jt, M(i,j) equals zero.

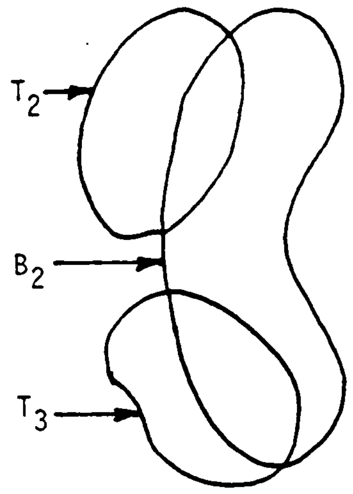
For best results when using this algorithm loop pairs should be of similar size and shape. The loop pairs should not be offset excessively. For this reason, MOSAIC automatically maps both loops onto the same unit square before triangulating.

3. Loop Connectivity and Branching

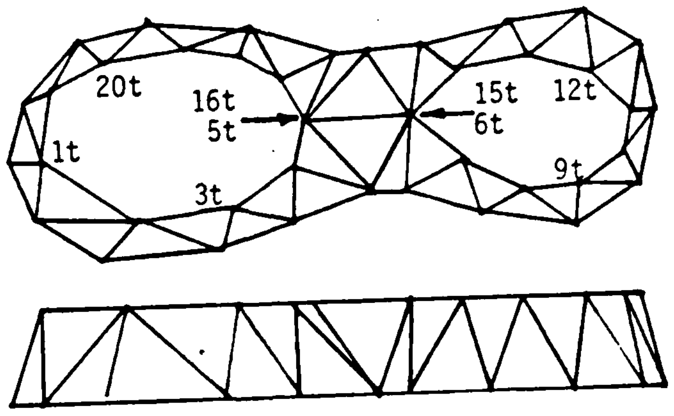
When more than one loop exists on adjacent levels potential problems may occur. To alleviate this problem MOSAIC allows the user to interact and specify which loops on the bottom layer connect to which loops on the top layer. As a default MOSAIC uses an overlap test to determine which loops to connect. See Figure 4.4 for an example. This is



a. Loops to be Connected



b. Loops to be connected



c. After Loops Connected

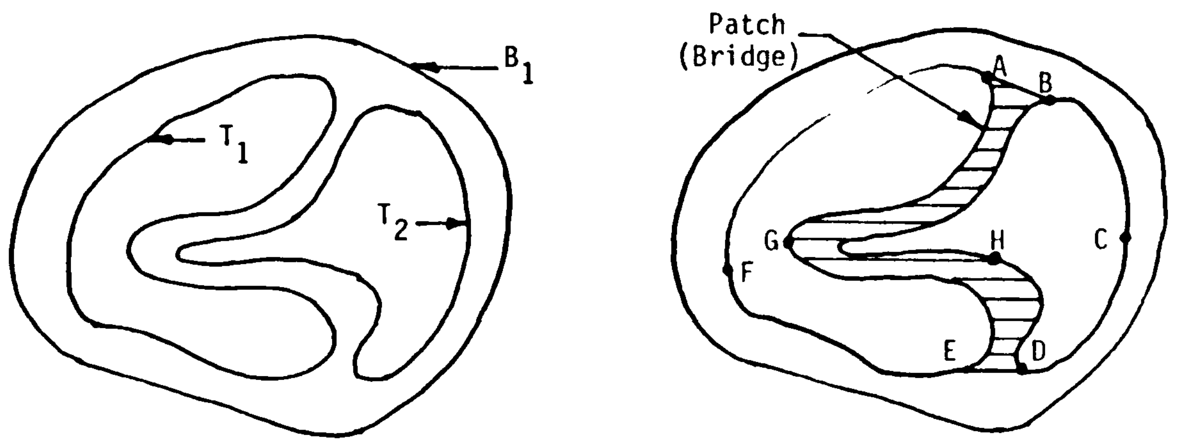
Figure 4.4

good for many cases but there are cases of connectivity that overlap will not resolve correctly. After the user has forced any desired loop connectivity he may return the control of the process back to the algorithm.

The other problem that sometimes occurs when there are more than one loop on a level is branching. The method used by MOSAIC treats all branches on a level as one continuous loop. It does this by locating the closest nodes between branches and then renumbering the branching loops so that they collectively appear to be one large loop that has been squeezed together over segments where the linkups occur. It then triangulates as before. If the user is unhappy with this branching he may force branching between given nodes of the loops. Figure 4.5 illustrates this concept. For the loops shown in Figure 4.5a, a flat patch between the two co-planar loops would be a better solution than the automatic routine. To accomplish this, the user must input the node numbers associated with nodes A, E, B, and D. MOSAIC will then automatically form the patch by lacing the two loops together from nodes A to E and B to D. The remaining portions of the top loops are then renumbered and triangulation carried in a normal manner.

4. <u>Other User Options</u>

A large data base of contour information sometimes becomes difficult to use. If this is the case, it may be desirable to eliminate unessential nodes. MOSAIC has four



(a) Difficult Branching (b) Possible Solution

Figure 4.5

different ways in which it can thin out a data base. It is able to eliminate nodes that are too close together. It can combine line segments between which the change of direction is minimal. It can skip entire contour levels as the data is read in. And finally, it can join pairs of adjacent triangles to form a quadrilateral if the angle by which they are out of plane is sufficiently small.

All of these options require input by the user regarding to the acceptable distance between nodes, amount of direction change between elements, Z level spacing, and quadrilateral warp.

5. Command Structure

MOSAIC uses a multi-level prompt command structure for user interaction. There are a total of four levels and one pseudo level. At each level the user is either asked a specific question or given a prompt (">" for level 1, ">>" for level 2, ">>>" for level 3, ">>>>" for level 4, and ">>>>>" for level 5). At any time the command HELP may be entered to obtain a complete listing of all available commands for the current level. "Escape" to a lower (numbered) level is accomplished by entering a carriage return.

In addition to the commands unique to each level there are eight global commands which may be entered at command levels one through four. These commands are DEVIce, EXIT, HELP, MAP, PART, TOTAls, WARP, and WRITe. For these commands and all other commands only the first four letters need to be entered by the user. The MOVIE.BYU User's Manual [Ref. 11] contains a complete description of commands, and a brief description of each command is contained in Appendix A. Table I is a listing of all available commands.

6. Installation of MOSAIC

Using Chapter 8 of the MOVIE.BYU User's Manual [Ref. 11], MOSAIC was installed on a VAX 11/780 computer running under a VMS operating system. The version of VMS was 4.2. The language used was FORTRAN 4.3.

Due to the large size of the contour data base to be used with MOSAIC, it was necessary to increase most of the parameters associated with MOSAIC. It was also necessary to increase the dimension(s) of the corresponding arrays associated with these parameters.

Table II lists all parameters changed along with their original values and current values. Table III lists all arrays whose dimensions were changed along with their dependent parameter, original dimension, and current dimension.

TABLE I

Commands Available In MOSAIC

COMMAND	LEVEL
DEVIce	global
EXIT	global
HELP	global
MAP	global
PART	global
TOTAls	global
WARP	global
WRITe	global
CLIP	one
CLOCkwise	one
CONVert	one
MAKE	one
RANGe	one
READ	one
THINning	one
AUTOmatic	two
CAP	two
EDIT	two
EDIT-ADD node	three
EDIT-DELEte node	three
EDIT-LOOP	three
EDIT-MOVE node	three
INSPect	two
MANUal	two
MANUal-AUTOmatic	three
MANUal-BRIDge	three
MANUal-CLEAr	three
MANUal-DRAW	three
MANUal-GUIDe	three
GUIDe-AUTOmatic	four
GUIDe-DENSity	four
GUIDe-ERASe	four
GUIDe-RENUmber	four
GUIDe-TRIAngulate	four
MANUal-INCLude	three
MANUal-NEXT	three
POST	two

TABLE II

Changes To MOSAIC Parameters

PARAMETER	EXPLANATION	ORIGINAL VALUE	CURRENT VALUE
MLP	maximum number of total loops	100	500
MNLPL	maximum number of loops per level	5	120
MNNPL	maximum number of nodes per level	100	900
MNODES	maximum number of nodes	1007	5000
MNPLP	maximum number of nodes per loop	50	500
NBNMAX	maximum number of loops displayed	20	150
NPTMAX	maximum number of elements	1007	3000
NVMAX	maximum number of nodes being triangulated on adjacent levels	120	5000

TABLE III

Changes to MOSAIC Arrays

ARRAY	DIMENSION PARAMETER	SUBROUTINES USED IN
DTOT	NVMAX	ATRIAN, BKTRAK
P2	NVMAX	ATRIAN, BKTRAK, INIMAP
V	NVMAX	ATRIAN, AUTO, BKTRAK, DFINV, DRAWV, FPATCH, GUIDE, INIMAP, LABLP, LINK, MAKV, MANUAL, MAPP REORDR
TEMP	NVMAX	REORDR
FLAG	MLP	AUTO, CONECT, DFINV, FPATCH, LABLP, LINK, LNKMAN, LPLST, MAKV MANUAL, SELECT
EX	MLP	BLOCK, CAP, CLIPIT, CONECT, DFINV DRAWLP, EDIT, GEOMWR, LABLP, LNKMAN,LPLST, MAKV, MANUAL, PROCES, THNOUT
P1	MLP	BLOCK, CAP, CLIPIT, CONECT, DFINV DRAWLP, EDIT, GEOMWR, LABLP, LNKMAN, LPLST, MAKV, MANUAL, PROCES, THNOUT
LIST	NBNMAX	DRAWLP, EDIT, LABLP, LPLST
NIPL	MNLPL	ATRIAN
LPSTK	MNLPL	AUTO, CONECT, FGINV, FPATCH, LABLP, LINK, LNKMAN, LPLST, MAKV MANUAL, SELECT
O	MNLPL	AUTO, DFINV, LINK, LNKMAN, MAKV
LBN	MNLPL	BKTRAK, DFINV
NPL	MNLPL	CLIPIT, GEOMRD, PROCES, THNOUT
C	MNLPL	CONECT, FPATCH, LPLST, MAKV, MANUAL, SELECT
CP	MNLPL	DFINV
D	MNLPL	DFINV
DT	MNLPL	DFINV
ORDER	MNLPL	DFINV
LCFLG	MNLPL	DFINV
BRG	MNLPL	DFINV, FPATCH, LINK, LNKMAN
NCF	MNLPL	MANUAL, SELECT

TABLE III (continued)

Changes Made to MOSAIC Arrays

ARRAY	DIMENSION PARAMETER	SUBROUTINES USED IN
BRANCH	MNLPL	SELECT
STACK	MNLPL	SELECT
FPNT	MNPLP	CLIPIT
NCP	MNPLP	CLIPIT
TEMP	MNPLP	CLIPIT
P	MNODES	ATRIAN, BKTRAK, CAP, CLIPIT, CLKWSE, CROSS, DFINV, GEOMWR, INIMAX, LIST, MAPP, MOVE, PREPLT THNOUT
P3	MNNPL	CLKWSE, GEOMRD, MAKE, THNOUT
IP	NPTMAX	ATRIAN, BKTRAK, CAP, DRAWEL, GEOMWR

V. EXPERIMENTAL RESULTS

A. PRELIMINARY WORK

After installing MOSAIC, the first question to be answered was what images to use for the reconstruction process. The choice was quickly narrowed down to compressed images with either a compression ratio of 30 or 40. The reason for this was to have an image that had easily noticeable differences from the original image. It was finally decided to use a compression factor of 40. For reasons to be discussed later this was a good choice. The digitized image of Figure 5.1 is the basis for the remainder of this study.

Next it became necessary to convert the format of the digitized compressed image into a format suitable for MOSAIC. MOSAIC requires the data to be in a single file that contains the number of nodes in a loop, the Z coordinate of the contour level, and then the (X, Y) coordinates of the nodes in the loop. This information is required for each loop. A "0" at the end of the last loop terminates the file. The Z coordinate corresponds to the gray level of the contour.

The information needed to construct the input file for MOSAIC is contained in two separate files. These two files are the Contour Record and the Layer Record which were

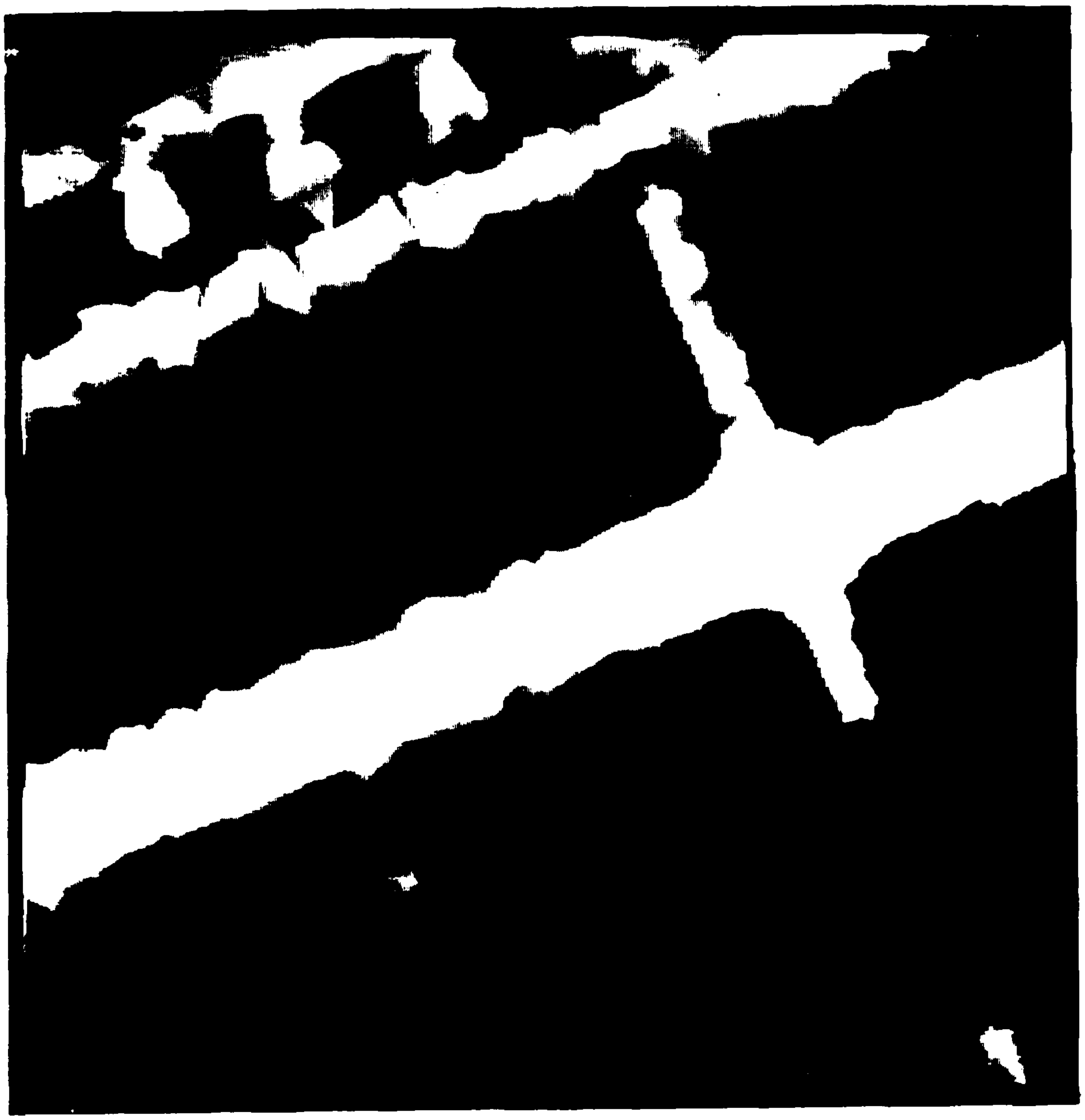


Figure 5.1 Original Compressed Image

discussed in Chapter 2. To accomplish this conversion the program CONVBYU.FOR was written in FORTRAN. The source code is contained in Appendix B. CONVBYU.FOR accomplishes its purpose by opening and reading the two files, Contour Record and Layer Record, and then merging the information needed by MOSAIC into one file.

B. RECONSTRUCTION USING MOSAIC

1. Closed Contours

With MOSAIC installed and the contour data needed by MOSAIC converted to the correct format, MOSAIC is executed and the contours are read in. Using the INSPect command all levels were displayed individually. The first thing noticed was the fact that all contours were closed, even those that were originally open contours. It turns out that MOSAIC requires all contours to be closed and, if a contour is not closed, MOSAIC closes the contour by connecting the first and last nodes.

By closing contours in this arbitrary manner, contours that were created sometimes crossed over themselves. This could prove confusing to the triangulation algorithm in MOSAIC and result in a poor reconstruction.

After further study of the individual levels, it appeared that instead of closing each loop a better choice might be to combine several groups of loops into one loop per group.

A quick and easy wasy to simulate the combining of a pair of loops and achieving the desired appearance of one loop in triangulation is to use the BRIDge command. Unfortunately due to the manner in which the loops were closed by MOSAIC (contour crossover) this was not a viable option.

The only other option was to use the EDIT commands. By adding, deleting, and/or moving nodes, two contours could be merged into one contour. This was an easy but time consuming task. It is necessary to duplicate the nodes of loop one on loop two and then go back and delete the nodes of loop one.

The choice of which loops to combine has no set of rules. It comes down to a decision made by the observer. This decision is based on a study of the loops on a level and the visual relationship of the loops on adjacent levels. When contours are fairly large and have a well-defined shape it is an easy matter to determine which loops to combine. It is a difficult endeavor when dealing with small loops that have no well-defined shape.

Figures 5.2 through 5.17 show the original contour representations of the eight levels of the digitized image and the modified contour representations obtained through adding and deleting nodes as discussed above.

