NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

THREE - DIMENSIONAL IMAGE GENERATION FROM
AN AERIAL PHOTOGRAPH

by

Leland G. Coleman

September 1987

Thesis Advisor Chin H. Lee

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribtu-on 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 Postradaute 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)

Three - Dimensional Image Generation from an Aerial Photograph

12 PERSONAL AUTHOR(S)

Lt. Leland G. Coleman

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM ____ TO ____		106

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	3-D Image Generation, Photogrammetry

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

This thesis concerns developing a program that takes an aerial photograph, and a set of Digital Terrain Elevation Data (DTED) that is defined over the area of a photograph, and generates a synthesized view that represents what a camera would see from a different location. The elevation data points are grouped into triangular panels that are projected to the reference image by three dimensional transformation equations. Shading for the synthesized image is determined from the reference image. The pixels of the reference image that fall within a triangular panel are collected and averaged. When a new observer location is selected, the panels are projected to the new synthesized image plane. A z-buffer approach and a polygon fill algorithm were used to remove hidden surfaces of the synthesized view.

This program is tested on both artificial and real data. Other characteristics and performance measurements of the program are also analyzed here. The quality of the synthesized image from real data was affected by the low resolution of the terrain elevation data,

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

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

SECURITY CLASSIFICATION OF THIS PAGE

and yielded less desirable results than could be expected of a higher resolution terrain model.

Three - Dimensional Image Generation from an Aerial Photograph

by

Leland G. Coleman
Lieutenant, United States Navy
BS Electrical Engineering
University of Washington

Submitted in partial fulfillment of the requirments for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 1987

ABSTRACT

This thesis concerns developing a program that takes an aerial photograph, and a set of Digital Terrain Elevation Data (DTED) that is defined over the area of the photograph, and generates a synthesized view that represents what a camera would see from a different location. The elevation data points are grouped into triangular panels that are projected to the reference image by three dimensional transformation equations. Shading for the synthesized image is determined from the reference image. The pixels of the reference image that fall within a triangular panel are collected and averaged. When a new observer location is selected, the panels are projected to the new synthesized image plane. A z-buffer approach and a polygon fill algorithm were used to remove hidden surfaces of the synthesized view.

This program is tested on both artificial and real data. Other characteristics and performance measurements of the program are also analyzed here. The quality of the synthesized image from real data was affected by the low resolution of the terrain elevation data, and yielded less desirable results than could be expected of a higher resolution terrain model.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to express my deep appreciation to my Thesis Advisor, Dr. Chin-Hwa Lee, for his guidance and counsel in assisting in the completion of this Thesis.

I would also like to thank Dr. Mitchell L. Cotton and others who contributed their assistance in the accomplishment of this Thesis.

Finally, I wish to express my gratitude to my wife, Cynthia L. Coleman, who helped and supported me through these laborious times to achieve this educational goal.

I. INTRODUCTION

A. COMPUTER IMAGE GENERATION FROM AERIAL PHOTOGRAPHY

The main objective of this study was to develop a program that takes a digital photographic image and a file of terrain elevation points defined over that image as input, then produces as an output a synthesized perspective view. The synthesized view is a rotated 3-dimensional (3D) perspective representation of the original photographic image. The main application of this study is to generate a different perspective of a terrain model. This may be used to generate different views that a pilot of an aircraft could expect following different flight paths through the same area. Further study may make it feasible to generate synthesized images fast enough to simulate a real time image display of a flight for a mission briefing or to be used as a training aid. Another application could be for training men on optically guided missiles. With high resolution images, a flight path through a battlefield could be simulated that would have all the visual characteristics of an actual flight without the expenditure of a live missile. The generation of a shaded image as a 3D picture provides unique problems for 3D graphic displays. The data which comprises a photographic image consists of an array of pixels, each of which has a defined grey level or shade.

There are 256 different levels of grey that may be assigned to a pixel. This study concerns taking a photographic perspective image and a 2-dimensional (2D) array of elevations defined in a grid covering the area of the image as inputs. The grid of elevations, called the terrain model, is geometrically related to the photographic image through a perspective projection transformation that equates the world coordinates of the elevation points to the object coordinates of the image. A synthesized image from a different observer location is then generated. The new synthesized view should approximate what would be seen by a camera from the new observer position.