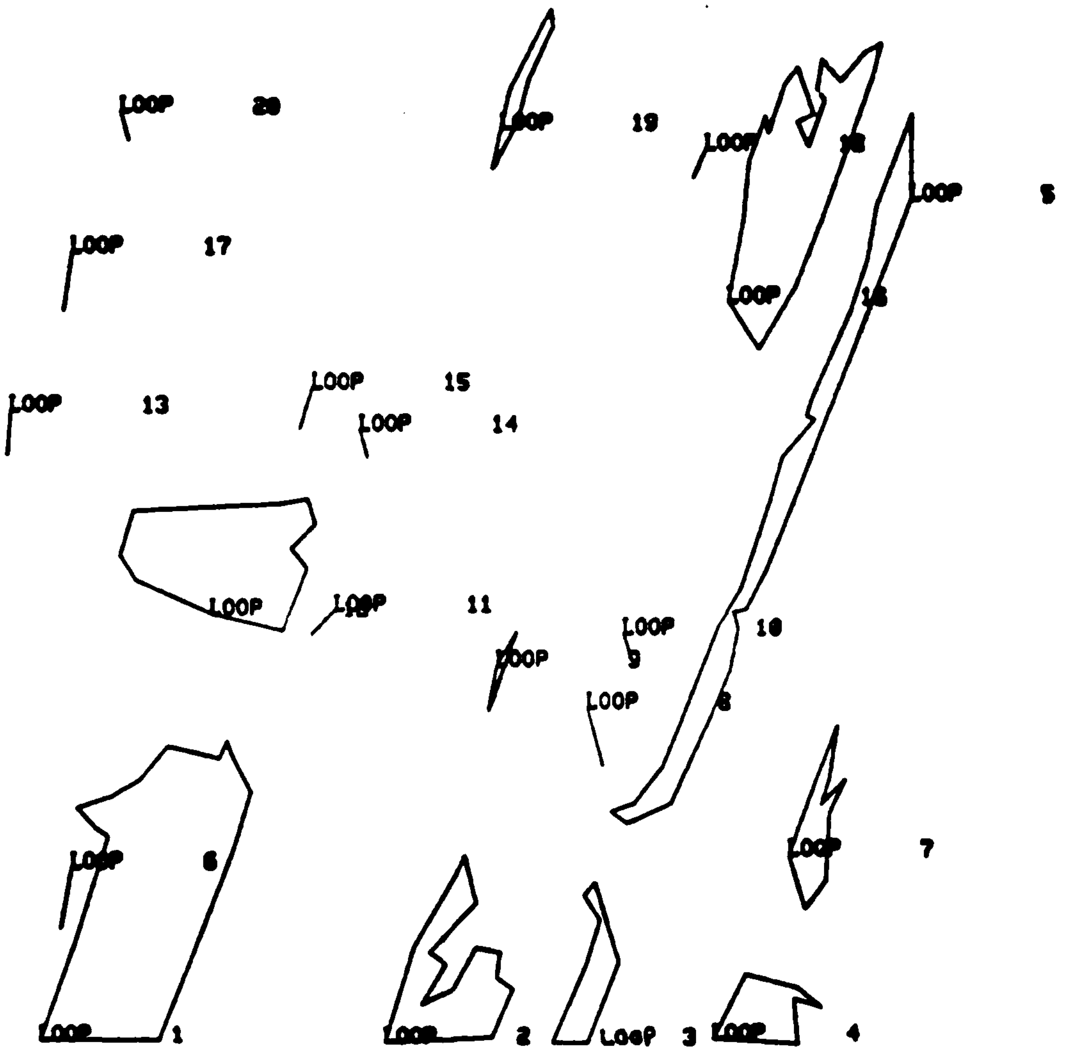


Figure 5.2   Level 1 (original)

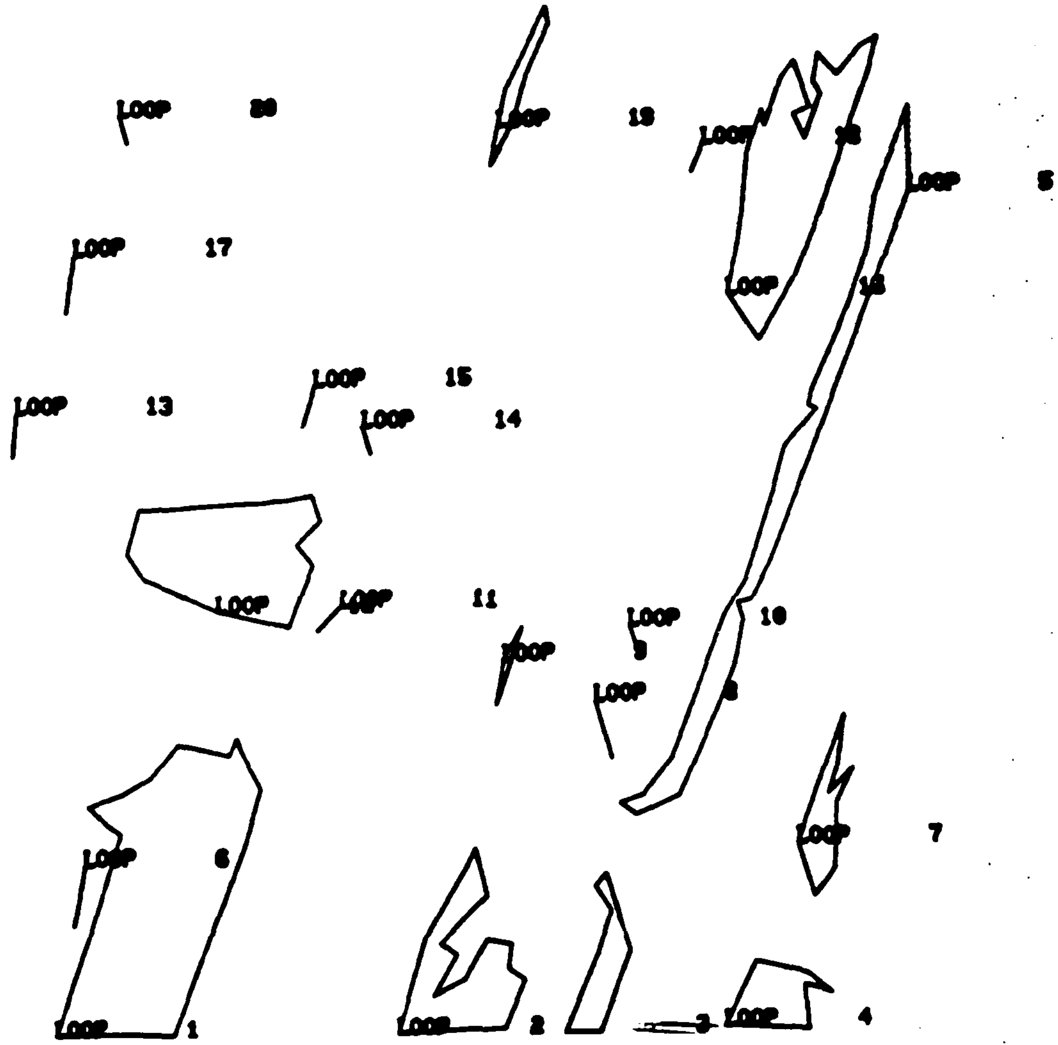


Figure 5.3 Level 1 (modified)

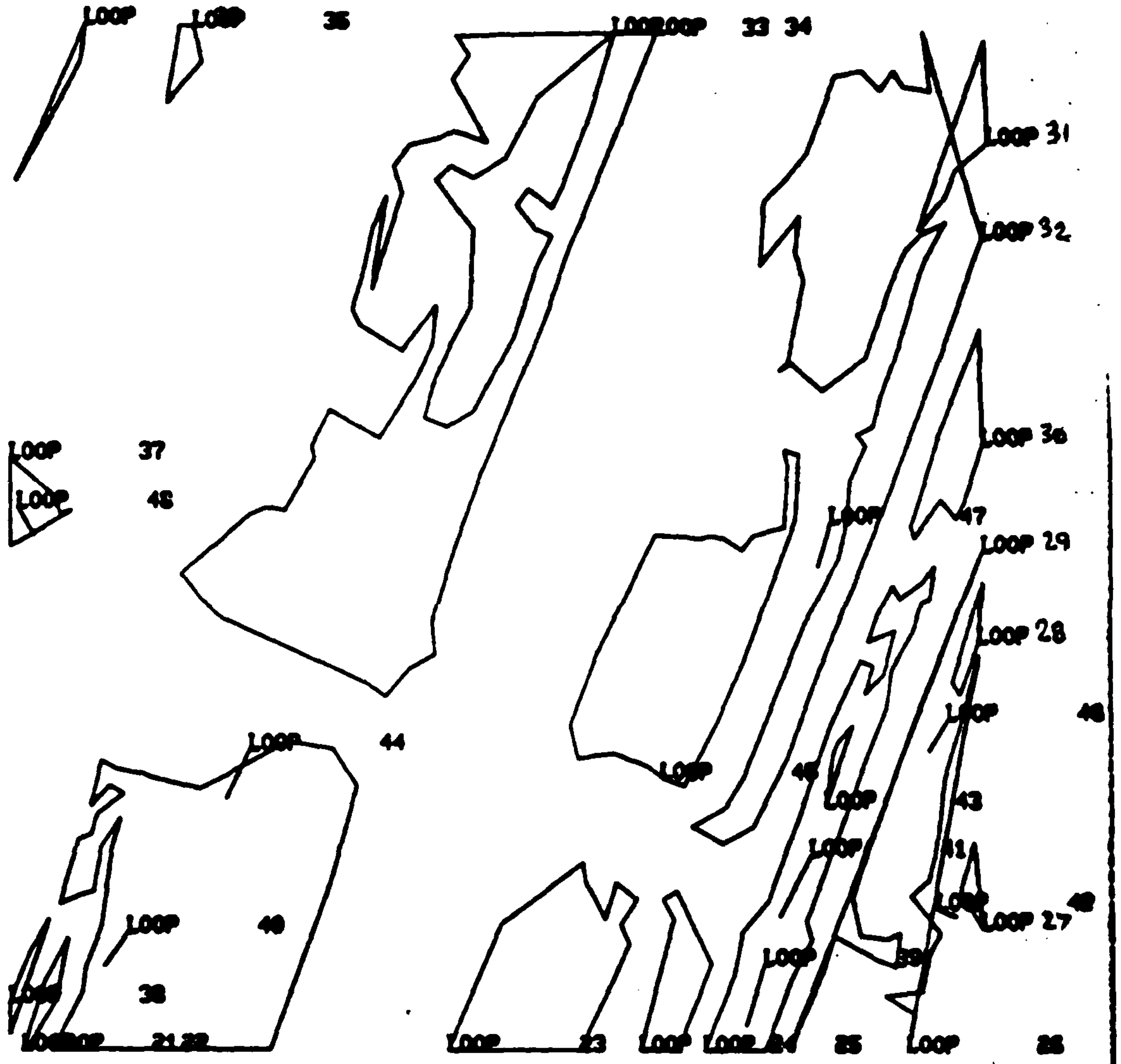


Figure 5.4 Level 2 (original)

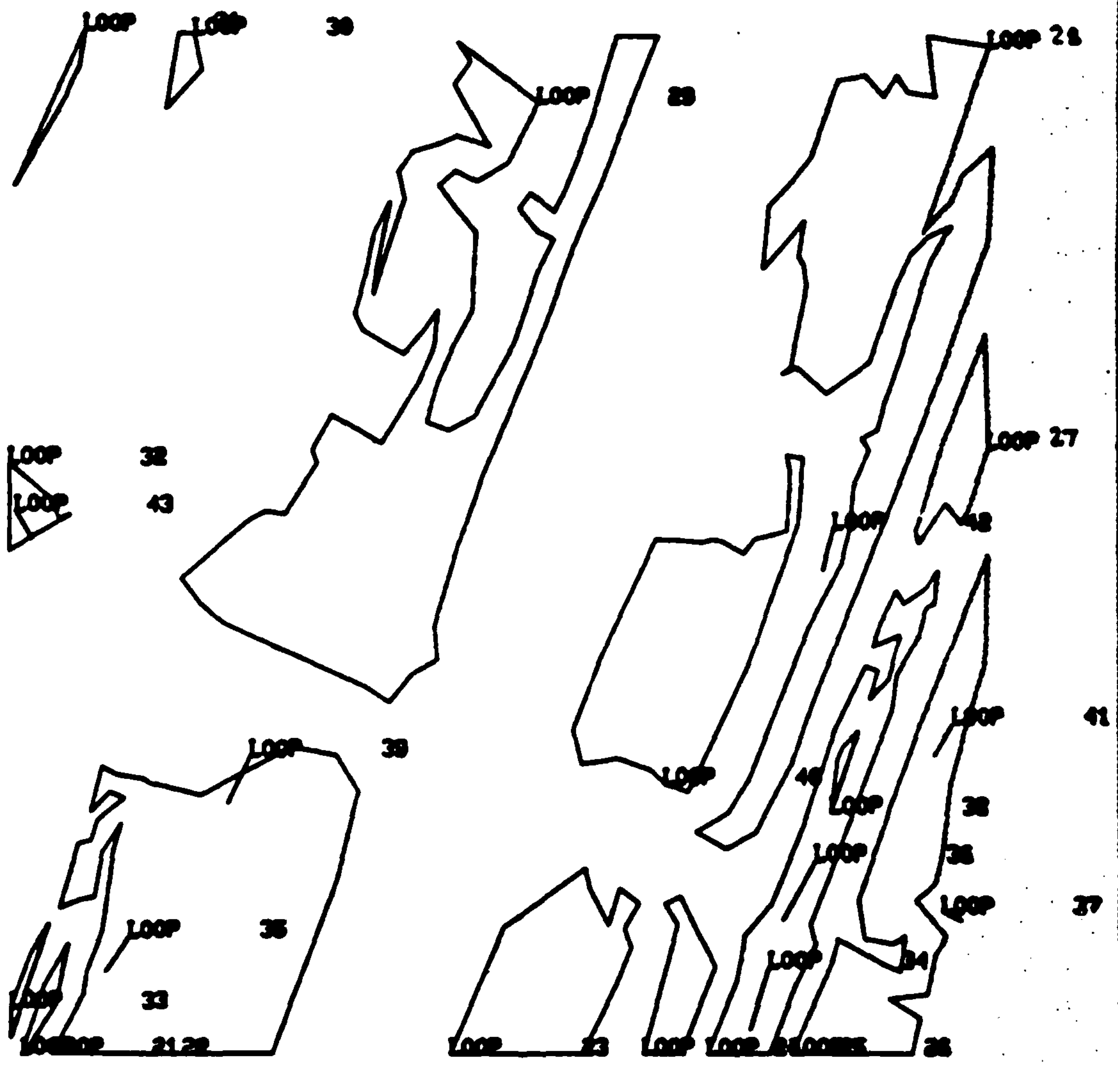


Figure 5.5 Level 2 (modified)

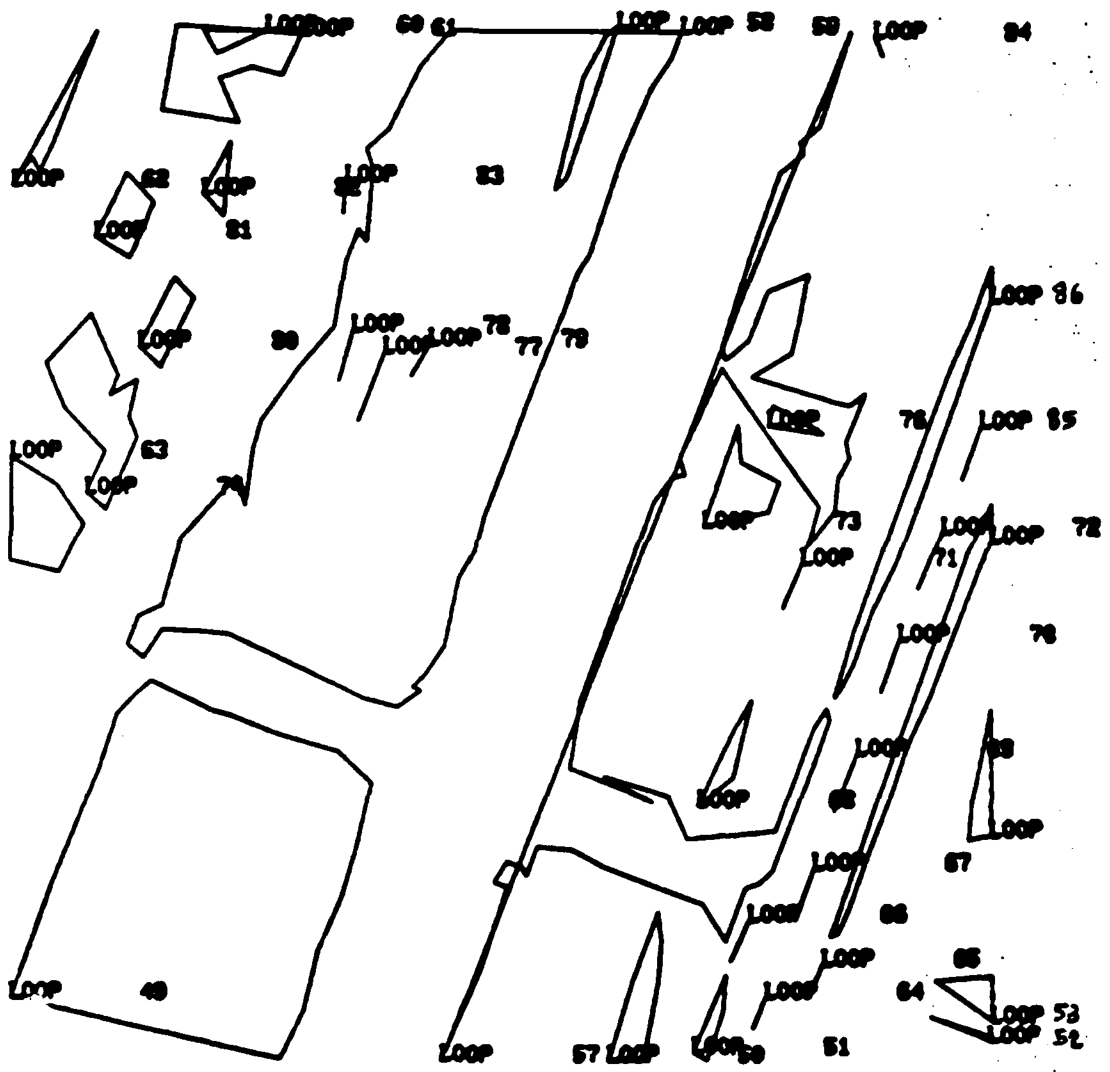


Figure 5.6 Level 3 (original)

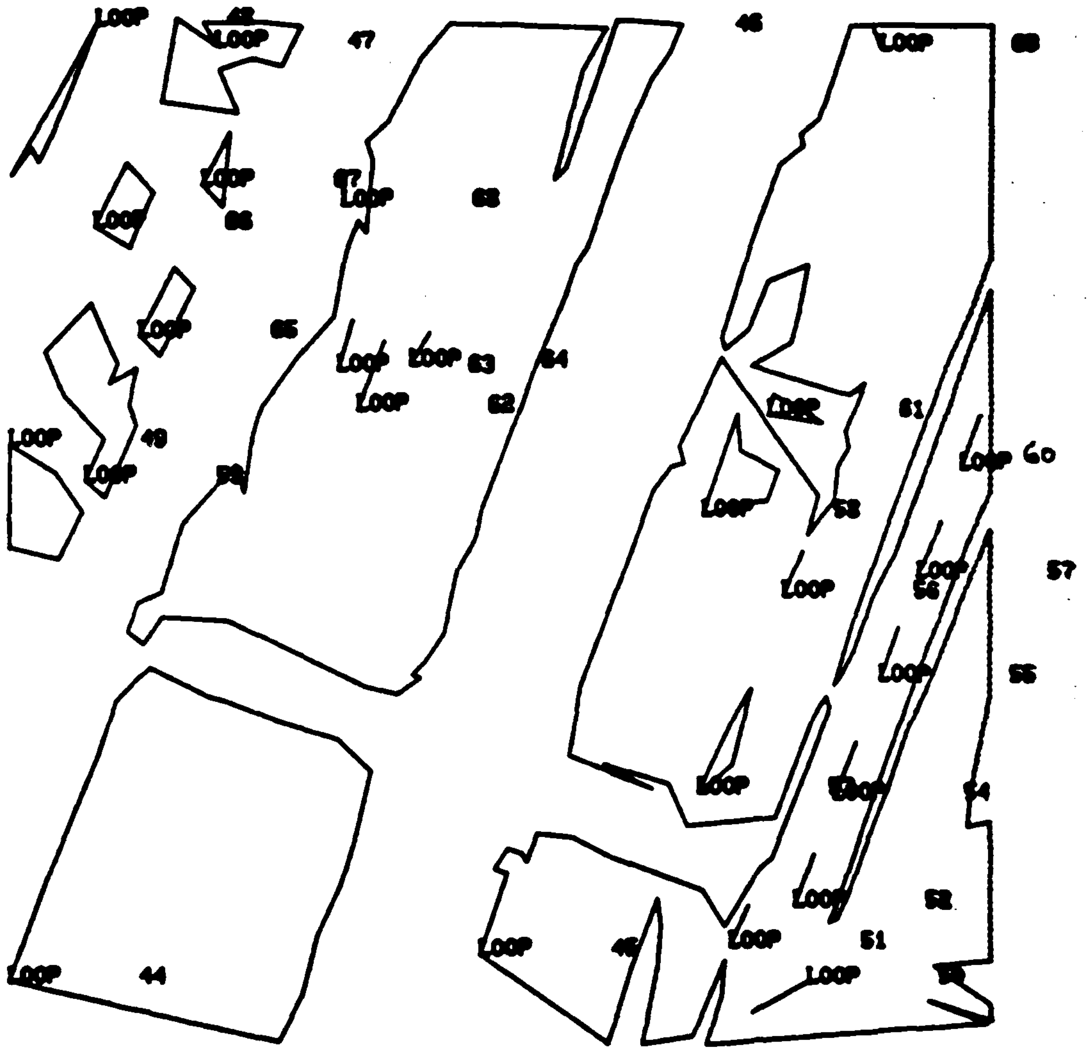


Figure 5.7 Level 3 (modified)

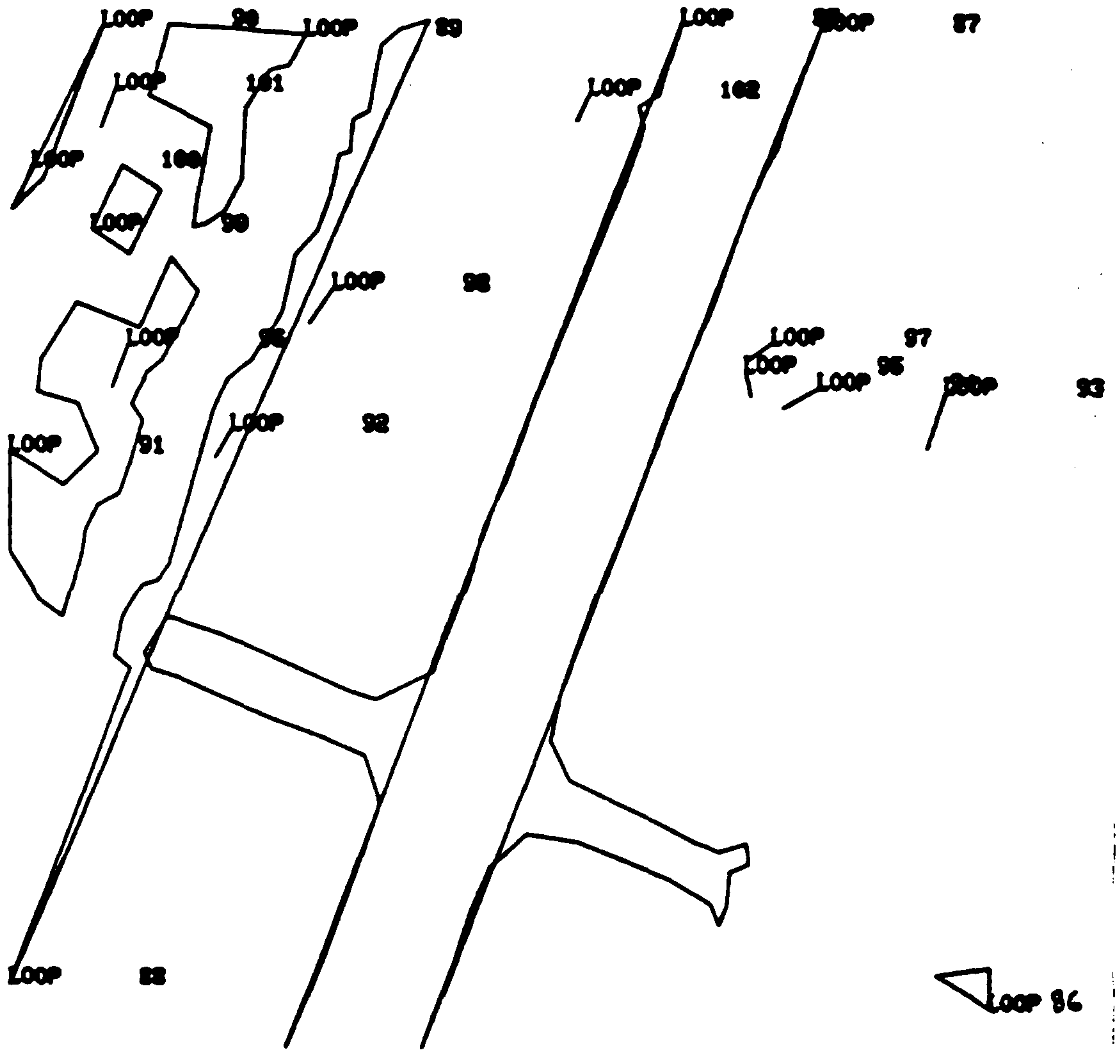


Figure 5.8 Level 4 (original)

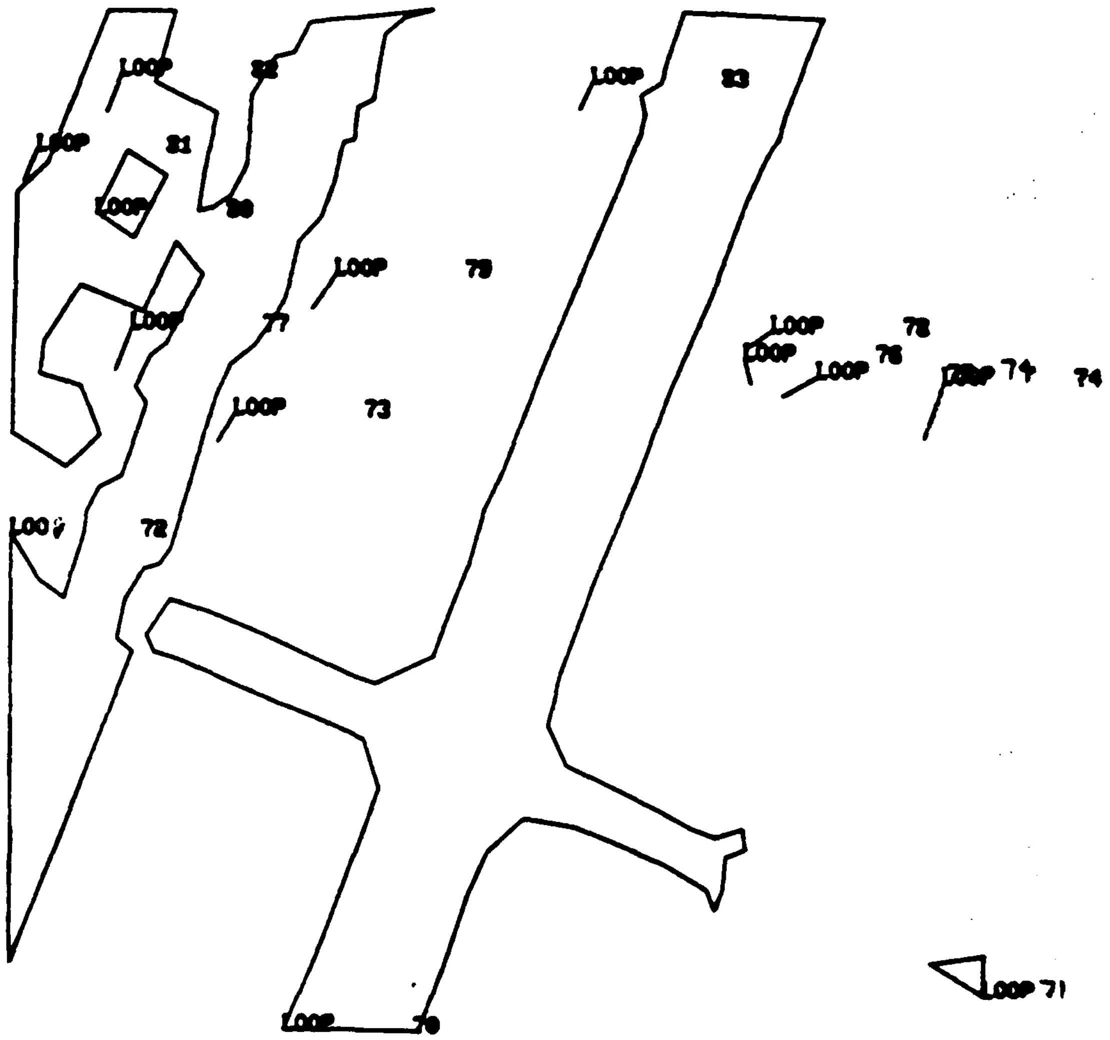


Figure 5.9 Level 4 (modified)

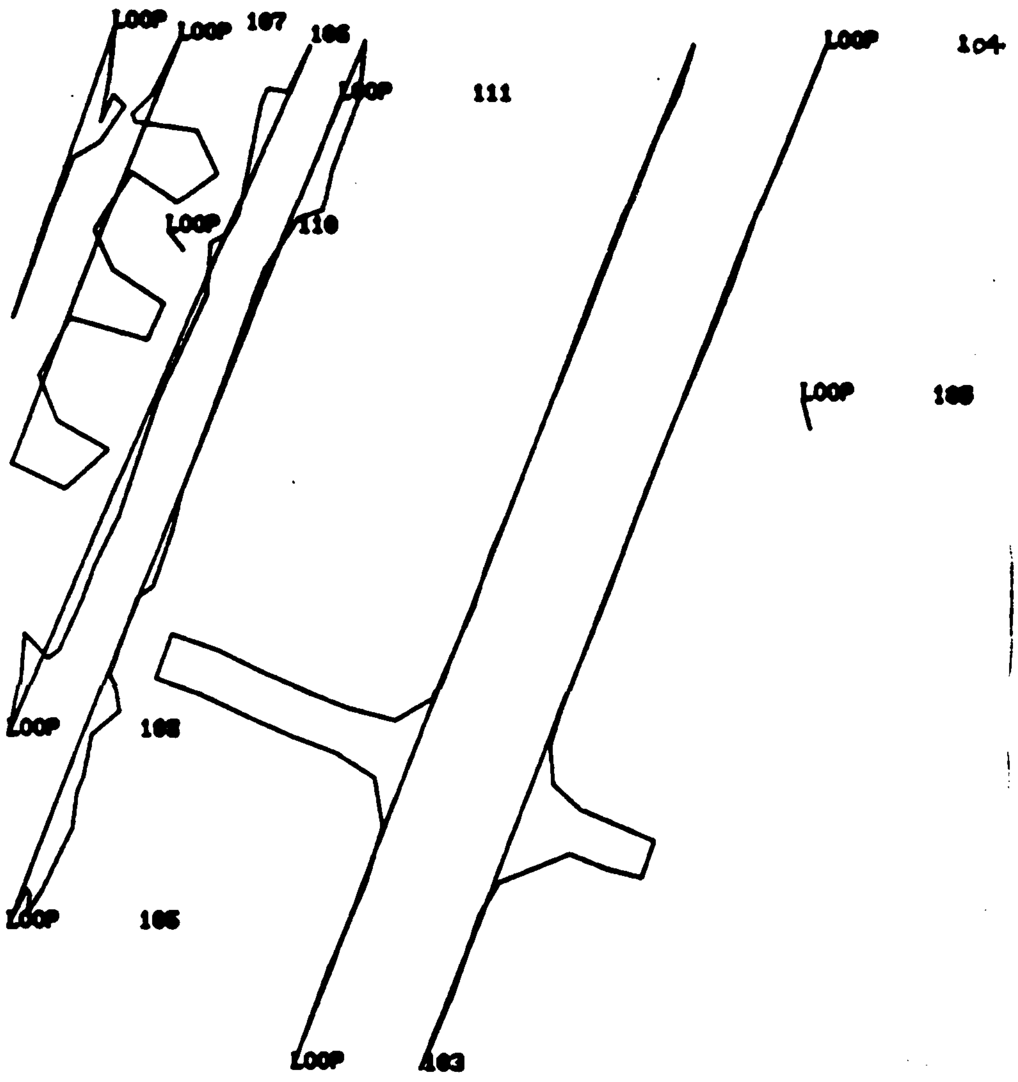


Figure 5.10   Level 5 (original)

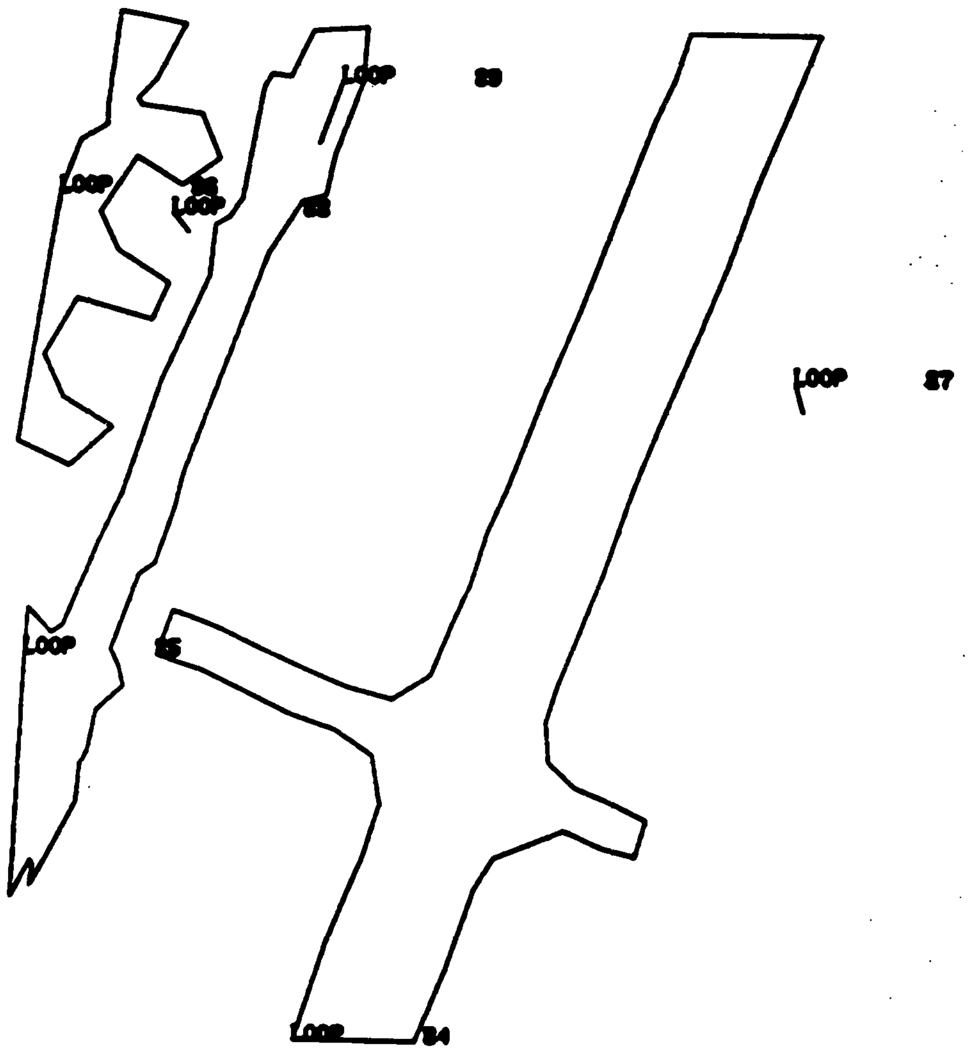


Figure 5.11   Level 5 (modified)