The differences between the original and synthesized images will be affected by the resolution of both the photographic image and the terrain models. Higher resolution of the original models will result in a closer approximation in the synthesized image. Another complication or ambiguity arises when details which should show up in the synthesized view were not present in the original image. A method must be devised so that it can fill in areas which become visible in the synthesized image that were hidden in the original reference image. The solution to this hidden surface problem is further addressed in the discussion of the grey scale referencing algorithm.[Ref. 1]

B. TERRAIN ELEVATION AND PHOTOGRAPH AS INPUTS

In this study a high altitude aerial image of Moffett Field, California was used as the original reference photograph. The photographic image was supplied by the Defense Mapping Agency (DMA) and had a resolution of approximately 1 meter per pixel. The terrain model corresponding to the reference image was provided as Digital Terrain Elevation Data (DTED) by the DMA and consisted of elevation points taken every second of a degree change in latitude and longitude. This gives an approximate resolution of 30 meters per elevation point in a north-south direction and 23 meters per elevation point in an east-west direction.

The synthesized view was restricted to a northerly direction which simulates an aircraft flying from south to north with the image plane perpendicular to the direction of flight. To allow for different flight patterns would require development of an algorithm that would provide for rotation of the image coordinates which is beyond the scope of this study. The main idea is to generate a synthesized view that is rotated from the original photograph by approximately 90° and explore the concepts of the algorithms required to do this. Although speed was not a major issue, the size of the terrain model was limited to a 50 x 50 grid array, or 2500 data points, to minimize the time for synthesized image generation.

C. ALGORITHM ISSUES

1. Grey Scale Referencing

To determine the grey scale values of the pixels that make up our synthesized view, the terrain model is first divided into triangular panels. The vertices of the triangular panels are then mapped into the original DMA image using the perspective projection transformation that projects georectangular coordinates into reference image coordinates. The pixel grey scale values that fall within the projected triangular panel are then averaged. The average grey scale value is permanently assigned to that particular panel. When the synthesized image is constructed the triangular panel is mapped to the new image view and filled with the assigned average grey scale value. In this way the sample of pixels that fall within the triangular panel are mapped from the original reference image to the synthesized image. This method of mapping the triangular panels to the synthesized image also solves the problem of assigning a grey scale value to hidden surfaces of the reference image because they are automatically assigned the value of the surrounding pixels. Since the average grey scale value for a triangular panel is dependant upon the resolution of the terrain model, the grey scale value assigned to the hidden surface will also be affected [Ref. 1].

The smaller the triangular panels are, the smaller the area that must be collected and averaged in the original image. This means a much better synthesized view can be constructed that contains more of the attributes of the original image. For this reason the resolution of the terrain and reference image model is very critical to obtaining an accurate synthesized view. Using the resolution of the terrain and image models used in this study, the approximate number of pixels that must be averaged in the reference image for each triangular plane would be 1/2 (30m x 23m)(1 pixel/m) = 345 pixels. This is very coarse and does not allow for optimal generation of the synthesized view.

2. Hidden Surface Elimination

There are surfaces that may be discernible in the reference image but become hidden in the synthesized view. The z-buffer algorithm was used to accomplish the hidden surface elimination. The z-buffer is an array that contains the depth or distance to the observer location for each pixel that is to be visible in the synthesized image. As each triangular plane is mapped to the synthesized view the location and depth of each pixel within the plane is determined. The depth of the pixel to be written at a certain location is compared to the depth of any pixel that may have been previously written to the same location. If the depth or distance of the new pixel to the observation point is shorter than the previous pixel, its depth is written to the z-buffer and the grey scale value of the pixel is placed into the synthesized image. If the depth is larger, no updating occurs and a new pixel is obtained in the process. The z-buffer works very closely with the next algorithm to be considered. [Ref. 2, pp. 265-267]