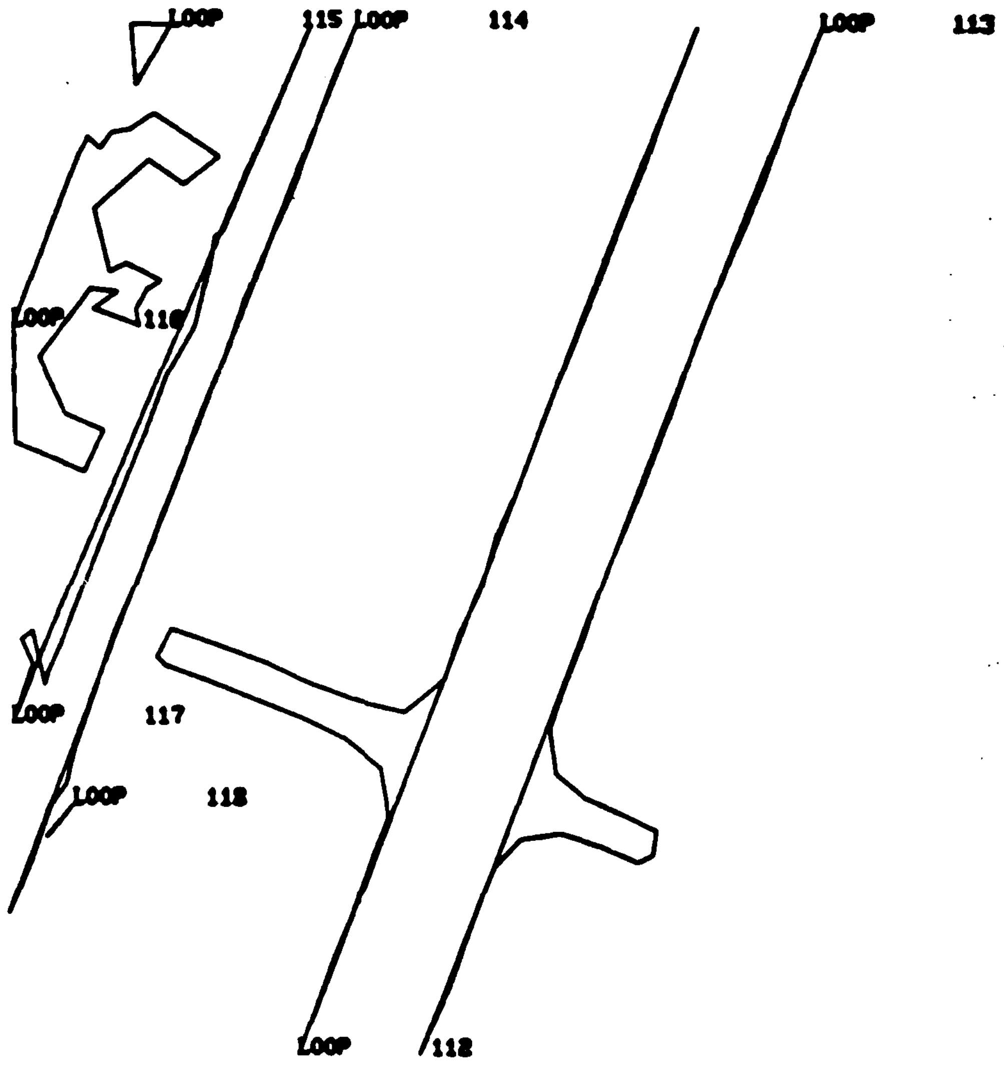


Figure 5.12   Level 6 (original)

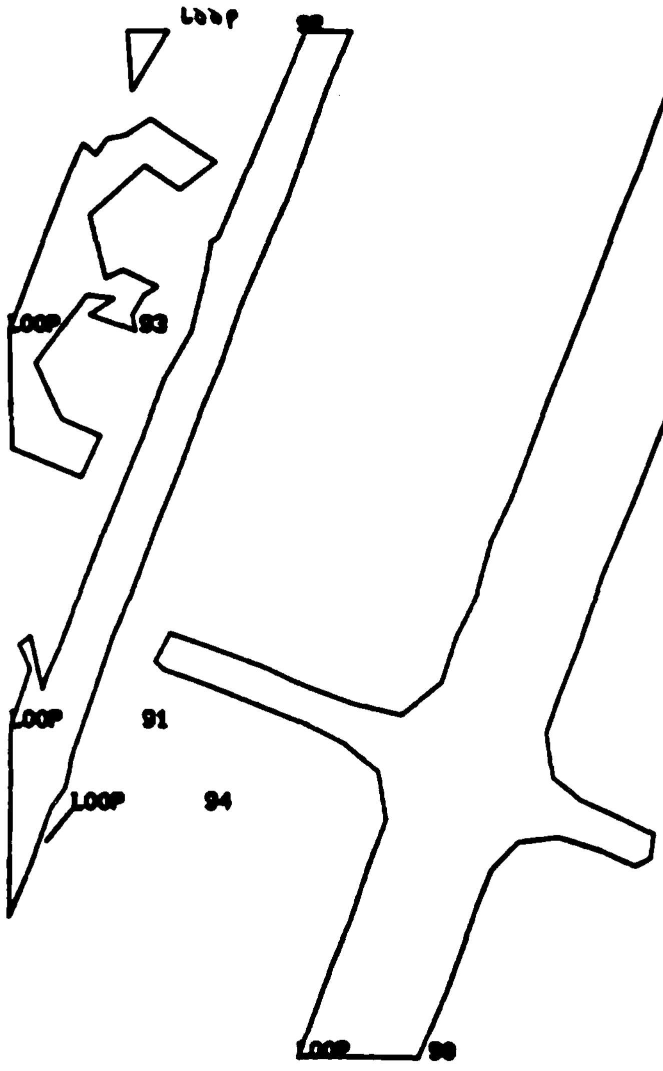


Figure 5.13 Level 6 (modified)

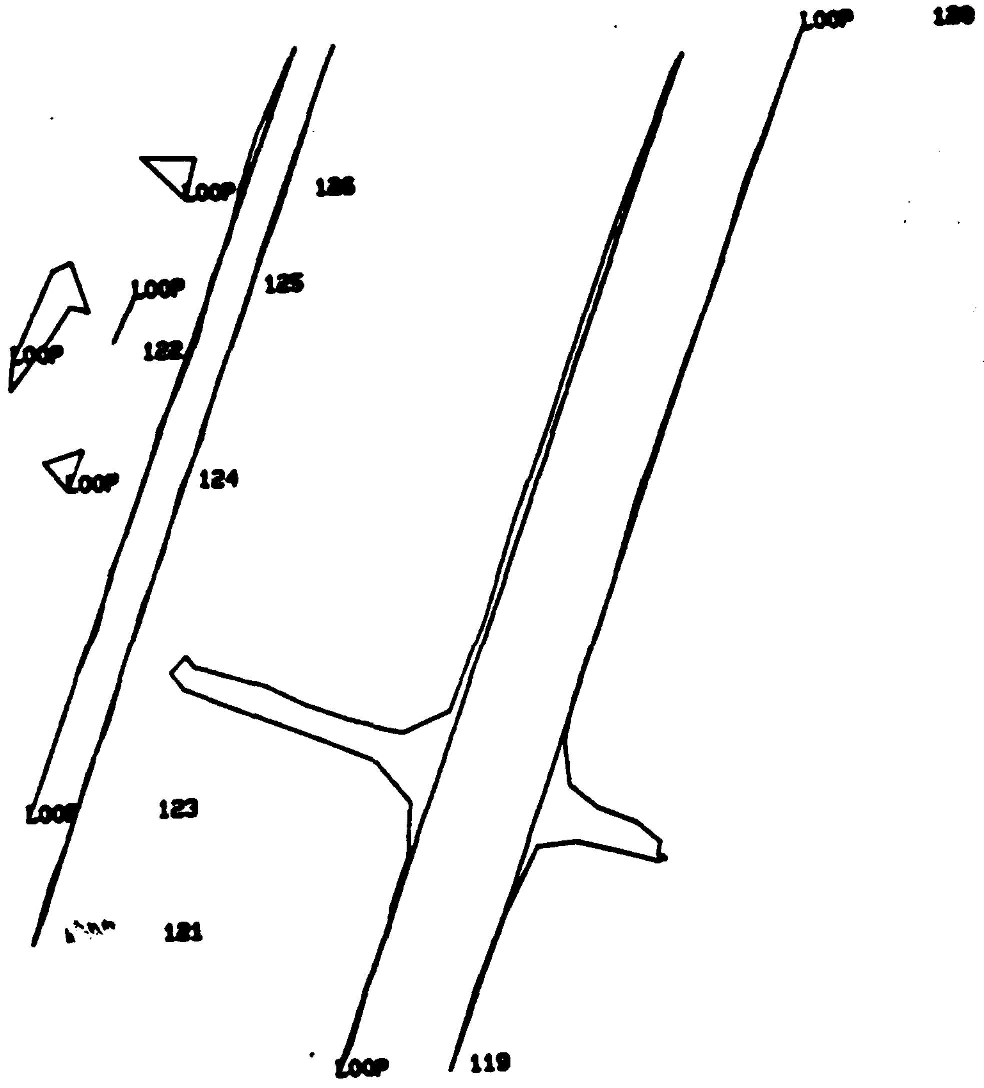


Figure 5.14   Level 7 (original)

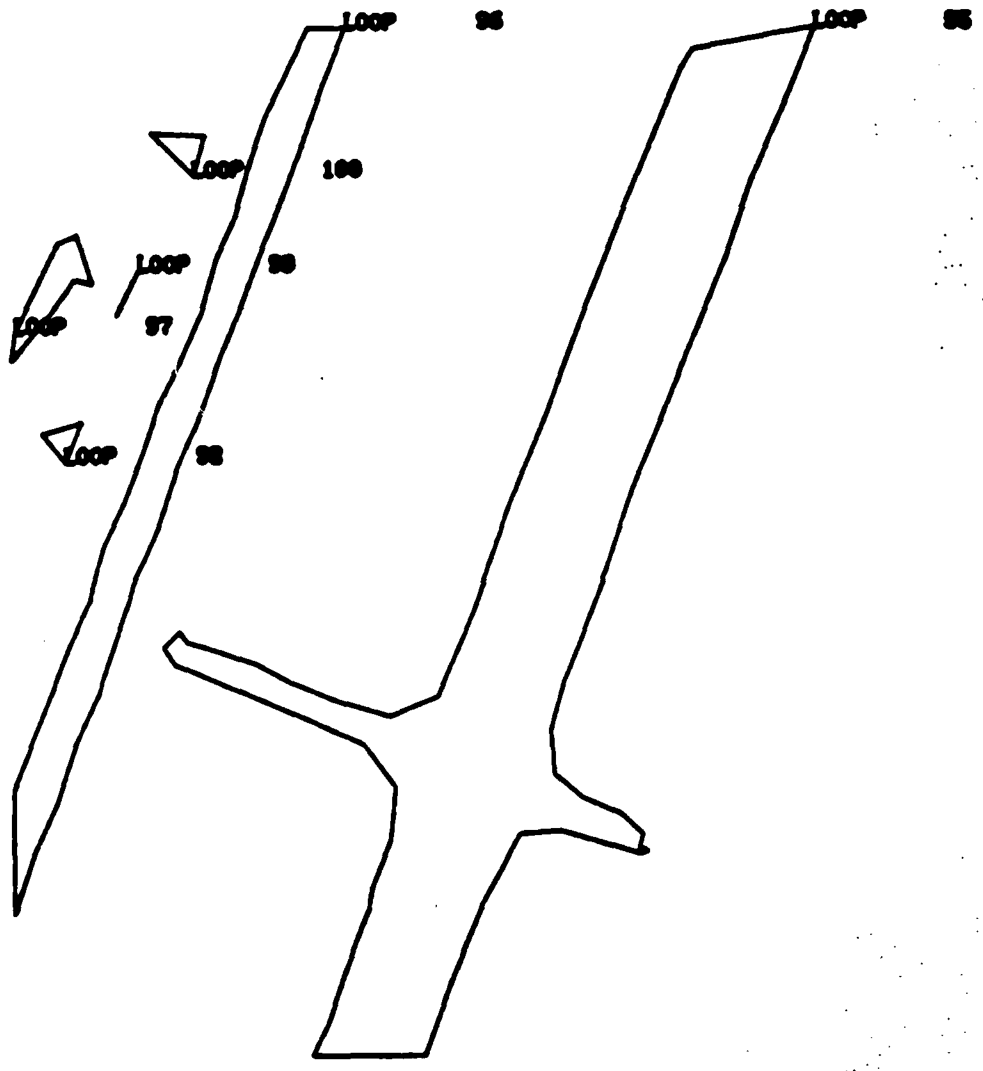


Figure 5.15 Level 7 (modified)

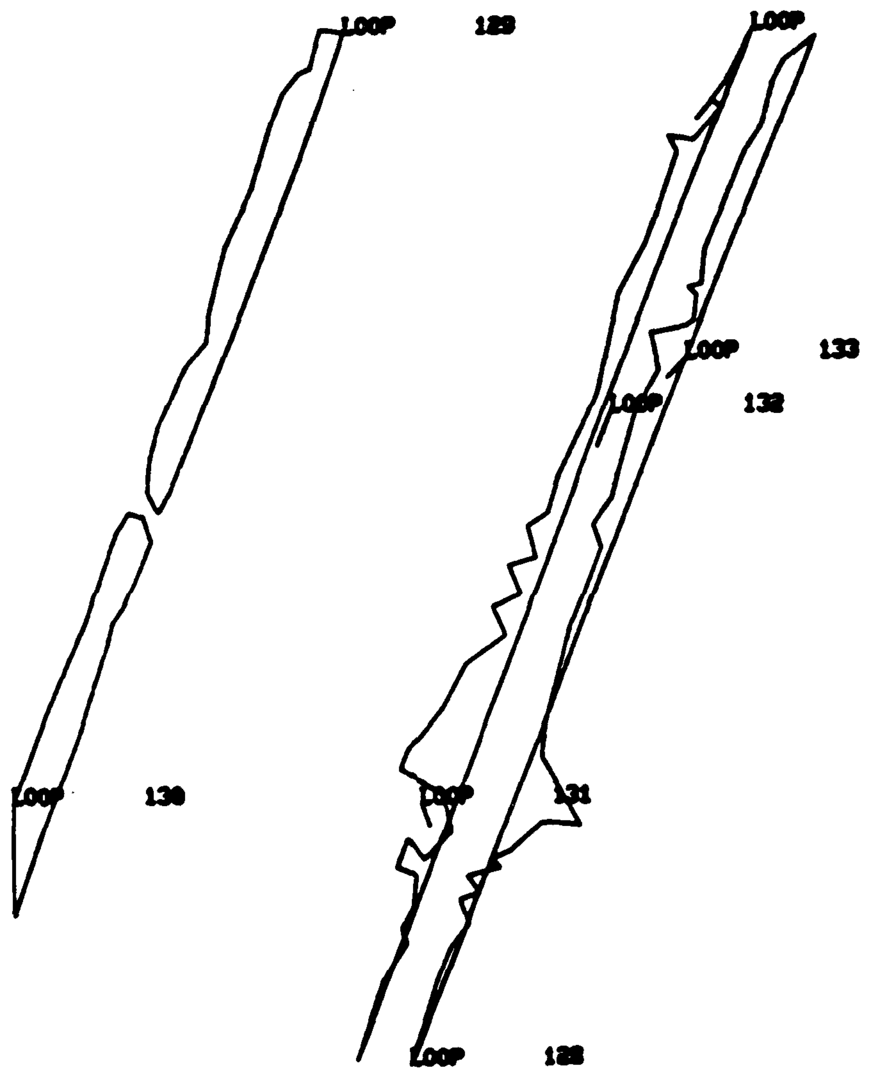


Figure 5.16   Level 8 (original)

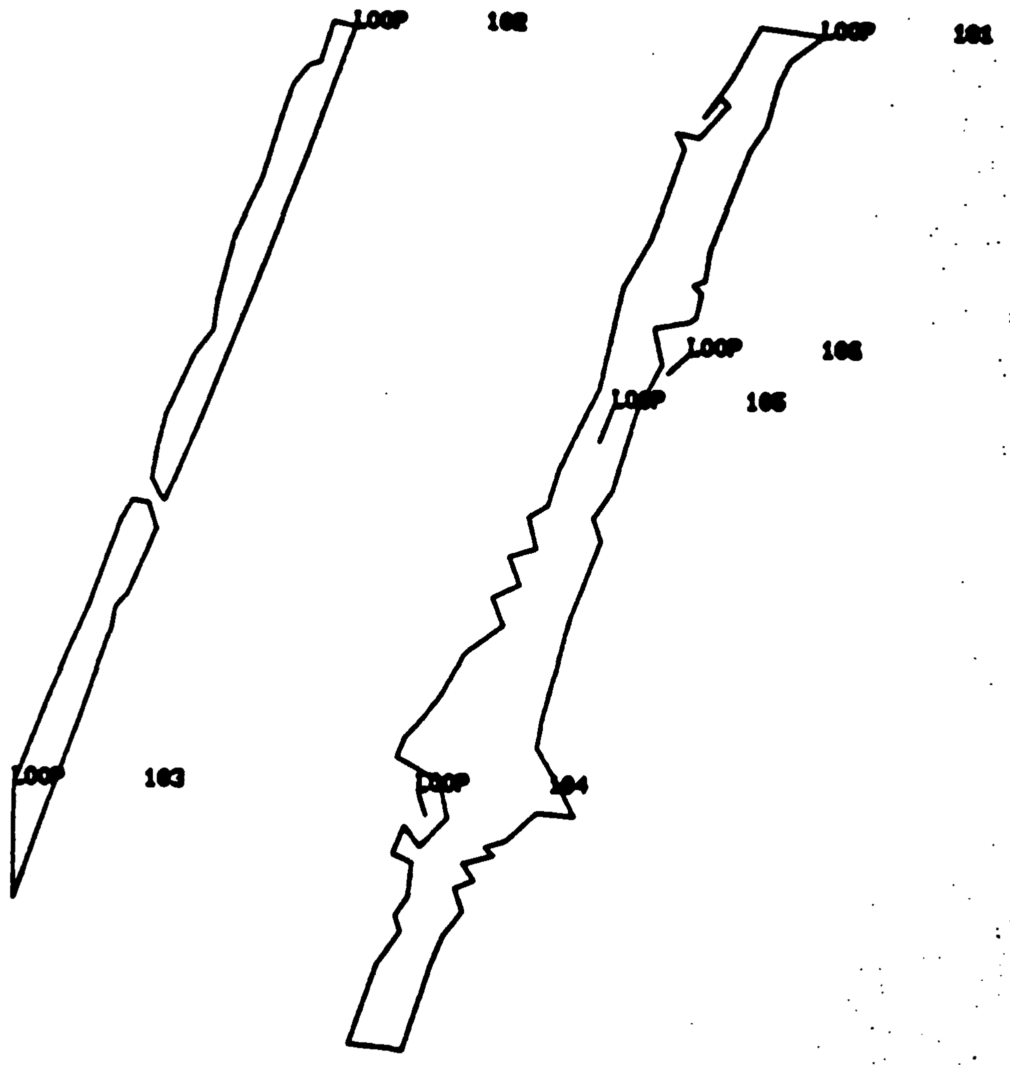


Figure 5.17 Level 8 (modified)

2. Loop Connectivity

With all contour levels redrawn, decisions must be made on which loops to connect to which loops. When left to proceed automatically MOSAIC uses an overlap criteria to decide which loops to connect. Any lower loops that branch onto any upper loops will be triangulated together.