3. Polygon Fill Algorithm

Screen coordinates are generated for the three vertices of each triangular panel as it is mapped to the synthesized image. Screen coordinates are designated as IA and JA values with the IA values representing the columns and the JA values the rows. The location, IA,JA(0,0), designates the upper left hand corner of the screen and the maximum screen coordinate, IA,JA(512,512), the lower right hand corner. An active edge list (AEL) is generated by computing a line between each of the translated vertices. For each line, the IA coordinate corresponding to the maximum JA value of the line, the amount IA changes for each one unit step of JA, and the total span of JA are stored into the AEL. The three lines generated for each translated triangular plane will form another closed triangle. By using the parameters stored in the AEL the location of pixels enclosed by the translated triangular plane can be determined, and the corresponding array points within a frame buffer are changed from a 0 to a 1.
[Ref. 2, pp. 76-79]

After all of the enclosed pixels have been marked within the frame buffer, the buffer is scanned row by row. If a value of 1 is found, then the depth is calculated for that point and compared with the depth value stored in the z-buffer. If the depth value is smaller, that pixel is located closer to the observer location and the grey scale value for that pixel is written to the synthesized image file. As can be seen the fill and hidden surface algorithms work together to generate the new image. The implementation of these algorithms are explained in further detail later.

In Chapter II there will be a discussion of basic photographic geometry to develop an understanding of the transformation equations necessary to map object coordinates into image coordinates and for image plane rotation. Chapter III will detail program considerations based on image and elevation data as well as the algorithms used to generate the synthesized view. Chapter IV will discuss possible ways to improve the transformation program and discussion of topics for possible further study. An outline of the program is contained in Appendix A that gives a short discussion of each subroutine as to its purpose, input and output, modules that called, and modules that reference the subroutine. Appendix B contains the entire 3D transformation program.

II. PHOTOGRAPHIC GEOMETRY

A. BACKGROUND

To understand many of the concepts used in this study, a basic background in photographic geometry is presented. The relationship between the image space and object space is the basis for many of the equations that help to generate the reference and synthesized images for visual display. The objective of this chapter is to present the concepts that allow the transformation of 3D objects to a 2D image and the parameters evolved.

1. Perspective and Parallel Projection

A parallel projection is a projection in which the projection lines from the object to the image plane never converge. When an object is viewed by parallel projection, its size would never change as the camera is moved closer or further away. In contrast, a perspective projection has all the projection lines from an object converge to a perspective center. A perspective projection imitates how we see things. An example would be a picture of railroad tracks. The tracks would appear to become closer together when further away from the observation point. In a parallel projection the tracks would be the same distance apart along the entire length. [Ref. 3, pp. 133-134]

Since a camera is generally designed to photograph a rather large area, it involves perspective projection. The camera view represents what an observer would see standing at the same location, and the images generated are perspective images. This means that the equations used to transform the object space into the image space must be perspective transformations.

2. Image Coordinate Space

The image plane is the plane of the photograph to which the object points are mapped. It has a 2D coordinate system to which each point of a 3D object is translated to (Fig. 2.1). The indicated principal point (IPP) is the center of the image plane and has the coordinates of (x,y,0). The x, y, and z axis represent a right handed plane and the perspective center (L) lies along a line parallel to the z-axis; then a perpendicular line is drawn from L to the image plane. The point at which this perpendicular line intersects the image plane is called the principal point (o). This offset of the principal point from the IPP is compensated for in the transformation equations by xo and yo. The focal length of the plane is defined as the distance from the principal point to the perspective center. For the image plane coordinate system, each object point (A) is graphed to a corresponding image plane point (a) located at (xa,ya,0), and the perspective center or focal point is located at (xo,yo,f). [Ref. 3, pp. 135-136]

In generating a synthesized view, the perspective center may be placed at any location desired with reference to the image plane. It is therefore desirable to select a point along the z-axis such that xo and yo become 0. This will decrease the number of calculations required in generating the synthesized view.

3. Object Coordinate Space

The observer location and each object point position is described in world coordinates, called georectangular coordinates, of X, Y, and Z. The center of the earth is given as (0,0,0), the Z-axis points directly to true north, the X-axis points to the intersection of 0° Latitude and 0° Longitude, and the Y-axis the intersection of 0° Latitude and 90° E. Longitude. The principal or focal point that would describe the observer location in georectangular coordinates is XL, YL, and ZL. Each object point (A) located in the object space is identified by XA, YA, and ZA as shown in Figure 2.2. [Ref. 3, p. 136]