This works fine for contours that have not suffered any degradation of shape due to noise, transmission loss, or compression of the original image. When a contour's shape has been degraded, it sometimes results in a loop branching onto another loop. MOSAIC will then attempt to connect these loops together.

The alternative is to specify specifically which loop(s) to connect to which loop(s). Once again this is a subjective decision based on an examination of adjacent levels and formation of a mental three dimensional picture of the levels. With well-defined contours this is not difficult. Other contours present difficulties. Additionally, this stage of triangulation relies on accurate choices being made on how the contours were closed.

Once all levels have been triangulated DISPLAY, another module of MOVIE.BYU, may be used to view on a display device a three dimensional representation of all levels along with the connections made. If this shows unsatisfactory results, the user may then go back into MOSAIC and respecify which loops to connect.

Although this process is for the most part subjective, they are a few guidelines to aid the user in deciding which loops to connect.

Triangulating loops on two levels can be compared to "sewing". With this in mind one would expect the triangulation of loops to be concentrated along the edges of the contours. The interior of the inner loop should be relatively empty. When the triangulation causes lines to be drawn back and forth across the contours unevenly it is an indication that something is not right. The cause could be that loops were not closed correctly or that the wrong loops were connected. A user may not always be able to resolve this inconsistency. This results in a choice of what loops not to connect to other loops.

Once the user is satisfied with the triangulation process, MOSAIC allows the user to save this data to a file. The file contains the (X, Y, Z) coordinates of all nodes along with the node numbers of the vertices of all triangles formed.

Figures 5.18 through 5.24 show the triangulation between each pair of adjacent levels that was finally settled on.

3. Reconstruction of the Image

The reconstruction process of the image consists of two steps. The first step is the triangulation process discussed above. The second step involves evaluating an

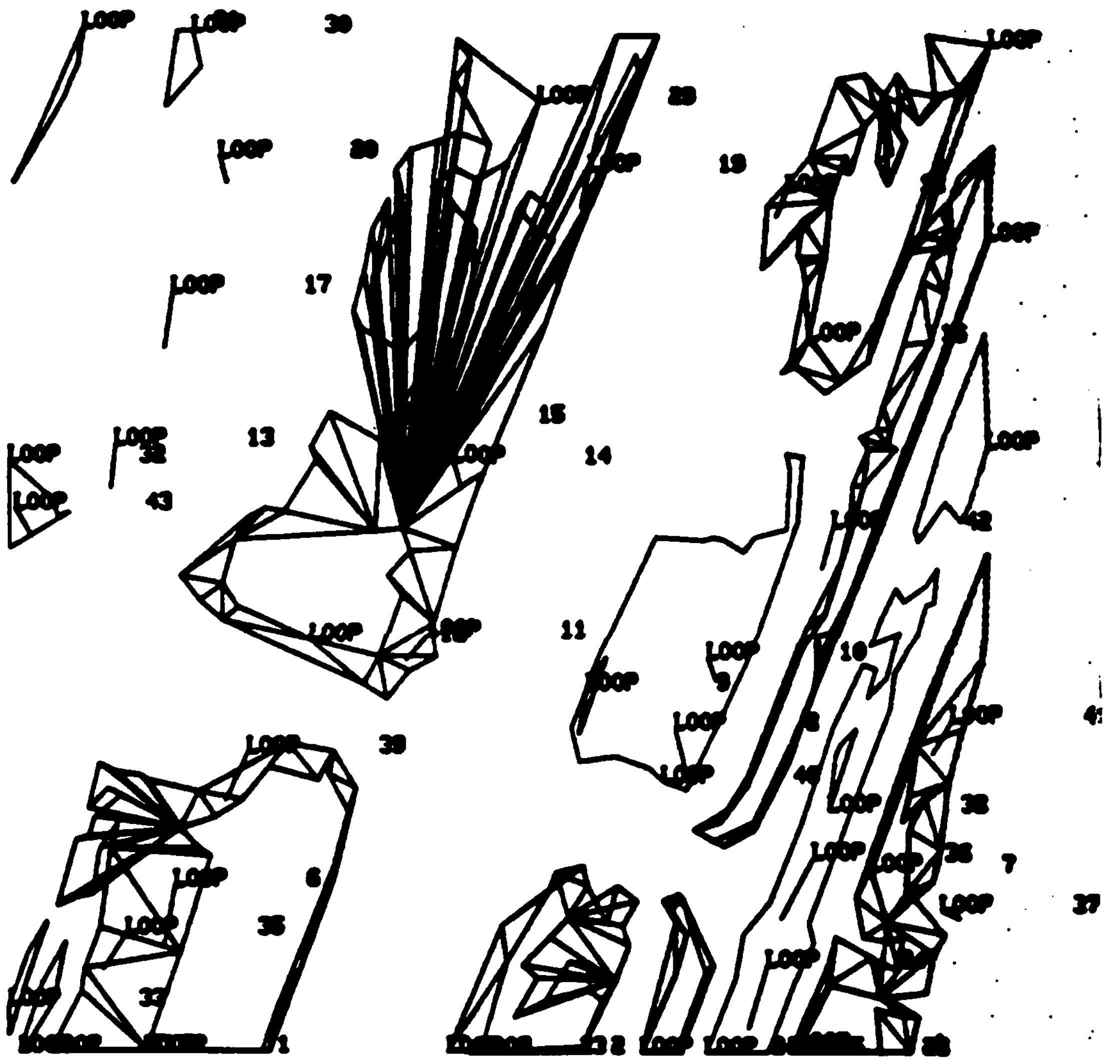


Figure 5.18 Triangulation - Loops 1-2

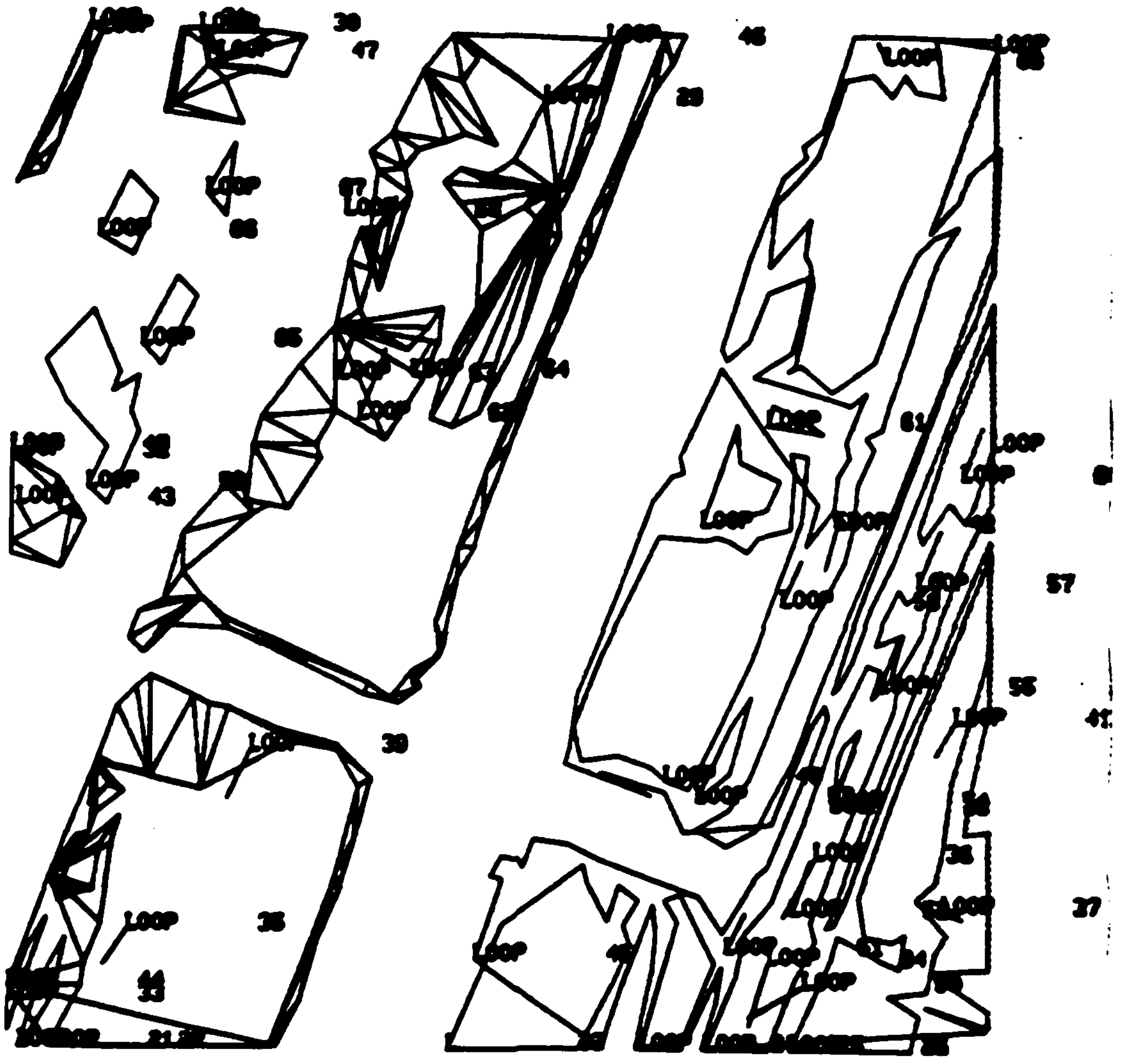


Figure 5.19 Triangulation - Loops 2-3

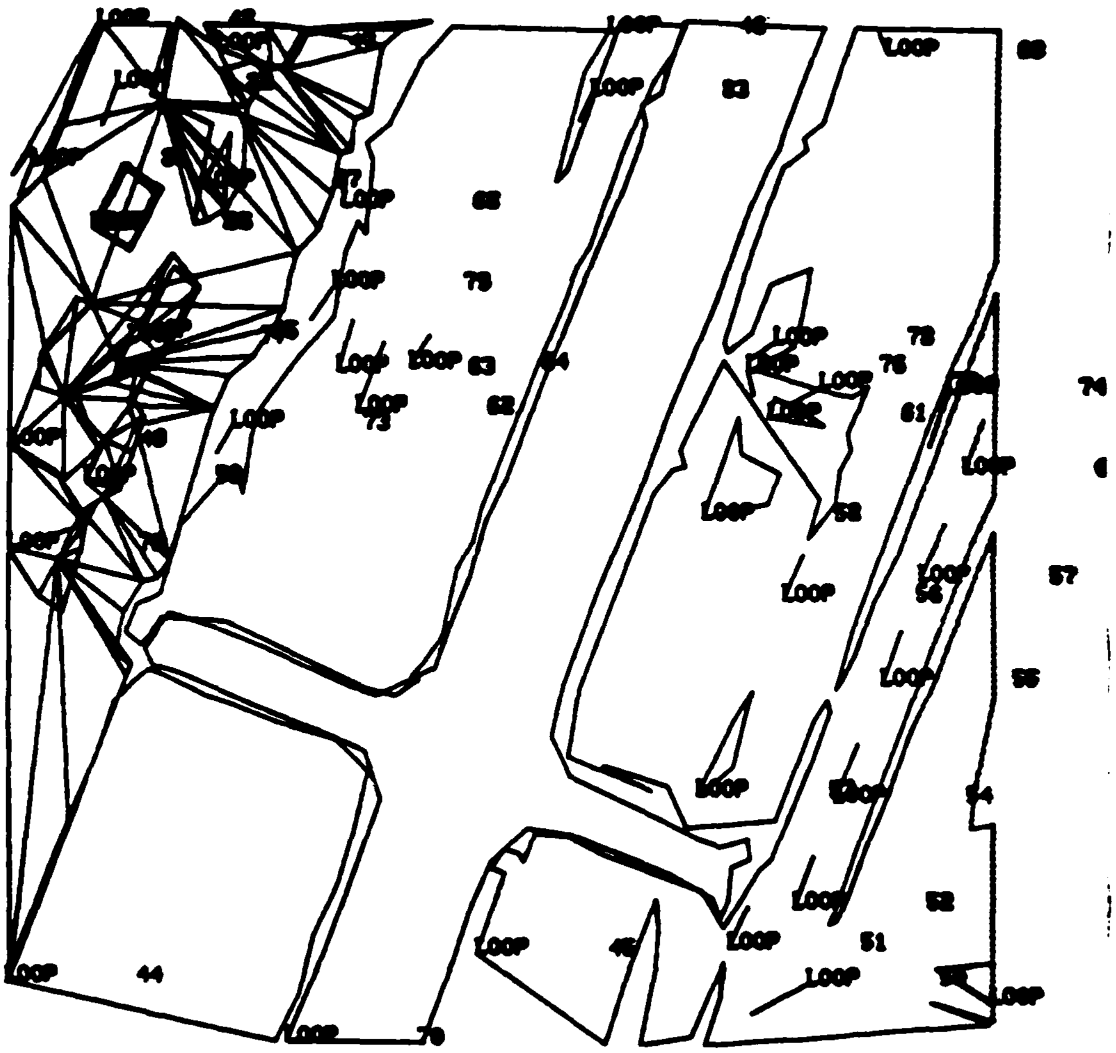


Figure 5.20   Triangulation - Loops 3-4

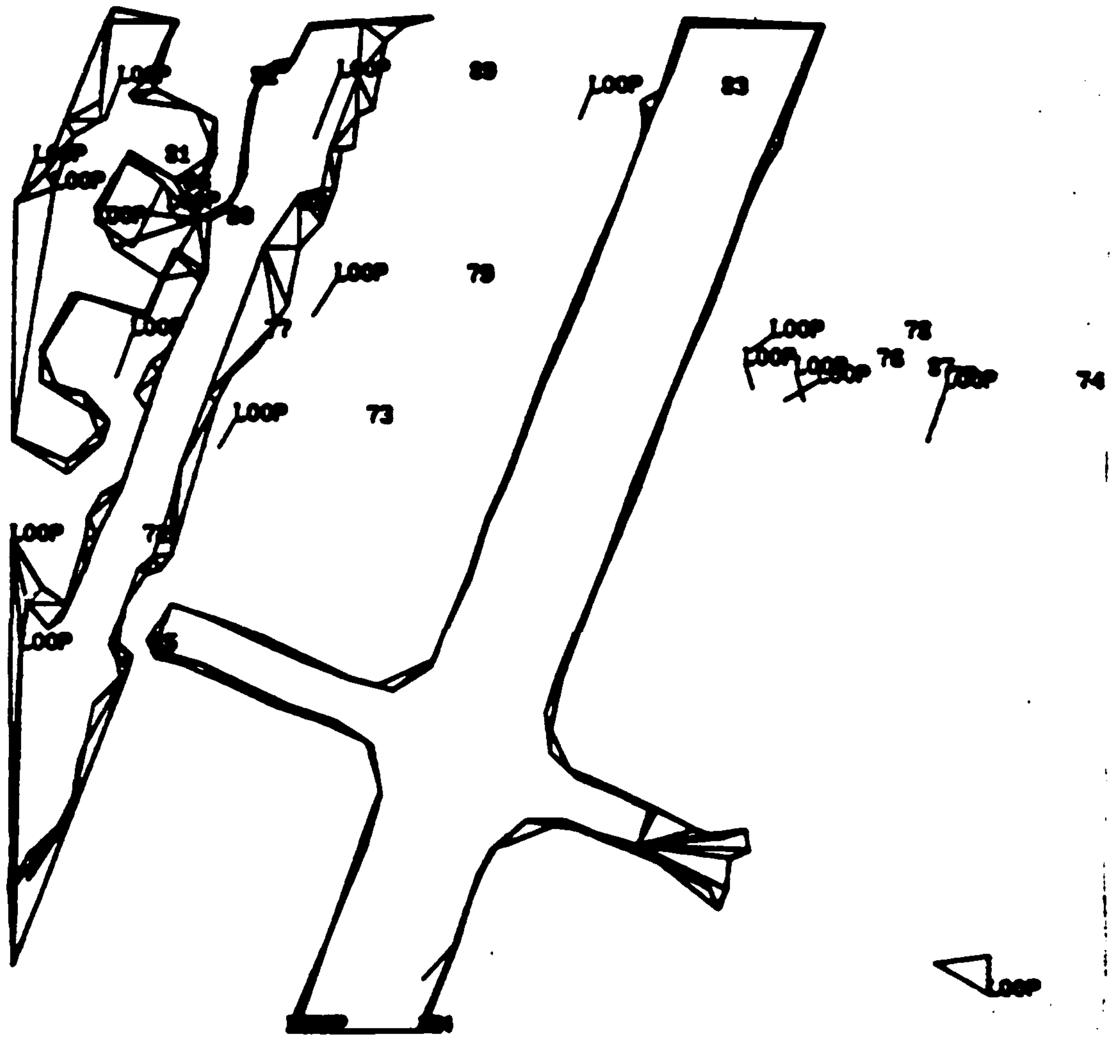


Figure 5.21   Triangulation - Loops 4-5

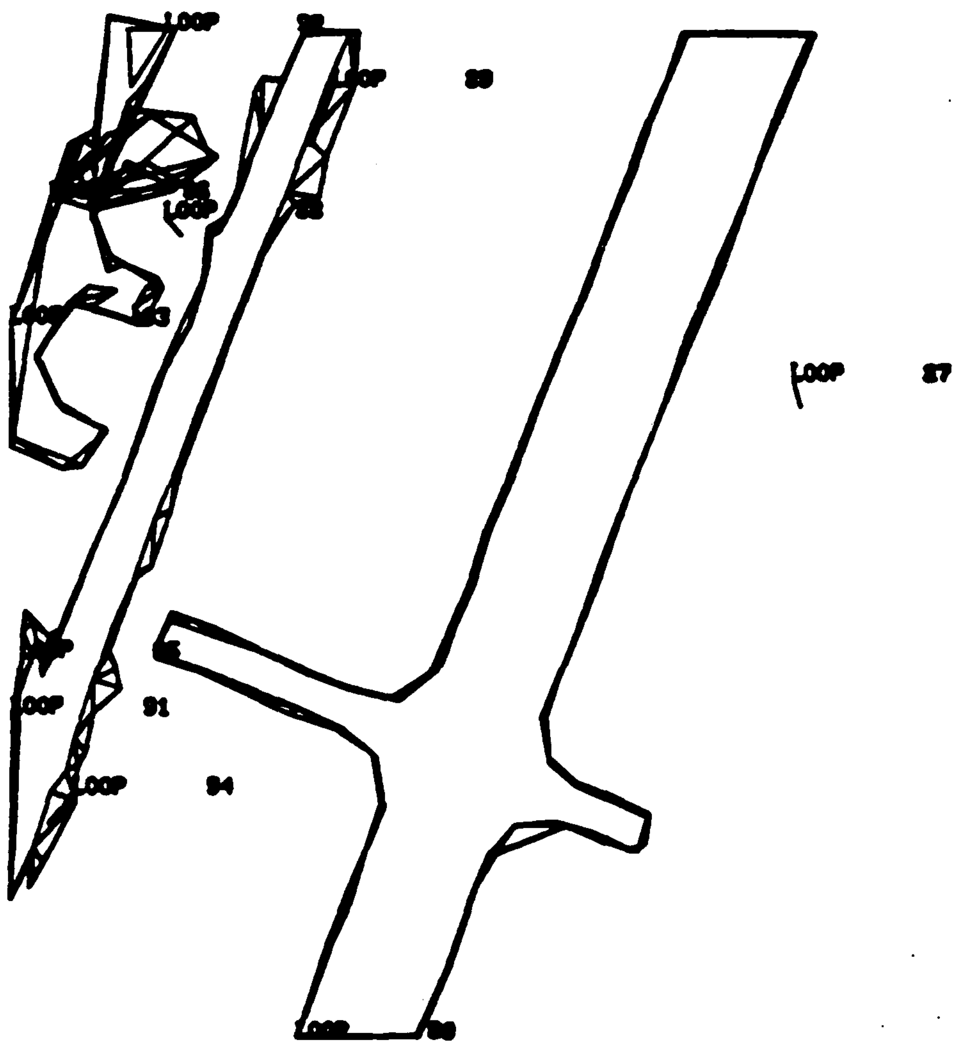


Figure 5.22 Triangulation - Loops 5-6

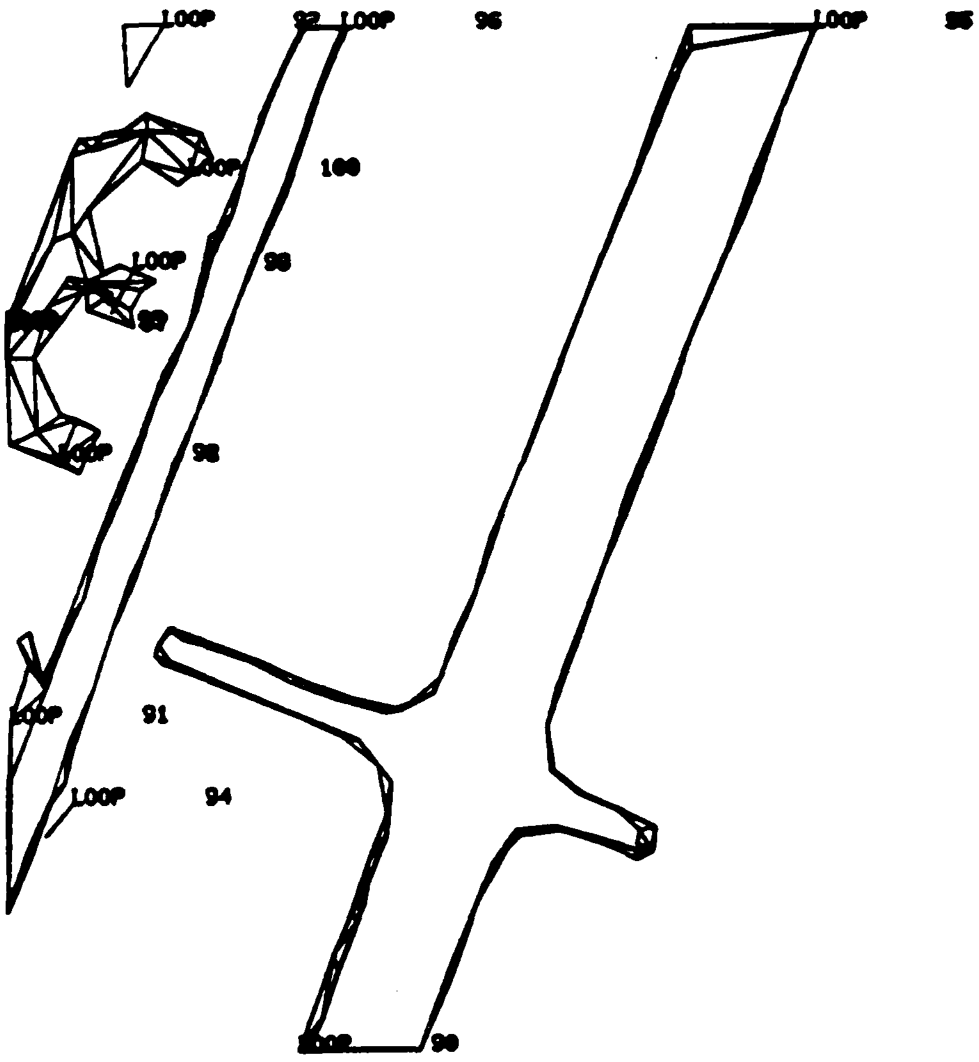


Figure 5.23 Triangulation - Loops 6-7

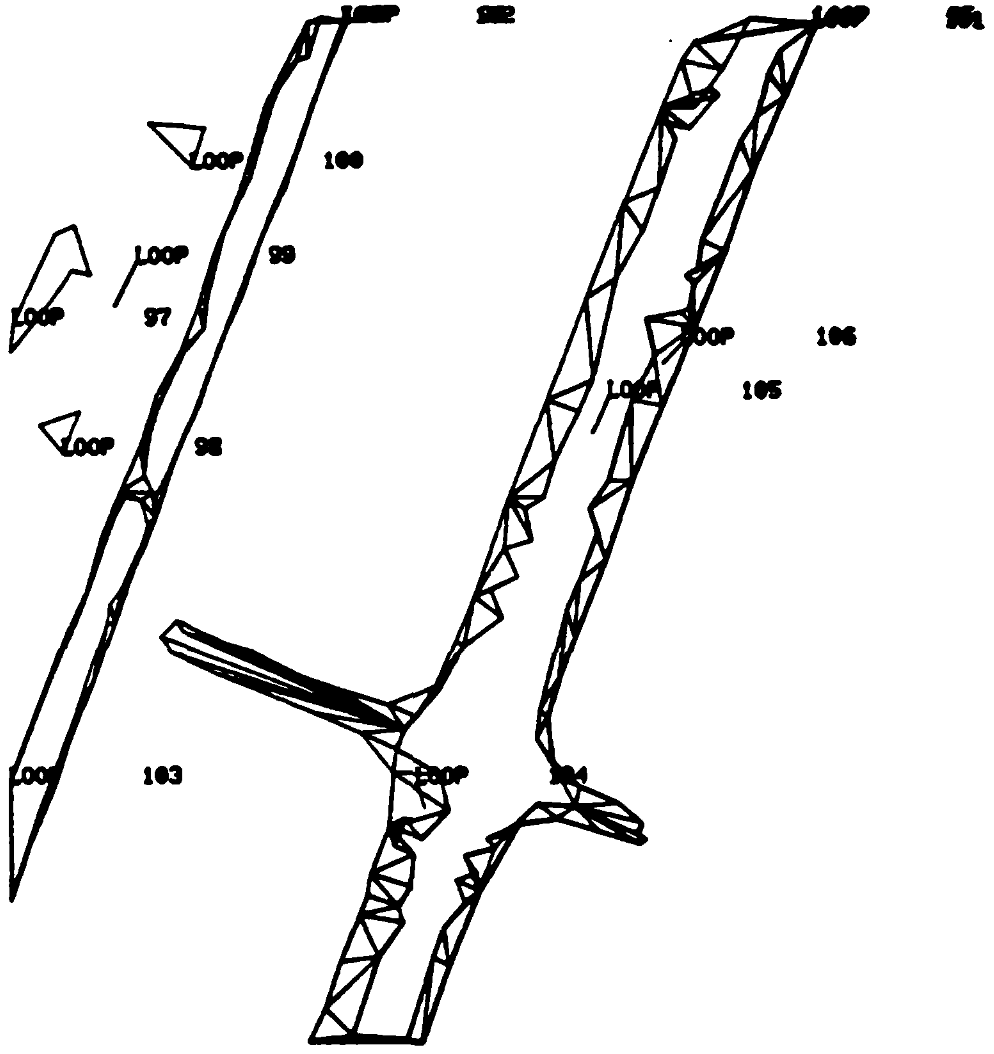


Figure 5.24 Triangulation - Loops 7-8

arbitrary point (X, Y) of a uniform (256x256) grid to determine which triangle it lies in. After the appropriate triangle is computed, a gray level is assigned to this point by doing a linear interpolation inside the triangle where the point lies.

A program, FRA.FOR, was written to accomplish this step. This program reconstructs the image as an uniform grid of pixels with the same size as the original (256x256). To accomplish this the data file created by MOSAIC is fed into FRA.FOR. The output of FRA.FOR is a uniform (256x256) grid representing the reconstructed image. Source code for FRA.FOR is contained in Appendix B.

The COMTAL system was used to display this reconstructed image. Before this could be done the uniform (256x256) grid created by FRA.FOR must be converted into a COMTAL format, a uniform (512x512) grid. DISP3M.FOR accomplished this by carrying out a two by two expansion. The source code for DISP3M.FOR is contained in Appendix B.

4. Analysis of the Reconstructed Image

Figure 5.25 is the original compressed image and Figure 5.26 is the reconstructed image. A qualitative analysis of the two images reveals several points. First, in the original image the edges of the contour levels do not appear sharp. The jagged edges appear to be a series of Z's. Comparing this to the reconstructed image, the edges are more well-defined with a low overshoot.

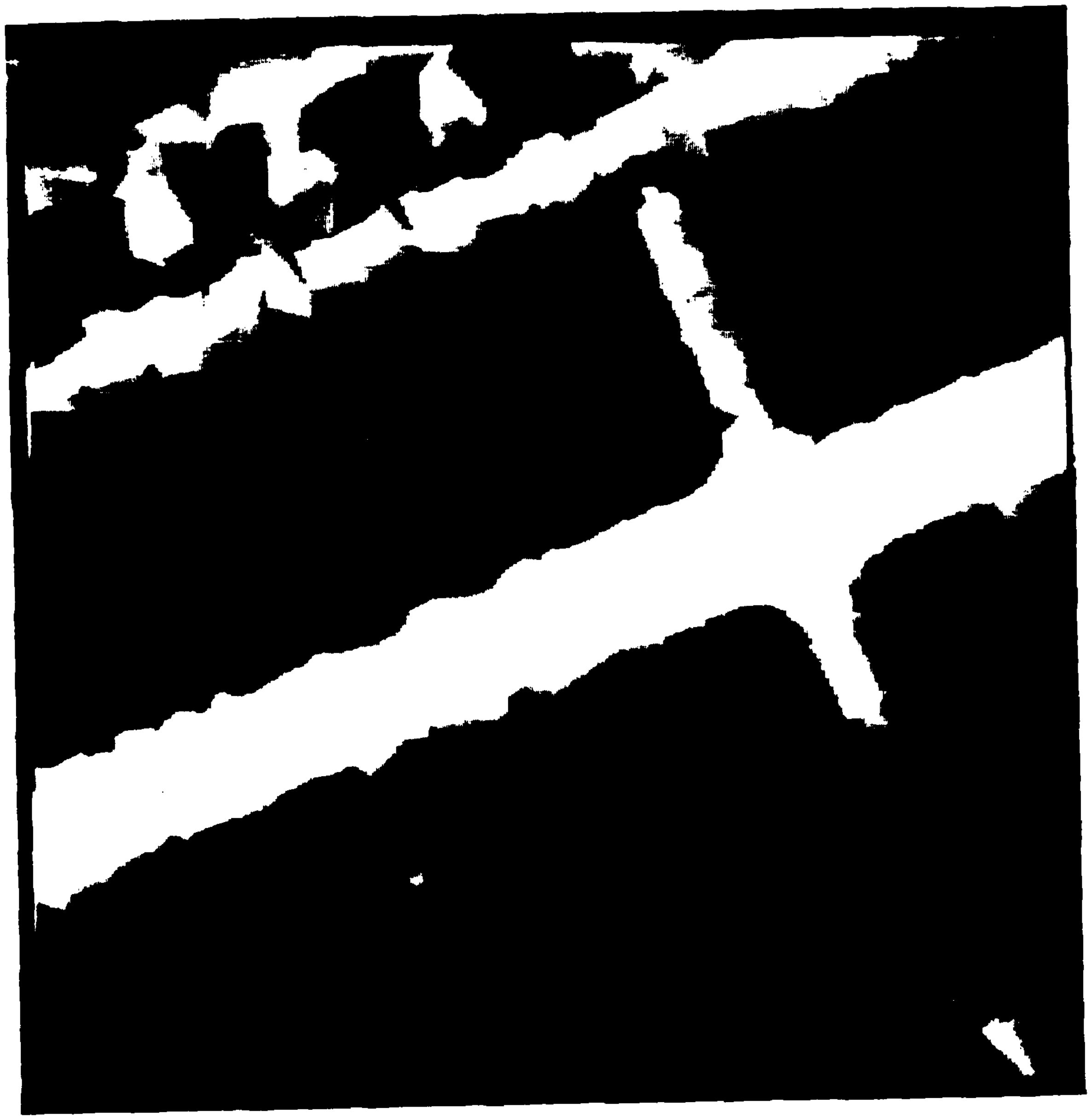


Figure 5.23 Original Compressed Image

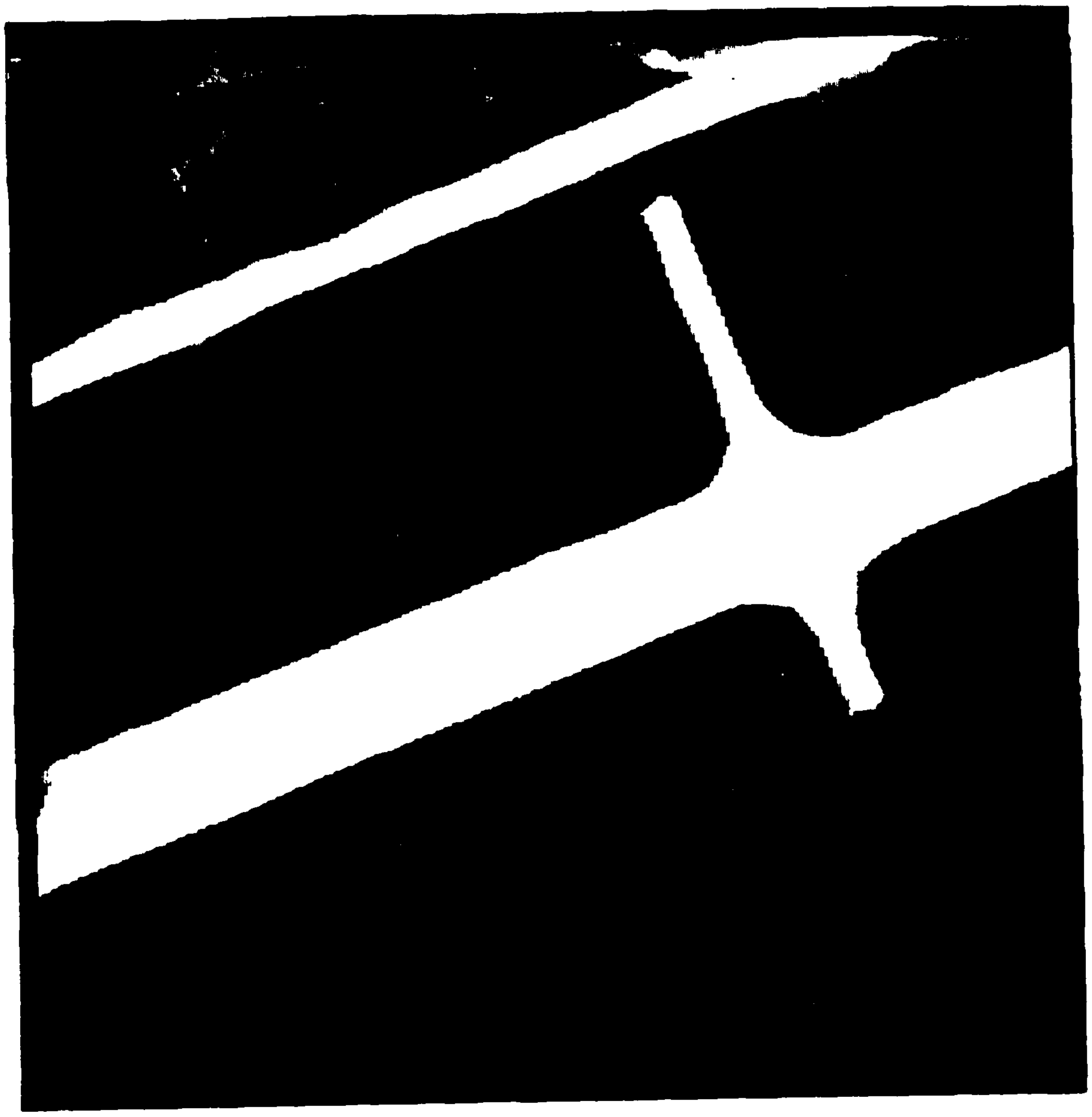


Figure 5.26 Reconstructed Compressed Image

On the minus side there is a loss of detail in the reconstructed image where the image consists of small, irregular shaped contours. This can be attributed to the difficuly in determining how to close these loops properly and to which loops they connect to. Also MOSAIC is unable to handle any loop that has less than three nodes. This meant that these loops were not read into MOSAIC and therefore had no effect on the reconstructed image.

The final decision of whether the reconstructed image can be considered an improvement over the original image can be arguably called favorable. The major features of the image are definitely improved. The degradation of the small detail is something that the normal viewer would not notice.

Next a quantitative analysis was done comparing both the compressed image and the reconstructed compressed image to the original uncompressed image.