B. IMAGE PLANE ROTATION

To align the x, y, z coordinates of the image plane to the desired viewing direction requires rotation about the X, Y, and Z axis of the georectangular coordinate system. In general to transform one 3D coordinate system requires a matrix multiplication of the form **A** = [M]**B**. The **A** represents a vector in the image space with x, y, and z coordinates, and **B** is a vector in the georectangular coordinate

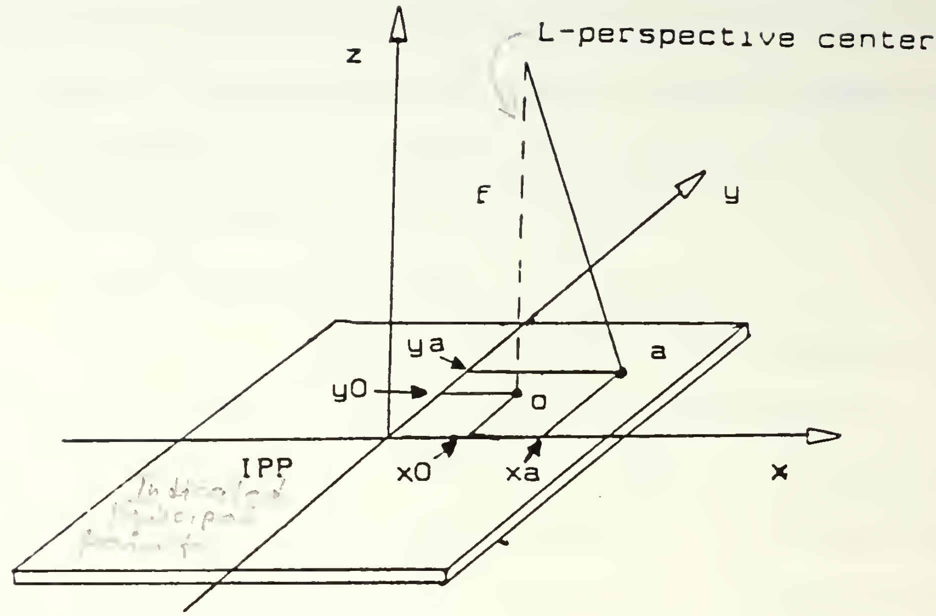


Fig. 2.1 Image Plane Coordinates (Ref. 3, p. 135)

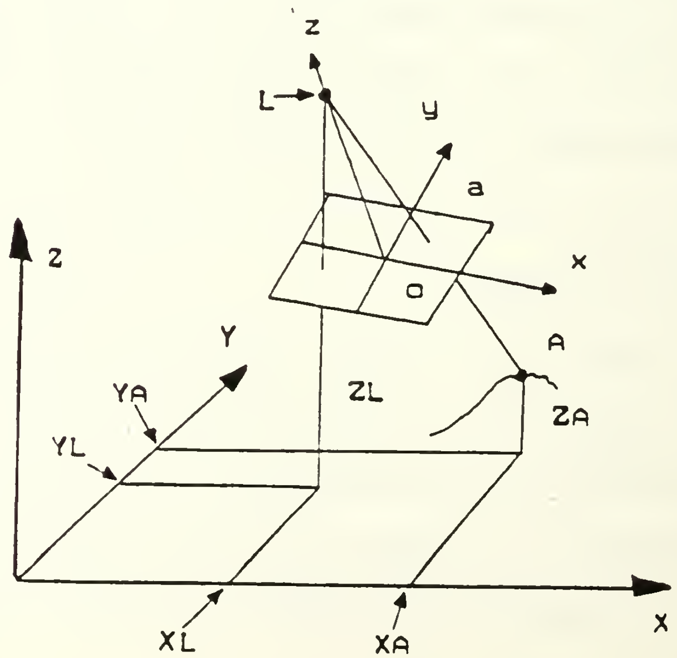


Fig. 2.2 Object Plane Coordinates (Ref. 3, p. 136)

system with X, Y, and Z components. This may be written as

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad (2.1)$$

where

$$M = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = \begin{bmatrix} \text{Cos}xX & \text{Cos}xY & \text{Cos}xZ \\ \text{Cos}yX & \text{Cos}yY & \text{Cos}yZ \\ \text{Cos}zX & \text{Cos}zY & \text{Cos}zZ \end{bmatrix} \qquad (2.2)$$