The definition of error used was the root mean square error. It was chosen since it is probably the most commonly used method of error measurement used in image processing. Background on RMS error was discussed in Chapter 3. RMS error may be defined as the square root of the squared error averaged over the image array. The averaging sense of this definition resembles that of the intrinsic human visual system. It tends to filter out high frequencies and introduces smoothing effects in the image.

The program RMSER.FOR was used to compute the RMS error. Source code for this program is included in Appendix B.

The results of this analysis showed an RMS error of 17.3 percent for the original compressed image and 24.8 percent for the reconstructed image. This is not surprising since before even starting to use MOSAIC for triangulation all contours with less than three nodes were lost due to limitations of MOSAIC. The second factor that contributes to this error rate is the fact
that other small contours were effectively ignored because their connectivity to other contours could not be determined. Still the percentage of error is relatively low when compared to the error rate of the original compressed image.

Figure 5.27 contains a flow chart for the experimental work carried out. The next chapter will discuss lessons learned and recommendations for future work.

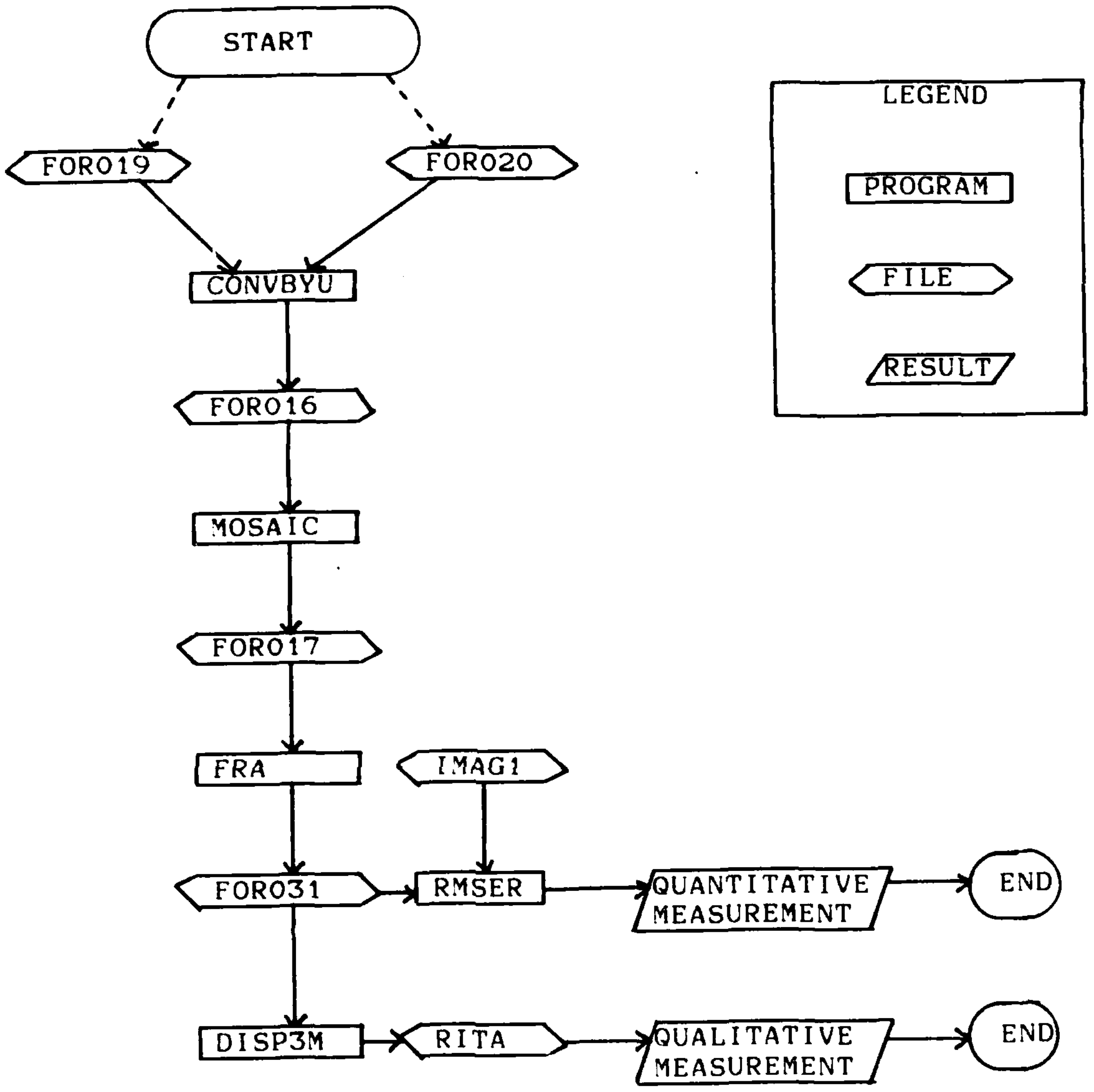


Figure 5.27 Process Flow Chart

VI. CONCLUSIONS

A. COMMENTS

Results obtained in this study can be summed up as both promising and disappointing. There is no clear outcome one way or the other. On the positive side the user is able to improve the presentation of contours that represent large man-made objects or well-defined terrestial objects. The edges of the contour levels represented in such images are sharp with less crossover into adjacent levels.

On the negative side, the interactive restoration scheme used here results in a significant loss of detail in the information contained in small, irregular shaped contours. Depending on the image being reconstructed or the use of this reconstructed image, this may or may not be of importance. If the image is intended for visual interpretation only, the loss of detail may not even be noticed. However, if the image is to be subject to computer analysis, it may make a big difference.

B. LESSONS LEARNED

1. MOVIE.BYU

Although MOVIE.BYU is a very powerful set of software tools, it may not be the ideal set of programs to use in image reconstruction. MOSAIC is better at manipulating contours that are well-defined and close naturally. When it is given irregular contours that are manually closed, undesirable results sometimes arise after the triangulation algorithm is applied.

Some algorithms overcome this by triangulating a surface based on the (X, Y) coordinates of the nodes with no attention paid to the gray contour levels. This, however, gives rise to jagged edges of gray levels when the reconstruction of an image is finished.

Another minor problem with MOSAIC is the amount of work needed to install MOSAIC. As originally configured, MOSAIC is capable of only handling a small number of contours with a small number of nodes. This is not sufficient to handle the complexity of most images. To modify MOSAIC to handle these images one must not only increase the values of the appropriate parameters but also search through all subroutines and increase the dimensions of arrays that are related to these parameters.

A more serious problem is the fact that MOSAIC is not a very forgiving program. Entry of a wrong number can cause repercussions through all contour levels and not just the contour being currently worked on. Correction of this error often becomes a difficult, if not impossible, task. Unless the error can be discovered and corrected quickly, it is usually more time efficient to start over from the beginning. The user soon learns after a couple of mistakes to be extremely careful when entering data or risk losing several hours of work.

In its defense, MOSAIC is a very complicated program that offers a large degree of user interaction. Due to the complexity of the above problems there is no easy solution. When dealing with irregular shaped contours it is a very difficult task to offer solutions to every possibility.

2. Hardware Requirements

As the complexity of a contour level increases it becomes more difficult to distinguish individual nodes and loop numbers. This has a detrimental effect when it comes to closing contours manually and connecting contours together.

The Tektronix 4010 proved to be no more than an adequate display device. This is easily confirmed by examining the contour levels presented in Chapter V. It is impossible to accurately read the loop number of every loop. Better reconstruction results may be expected if the display presentation of the contours is improved.

3. Skill Level of the Interactive User

The skill level of the interactive user plays a key role in the reconstruction process. This is because interactive reconstruction methods are of a subjective nature.

A user must be able to spot inconsistencies in the contour presentations. Unfortunately there is no all encompassing list of possible errors. The user must catch these inconsistencies through a combination of general image processing knowledge and common sense. As an example, image photographs are not able to show what is below a layer, they only show what is on top. The user should be able to notice when the triangulation process starts to violate this basic fact.

With a basic understanding of image processing, the user can be expected to progress in his ability to accurately reconstruct an image. He will be able to spot prospective trouble spots through the experience he has gained. Also his understanding of the effect of available program commands will allow him to manipulate contours easier. Although a new user is able to build on the experience gained by others, he must still progress through a learning curve of his own before he can be considered proficient.

C. CONCLUSIONS

There is a need for interactive restoration techniques. Due to the complexity of images there will always be situations where a conventional restoration algorithm falls short. This is especially true when the primary use of a restored image is for visual presentations.

More work must be performed in the area of reconstruction of small, irregular shaped contours. At the present time it takes a combination of a highly skilled user and prior knowledge of what to expect from the contours to obtain an acceptable reconstruction of such a region.

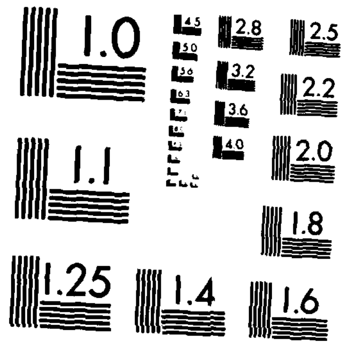
Interactive programs must be developed that give the user even more control in the manipulation of contours. A possible enhancement is the use of a trackball or mouse to control the addition, deletion, or movement of nodes. Another essential feature of a program is the ability to zoom in on a particular area of interest. This allows the user to pick loop and node numbers of interest when an area is congested with information. Both of these features were lacking in MOVIE.BYU. With these features it is felt that a better image reconstruction could be obtained.

Additional research in the area of image combining may prove of interest. The word "optimum" as applied to interactive reconstruction usually implies an optimum response of the human visual system. When "optimum" is used in conjunction with automated algorithms, it usually is in reference to a mathematical concept. By combining the best features of each image obtained with different approaches into one image, it should be expected that a superior reconstruction result will result.

Image processing is an area that has experienced vigorous growth recently. It can be expected that this

AD-A172 323 INTERACTIVE RECONSTRUCTION OF COMPRESSED IMAGES(U) 2/2
NAVAL POSTGRADUATE SCHOOL MONTEREY CA A LEDESMA JUN 86

UNCLASSIFIED F/G 9/2 NL


1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

growth will continue. Along with this growth will be an increased need for accurate reconstructed images. With improved software and display devices interactive methods will certainly play a role. The reason is that, although machines play a significant part in image processing, it is a human observer who must ultimately interpret or approve an interpretation of an image.

APPENDIX A

MOSAIC COMMANDS

A. GLOBAL COMMANDS

1. DEVIce allows the user to select the display device.

2. EXIT provides a controlled exit from the program.

3. HELP types the available commands at the current point on the user's terminal.

4. MAP functions as a toggle switch and allows the user to bypass the mapping algorithm or to restart its use.

5. PART allows the user to group elements into parts as they are generated.

6. TOTAls causes the number of parts, nodes, elements, and the total surface area of the elements to be typed on the output device.

7. WARP allows the user to specify the angle which controls the combining of triangular elements into quadrilateral elements.

8. WRITe enables the MOSAIC format contour data to be written to disk, the DISPLAY format panel data to be written to disk, and the DISPLAY format contour line segment data to be written to disk.

B. LEVEL ONE COMMANDS

1. CLIP acts as a toggle switch. When the switch is turned on the program requests the data necessary for clipping. The clipping plane algorithm can accommodate any number of clipping planes.

2. CLOCkwise allows the user to specify the ordering of the contour loops.

3. CONVert proceeds to level two. It has no value unless some data has been read.

4. MAKE allows the user to generate a contour data file.

5. RANGe allows the user to specify which portion of the contour data file is to be read into MOSAIC.

6. READ reads in the data file and prepares it for triangulation. If the CLIP, CLOCkwise, or THINning commands are to be used they must be invoked prior to the issuance of this command. This is because the data is scaled, thinned for economy, checked for rotational direction, and clipped as it is read in.

7. THINning allows the user to specify the three node elimination parameters. These are minimum segment angle, minimum segment length, and maximum segment length. All three are defaulted in such a way that the data is not modified.

C. LEVEL TWO COMMANDS

1. AUTOmatic places the program in an "automatic" mode of operation. This means that loop connectivity, loop linkup, and triangulation will all be attempted automatically.

2. CAP allows the user to generate caps on contour loops.

3. EDIT allows the user to edit loops by using the following commands.

   a. ADD node allows the user to add nodes between any two adjacent nodes of the existing contour data set.

   b. DELEte node allows the user to delete any node in the contour data set.

   c. LOOP allows the user to select a new loop to edit.

   d. MOVE allows the user to change the coordinates of an already existing node.

4. INSPect allows the user to inspect the contour data one level at a time.

5. MANUal allows user assistance in the interpretation of ambiguous contour data by using the following commands.

   a. AUTOmatic causes the program to proceed to automatically triangulate a group of loops.

   b. BRIDge enables specification of bridges between two or more loops on the same level which are branching out of a loop (or loops) on the other level.

c. CLEAr causes the program to delete all polygons that have been generated between the current pair of contour levels. Control is then returned to level three.

d. GUIDe is a last resort option for use on unreasonable data. The idea is to guide the program in triangulation by specifying limits between which to triangulate. The following commands are used while in GUIDe.

(1) AUTOmatic invites the program to triangulate the loops automatically.

(2) DENSity allows the user to specify the node number label density on the display.

(3) ERASe allows the user to start over on the current set of loops.

(4) RENUmber allows the user to redefine the location of the number one node on the level "A" loop. Use this command if the automatic selection of the node number one location is unsatisfactory.

(5) TRIAngulate allows the user to specify the limits on loop numbers, at the two levels, between which triangulation is to take place.

e. INCLude allows the user to override the alogorithm which selects which loop(s) on level "A" connect to which loop(s) on level "B".

f. NEXT indicates that the user is satisfied with the triangulation process for the current pair of contour levels and wishes to proceed to the next level.

6. POST is identical to AUTOmatic, with the exception that as each loop is triangulated it is displayed for approval.

APPENDIX B

SOURCE CODE LISTINGS

The source code for all programs used in conjunction with MOVIE.BYU is listed in this appendix. The source code for the MOVIE.BYU programs may be obtained from Brigham Young University, Provo, Utah.

Source code is listed in the following order:

1. READDAT.PAS
2. CONVBYU.FOR
3. FRA.FOR
4. DISP3M.FOR
5. RMSER.FOR

```
PROGRAM READDAT (IFILE,INPUT,OUTPUT);

(***********************************************************)
(*                                                         *)
(*  PROGRAM READDAT reads a packed file containing the     *)
(*  image data.  The image data is then represented by a   *)
(*  grid of 256 by 256 pixels in the range from 0 to 255.  *)
(*  The program also lets the user to selectively 'blank'  *)
(*  any portion of the image by selecting the range of the *)
(*  grid that is of interest.                              *)
(*                                                         *)
(*  INPUT  - TAD.DAT:   file containing image              *)
(*           keyboard                                      *)
(*  OUTPUT - IMAG1.DAT: file containing 256 by 256 pixels  *)
(*                                                         *)
(***********************************************************)

TYPE
   BYTE      = 0..255;
   IDATA     = PACKED ARRAY [1..512] OF BYTE;
   ODATA     = ARRAY[1..256,1..256] OF BYTE;
   OUTPUTDAT = ARRAY[1..256] OF BYTE;

VAR
   IFILE     : FILE OF IDATA;
   MYDATA    : ODATA;
   COLCOUNT  : 1..256;
   L,J,I,K   : 1..512;
   OFILE     : FILE OF OUTPUTDAT;
   RMIN,RMAX : INTEGER;
   CMIN,CMAX : INTEGER;
```

```
BEGIN

(** open files for reading and writing **)

   OPEN(IFILE,'TAD.DAT',HISTORY:=OLD,RECORD_LENGTH:=512);
   OPEN(OFILE,'IMAG1.DAT',HISTORY:=NEW,ACCESS_METHOD:=SEQUENTIAL,
      RECORD_TYPE:=FIXED);
   RESET(IFILE);
   REWRITE(OFILE);


   WRITELN('ENTER ROWMIN ==>  ');
   READLN(RMIN);
   WRITELN('ENTER ROWMAX ==>  ');
   READLN(RMAX);
   WRITELN('ENTER COLMIN ==>  ');
   READLN(CMIN);
   WRITELN('ENTER COLMAX ==>  ');
   READLN(CMAX);

(** check if file has data **)

   WRITELN('STARTING TO PROCESS ...');
   IF EOF(IFILE) THEN
     BEGIN
       WRITELN(OUTPUT);
       WRITELN('NO DATA FOR IFILE');
     END
   ELSE
       WRITELN('YES, THERE IS DATA');

(** read in image data **)

   FOR I:=1 TO 256 DO
```

```
      BEGIN
        GET(IFILE);
        FOR COLCOUNT:=1 TO 256 DO
          BEGIN
            K:=COLCOUNT*2 - 1;
            MYDATA[I,COLCOUNT]:=IFILE^[K];

(** assign gray level of 255 for all pixels outside area of
    interest **)

            IF (I<RMIN) OR (I>RMAX) OR (COLCOUNT< CMIN) OR
            (COLCOUNT>CMAX) THEN
               MYDATA[I,COLCOUNT] := (255);
               OFILE^[COLCOUNT]:=MYDATA[I,COLCOUNT];
          END;

(** write data to file **)

          PUT(OFILE);
          GET (IFILE);
      END
END.
```

```
C   ***********************************************************
C    PROGRAM CONVBYU converts contour files into a format that
C                    is usable by the MOVIE system.
C
C    INPUT    -  FOR019.DAT:  file containing number of contours
C                             for each gray level
C             -  FOR020.DAT:  file containing coordinate pairs of
C                             each contour
C    OUTPUT   -  FOR016.DAT:  file containing number of points of
C                             each contour, the gray level, and
C                             the coordinate pairs
C
C
C   ***********************************************************


C   ** declare variables  **

      INTEGER     NUMCONT, NUMPTS, STATUS, I, J
      REAL        GRAY
      DIMENSION   IX(1000),IY(1000),X(1000),Y(1000)

C   ** read number of contours for each gray level   **

      READ(19,100) NUMCONT, GRAY

      PRINT 900
900   FORMAT(' **  Starting conversion . . .')
      PRINT 905, GRAY
905   FORMAT('        GRAY LEVEL  ==>  ',F10.5)

      DO WHILE (NUMCONT .NE. -3)
        IF (NUMCONT .LT. 1) GOTO 10
        DO 15 I = 1,NUMCONT

C   ** read number of points of contour and X, Y pairs   **
```

```
            READ(20,110) NUMPTS, STATUS
            IF (NUMPTS .EQ. 0) GOTO 15
            DO 20 J = 1,NUMPTS
              READ(20,110) IX(J), IY(J)
              X(J) = FLOATJ(IX(J))
              Y(J) = FLOATJ(IY(J))
20          CONTINUE

C  ** write data to new file unless less than 3 points

            IF (NUMPTS .LT. 3) GOTO 15
            WRITE(16,120) NUMPTS, GRAY
            DO 25 J = 1,(NUMPTS/3)*3,3
              WRITE(16,130) X(J),Y(J),X(J+1),Y(J+1),X(J+2),Y(J+2)
25          CONTINUE
            IREM = JMOD(NUMPTS,3)
            IF (IREM .EQ. 1) WRITE(16,130) X(J),Y(J)
            IF (IREM .EQ. 2) WRITE(16,130) X(J),Y(J),X(J+1),Y(J+1)
15          CONTINUE
10        CONTINUE

          READ(19,100), NUMCONT, GRAY
          PRINT 905, GRAY

        END DO

C  ** set end of contours flag at end of file  **

        NUMPTS = 0
        GRAY   = 0.0

        WRITE(16,120) NUMPTS,GRAY

        PRINT 910
910     FORMAT (' **  End of conversion ! ! !')
```

```
100    FORMAT(I5,F10.5)
110    FORMAT(2I5)
120    FORMAT(I12,E12.5)
130    FORMAT(6E12.5)

       END
```

```
C   ************************************************************
C
C   PROGRAM:      FRA.FOR
C   AUTHOR:       LT A. LEDESMA
C                 (various ideastaken from routines written
C                  by Richard Franke, Naval Postgraduate School,
C                  Monterey, Ca)
C   DATE:         24 MAY 86
C
C   This program takes the triangulation data from MOSAIC.FOR and
C   converts it into a 256x256 uniform grid of function values.
C   Points lying within a triangle are interpolated to obtain a
C   gray value.
C
C   VARIABLES:
C
C    NPI    - number of data points
C    X      - array containing X coordinates of data points
C    Y      - array containing Y coordinates of data points
C    F      - array containing gray level of data points
C    FO     - 256x256 grid of function values
C    NXO    - number of values in XO
C    XO     - array of X values for function values
C    NYO    - number of values in YO
C    YO     - array of Y values for function values
C    NE     - number of segments
C    NT     - number of triangles
C    NL     - number of border line segments
C    IPT    - integer array where point numbers of the vertices of
C             the triangles are stored
C    IPL    - integer array where point numbers of border line
C             segments along with the segment's associated triangle
C             are stored
C    IPL1   - working IPL array
C    ITI    - triangle number
```

```
C   IMIN   - minimum gray level
C   IMAX   - maximum gray level
C   IRANGE - range of gray levels
C   JUNK1  - used to read in unneeded data from FOR017.DAT
C   JUNK2  - used to read in unneeded data from FOR017.DAT
C   WK     - work array
C   IWK    - work array
C
C  INPUT FILES:
C
C   FOR017.DAT   - file containingcoordinates and gray levels
C                  of data points and triangle vertices
C
C  *************************************************************

C  **  Variable declaration  **

      REAL X(15000),Y(15000),F(15000),FO(256,256),XO(256),YO(256)
      REAL WK(130000),B(3),F1(3),XT(3),YT(3)
      INTEGER IWK(480000), IPROV(256)
      DIMENSION IPT(70000),IPL(40000),IPL1(42000)
      COMMON/IDLC/NIT

C  **  Function statements  **

      SIDE(U1,V1,U2,V2,U3,V3) = (U1-U3)*(V2-V3)-(V1-V3)*(U2-U3)
      ACF(X1,Y1,X2,Y2,X3,Y3)  = (X1-X2)*(Y1-Y3)-(X1-X3)*(Y1-Y2)
      IND(I1)                 = MOD(I1-1,3)+1

      NIT = 0

C  **  Read in nodes' X,Y coordinates and gray level  **

      READ(17,500) JUNK1,NPI,NT,NE
      READ(17,500) JUNK1,JUNK2

      TYPE *,'***  Reading in coordinates and gray values . . .'
      TYPE *,' '

      DO 10 I = 1,NPI,2
         READ(17,510) X(I),Y(I),F(I),X(I+1),Y(I+1),F(I+1)
10    CONTINUE

      TYPE *,'***  Coordinates and gray values read in ! ! !'
      TYPE *,' '

C  **  Read in triangle vertices  **

      TYPE *,'***  Reading in triangles . . .'
      TYPE *,' '

      READ(17,500) (IPT(I), I=1,NE)
      DO 15 I = 1,NE
         IPT(I) = JIABS(IPT(I))
15    CONTINUE

      TYPE *,'       Number of triangles:  ',NT
      TYPE *,'***  Triangles read in ! ! !'
      TYPE *,' '

C  ** Form segment array, eliminate segments not on boundary **

      TYPE *,'***  Forming segments array . . .'

      ITI = 0
      I1  = 1
      DO 20 I = 1,NE,3
         ITI = ITI + 1
         IPL1(I1)   = IPT(I)
         IPL1(I1+1) = IPT(I+1)
         IPL1(I1+2) = ITI
         IPL1(I1+3) = IPT(I+1)
         IPL1(I1+4) = IPT(I+2)
         IPL1(I1+5) = ITI
         IPL1(I1+6) = IPT(I+2)
         IPL1(I1+7) = IPT(I)
         IPL1(I1+8) = ITI
         I1 = I1 + 9
20    CONTINUE

      TYPE *,'      ** Eliminating segments not on boundary . . .'

      NL  = 0
      DO 25 I = 1,I1-4,3
         IP1 = IPL1(I)
         IP2 = IPL1(I+1)
         DO 30 J = 1,I1-1,3
            IP3 = IPL1(J)
            IP4 = IPL1(J+1)
            IF (I .EQ. J) GOTO 30
            IF (IP1 .EQ. IP3 .AND. IP2 .EQ. IP4) GOTO 25
            IF (IP2 .EQ. IP3 .AND. IP1 .EQ. IP4) GOTO 25
30       CONTINUE

C  **  Segment is on boundary  **

         NL = NL + 1
         IPL(3*NL-2) = IPL1(I)
         IPL(3*NL-1) = IPL1(I+1)
         IPL(3*NL)   = IPL1(I+2)

25    CONTINUE

      TYPE *,'      **  Segment array complete ! ! !'
      TYPE *,'            Number of segments:  ',NL
      TYPE *,' '
```

```
C  **  Fill in grid arrays  **

        NXO = 256
        NYO = 256
        DO I = 1,256
           XO(I) = FLOAT(I)
           YO(I) = XO(I)
        END DO

C  **  Evaluate grid points to determine if point is in a
C  **  triangle  **

        TYPE *,'*** Checking if grid point is in a triangle . . .'
        TYPE *,' '

        ITI = 0

        DO 35 J = 1,NYO
           DO 40 I = 1,NXO
              FO(I,J) = 0.0
              DO 45 K = 1,NT

C  **  Get vertices of triangle  **

                 K1 = 3*K
                 I1 = IPT(K1-2)
                 I2 = IPT(K1-1)
                 I3 = IPT(K1)
C  **  Compute min and max values of X and Y coordinates of
C  **  triangle  **

                 XMN = AMIN1(X(I1),X(I2),X(I3))
                 XMX = AMAX1(X(I1),X(I2),X(I3))
                 YMN = AMIN1(Y(I1),Y(I2),Y(I3))
                 YMX = AMAX1(Y(I1),Y(I2),Y(I3))

C  **  Check if point in triangle  **
```

```
              IF (XO(I) .LT. XMN) GOTO 45
              IF (XO(I) .GT. XMX) GOTO 45
              IF (YO(J) .LT. YMN) GOTO 45
              IF (YO(J) .GT. YMX) GOTO 45
              IF (SIDE(X(I1),Y(I1),X(I2),Y(I2),XO(I),YO(J)).LT.0.0)
                 GOTO 45
              IF (SIDE(X(I2),Y(I2),X(I3),Y(I3),XO(I),YO(J)).LT.0.0)
                 GOTO 45
              IF (SIDE(X(I3),Y(I3),X(I1),Y(I1),XO(I),YO(J)).LT.0.0)
                 GOTO 45

C  **  Interpolate to find gray level of point  **

                DEL = ACF(X(I1),Y(I1),X(I2),Y(I2),X(I3),Y(I3))
                IF (DEL .EQ. 0.) GOTO 45
                FTEMP = 0.0
                F1(1) = F(I1)
                F1(2) = F(I2)
                XT(1) = X(I1)
                XT(2) = X(I2)
                XT(3) = X(I3)
                YT(1) = Y(I1)
                YT(2) = Y(I2)
                YT(3) = Y(I3)
                F1(3) = F(I3)
                DO 60 L = 1,3
                L1 = IND(L+1)
                L2 = IND(L+2)
           B(L) = ACF(XO(I),YO(J),XT(L1),YT(L1),XT(L2),YT(L2))/DEL
                FTEMP = FTEMP + B(L)*F1(L)
60              CONTINUE

                IF (FTEMP .GT. FO(I,J)) FO(I,J) = FTEMP

45            CONTINUE
40          CONTINUE
35        CONTINUE
```

```
C  **  Find max and min of gray levels  **

      IMIN=NINT(FO(1,1))
      IMAX=NINT(FO(1,1))
      DO I=1,256
         DO J=1,256
            ITEMP=NINT(FO(I,J))
            IF(ITEMP.GT.IMAX) THEN
               IMAX=ITEMP
            ELSE IF (ITEMP.LT.IMIN) THEN
               IMIN=ITEMP
            END IF
         END DO
      END DO
      TYPE *,'     MAX GRAY SCALE:  ',IMAX
      TYPE *,'     MIN GRAY SCALE:  ',IMIN
      IRANGE=IMAX-IMIN
      TYPE *,'     RANGE OF GRAY LEVEL:  ',IRANGE

C  **  Store matrix in file FOR031.DAT  **

      TYPE *,'*** Storing grid matrix . . .'
      TYPE *,' '

      DO I=1,256
         WRITE(31,130) I
         DO J=1,256
            IPROV(J)=NINT(FO(I,J))
         END DO
         DO K=0,7
            IST=32*K +1
            IEND=IST+31
            WRITE(31,128)(IPROV(J),J=IST,IEND)
         END DO
      END DO
      TYPE *,'***  End of program ! ! !'

128   FORMAT(32I4)
130   FORMAT(I4)
500   FORMAT(16I5)
510   FORMAT(6E12.5)

      END
```

```
C   *****************************************************
C
C   PROGRAM disp3..FOR
C   WRITTEN BY  CESAR TADEU DE MIRANDA
C   DATE 17/OCT/83
C
C   *******************************************************

C VARIABLE DECLARATION
      BYTE MB(512)
      INTEGER IPROV(256,256),i,j,k,ivalue,iline,ist,iend

C READ THE MATRIX FROM FILE IR

       TYPE *,'ENTER DESIRED INPUT UNIT'
       ACCEPT *,IR
       TYPE *,'START READING MATRIX'
       DO I=1,256
          read(IR,130) Iline
          DO K=0,7
            IST=32*K +1
            IEND=IST+31
            read(IR,128)(IPROV(i,J),J=IST,IEND)
          END DO
       END DO

C TRANSPOSITION

      TYPE *,'STRAT TRANSPOSITION'
      DO I=1,256
         DO J=1,I
            IF(I.NE.J) THEN
               ITEMP=IPROV(I,J)
               IPROV(I,J)=IPROV(J,I)
               IPROV(J,I)=ITEMP
            END IF
         END DO
      END DO
      TYPE *,'END OF TRANSPOSITION'

C OUTPUT THE IMAGE

      OPEN(UNIT=2,NAME='rita.DAT',TYPE='NEW',ACCESS='DIRECT',
     1     RECORDTYPE='FIXED',RECORDSIZE=128)
      DO I=1,511,2
         IEX=I/2+1
         DO J=1,511,2
            JEX=J/2+1
            IVALUE=iprov(iex,jex)
            IF(IVALUE.LE.127) THEN
               MB(J)=IVALUE
            ELSE
               MB(J)= -256+IVALUE
            END IF
            MB(J+1)=MB(J)
         END DO
         WRITE(2'I) MB
         WRITE(2'I+1) MB
      END DO
      CLOSE(UNIT=2)
      TYPE *,'END OF PROGRAM'
      STOP
128   FORMAT(32I4)
130   FORMAT(I4)
132   FORMAT(I5,F10.5)
      END
```

```
C   *********************************************************
C
C     THIS PROGRAM WILL COMPUTE THE RMS ERROR BETWEEN THE
C     SMOOTHED DATA AND THE RETRIEVED DATA
C
C     WRITTEN BY C. T. MIRANDA -  2/NOV/83
C
C**********************************************************

C  VARIABLE DECLARATION

      INTEGER F(256,256),IM(256,256),ITA(256)
      INTEGER SAVE,ACT,I,J,ILINE,IST,IEND
      REAL ERTOT,ER2,A,B,ER2BAR,ERMS,ASUM,EPSILON,AMSE
      DATA N/256/

C  READ THE ORIGINAL DATA

      TYPE *,'ORIGINAL DATA BEING READ'
      OPEN(UNIT=7,NAME='IMAG1.DAT',RECORDSIZE=256,
     1     ACCESS='DIRECT',TYPE='UNKNOWN')
      DO I=1,256

         TYPE *,'RECORD £ ',I
         READ(7'I) ITA
         DO J=1,256
            F(I,J)=ITA(J)
         END DO
      END DO
      TYPE *,'ALL RECORDS WERE READ'

      TYPE *,'FIRST ROW'
      WRITE(6,82)(F(1,J),J=1,256)
      TYPE *,'256 ROW'
      WRITE(6,82)(F(256,J),J=1,256)
```

```
C  SMOOTHING OF DATA

      TYPE *,'STARTING OF SMOOTHING'
      DO I=1,256
         SAVE=F(I,2)
         F(I,2)=(F(I,1)+F(I,2)+F(I,3))/3.
         DO J=3,255
            ACT=F(I,J)
            F(I,J)=(SAVE+ACT+F(I,J+1))/3.
            SAVE=ACT
         END DO
      END DO
      DO J=1,256
         SAVE=F(2,J)
         F(2,J)=(F(1,J)+F(2,J)+F(3,J))/3.
         DO I=3,255
            ACT=F(I,J)
            F(I,J)=(SAVE+ACT+F(I+1,J))/3.
            SAVE=ACT
         END DO
      END DO
      TYPE *,'END SMOOTHING'

C     INPUT THE APPOXIMATED DATA
      TYPE *,'START READING THE APPROXIMATED MATRIX'
      DO I=1,256
         READ(31,130) ILINE
         DO K=0,7
            IST=32*K +1
            IEND=IST+31
            READ(31,128)(IM(I,J),J=IST,IEND)
         END DO
      END DO
      TYPE *,'FINISH INPUT OF APPROXIMATED DATA'
```

```
C  TRANSPOSITION OF THE MATRIX

      TYPE *,'START TRANSPOSITION'
      DO I=1,256
         DO J=1,I
            IF(I.NE.J) THEN
            ITEMP=IM(I,J)
            IM(I,J)=IM(J,I)
            IM(J,I)=ITEMP
         END IF
      END DO
   END DO
   TYPE *,'END TRANSPOSTION'

C  FIND THE RMS ERROR

   TYPE *,'EVALUATE THE RMS ERROR'
   ERTOT=0.
   ASUM=0.
   DO I=1,256
      DO J=1,256
      A=FLOAT(F(I,J))
      B=FLOAT(IM(I,J))
      ER2=(A-B)**2
      ERTOT=ERTOT+ER2
      ASUM=ASUM+A**2
   END DO
   END DO
      ER2BAR=ERTOT/(FLOAT(N)**2)
      ERMS=SQRT(ER2BAR)
      TYPE *,'ERROR RMS...',ERMS
      EPSILON=ERTOT/ASUM
      AMSE=SQRT(EPSILON)*100
      TYPE *,'RELATIVE MSE...',AMSE,' %'
      TYPE *,'END OF PROGRAM'
      STOP
   82 FORMAT(1X,16I6/)
  128 FORMAT(32I4)
  130 FORMAT(I4)
      END
```

LIST OF REFERENCES

1. Huang, T. S., Picture Processing and Digital Filtering, Springer-Verlag, 1975.

2. Gonzales, R. C. and Wintz, P., Digital Image Processing, Addison-Wesley Publishing Company, 1976.

3. Lee, C. H., Image Compression Using Nonuniform Samples with Controlled Compression Ratio, paper presented at the Conference Preceedings of the 19th Annual Conference on Circuits, Systems, and Computers, Monterey, California, 9 November 1985.

4. Dierckx, P., "Algorithm for Smoothing Data with Periodic and Parametric Splines," Computer Graphics and Image Processing, v. 20, pp. 171-184, 1982.

5. Miranda, C. T., Image Data Compression Using Uneven Knots Selection, MSEE Thesis, Naval Postgraduate School, Monterey, California, June 1984.

6. De Boor, C., A Practical Guide to Splines, Springer-Verlag, 1978.

7. Andrews, H. C., "Image Approximation by Variable Knot Bicubic Splines,' IEEE Transactions in Pattern Analysis and Machine Intelligence, PAM 1-3, no. 3, pp. 299-310, May 1981.

8. Frendenhall, G. I. and Behrend, W. L., "Picture Quality - Procedures for Evaluation Subjective Effects of Interference," Proc. IRE, v. 48, pp. 1030-1034, 1960.

9. Lawson, C. L., "Software for C' Surface Interpolation," Mathematical Software III, pp. 161-194, 1977.

10. Christiansen, H. and Stephenson, M., MOVIE.BYU Training Text, 1986 Edition, Brigham Young University, 1986.

11. MOVIE.BYU User's Manual, Version 5.3, Brigham Young University, 1985.

INITIAL DISTRIBUTION LIST

		No. of Copies
1.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5000	2
2.	Defense Technical Information Center Cameron Station Alexandria, Virgina 22304-6145	1
3.	Department Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	1
4.	Professor Chin-Hwa Lee, Code 62Le Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	2
5.	Professor Mitchell L. Cotton, Code 62CC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	1
6.	Commander Naval Security Group Command Naval Security Group Headquarters Attn: Lt. A. Ledesma 3801 Nebraska Ave., N.W. Washington, D. C. 20390-0008	2

END

DTIC

10 — 86