This maps the vector X, Y, Z to the image space x, y, z. The M matrix is derived from the definition of the direction of a vector **A** given by

$$\frac{\mathbf{A}}{|A|} = \text{Cos}\alpha\ \hat{i} + \text{Cos}\beta\ \hat{j} + \text{Cos}\tau\ \hat{k}$$

$$= \frac{Ax}{|A|}\hat{i} + \frac{Ay}{|A|} + \hat{j}\frac{Az}{|A|}\hat{k} \qquad (2.3)$$

where $\hat{i}$, $\hat{j}$, and $\hat{k}$ are unit vectors of the particular coordinate system in which vector **A** is contained. The quantities α, β, τ are the angles that vector **A** makes with the x, y, and z-axis respectively. Since there are three vector components in the X, Y, Z system, each one must make its own transformation into x, y, and z and thereby forming the 3 x 3 matrix of M. To translate x, y, z into X, Y, Z the inverse of the M matrix is taken giving $\mathbf{B} = [M]^{-1}\mathbf{A}$. [Ref. 3, p. 139]

If we can define three orthogonal vectors in georectangular coordinates as **R**, **S**, and **T** that would describe the desired viewing position of our image plane, the M matrix is easily derived by

$$M = \begin{bmatrix} \frac{Sx}{|S|} & \frac{Sy}{|S|} & \frac{Sz}{|S|} \\ \frac{Rx}{|R|} & \frac{Ry}{|R|} & \frac{Rz}{|R|} \\ \frac{Tx}{|T|} & \frac{Ty}{|T|} & \frac{Tz}{|T|} \end{bmatrix} \qquad (2.4)$$

Generally the image plane rotation is expressed in omega (ω), phi (Φ), and kappa (k). The ω is the rotation about the X-axis, the Φ is rotation about the Y-axis, and k is about the Z-axis. If we rotated first about the X-axis by an angle of ω radians, the terms of the M matrix would equate as follows

$CosxX = Cos(0°) = 1$

$CosyY = Cos(\omega)$

$CosyZ = Cos(90°-\omega) = Sin(\omega)$

$CoszY = Cos(\omega + 90°) = -Sin(\omega)$

$CoszZ = Cos(\omega)$

All other terms equate to cos 90° = 0, therefore the M matrix becomes

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{Cos}(\omega) & \mathrm{Sin}(\omega) \\ 0 & -\mathrm{Sin}(\omega) & \mathrm{Cos}(\omega) \end{bmatrix} \qquad (2.5)$$

Similarly a rotation ϕ about the Y-axis produces

$$M = \begin{bmatrix} \mathrm{Cos}(\phi) & 0 & -\mathrm{Sin}(\phi) \\ 0 & 1 & 0 \\ \mathrm{Sin}(\phi) & 0 & \mathrm{Cos}(\phi) \end{bmatrix} \qquad (2.6)$$

and for a rotation k about the Z-axis

$$M = \begin{bmatrix} \mathrm{Cos}(k) & \mathrm{Sin}(k) & 0 \\ -\mathrm{Sin}(k) & \mathrm{Cos}(k) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (2.7)$$

By multiplying all three matrices together we derive the overall M transform in ω, ϕ, and k,

$$M = \begin{bmatrix} \mathrm{Cos}(\phi)\mathrm{Cos}(k) & \mathrm{Cos}(\omega)\mathrm{Sin}(k) + \mathrm{Sin}(\omega)\mathrm{Sin}(\phi)\mathrm{Cos}(k) & \mathrm{Sin}(\omega)\mathrm{Sin}(k) - \mathrm{Cos}(\omega)\mathrm{Sin}(\phi)\mathrm{Cos}(k) \\ -\mathrm{Cos}(\phi)\mathrm{Sin}(k) & \mathrm{Cos}(\omega)\mathrm{Cos}(k) - \mathrm{Sin}(\omega)\mathrm{Sin}(\phi)\mathrm{Sin}(k) & \mathrm{Sin}(\omega)\mathrm{Cos}(k) + \mathrm{Cos}(\omega)\mathrm{Sin}(\phi)\mathrm{Sin}(k) \\ \mathrm{Sin}(\phi) & -\mathrm{Sin}(\omega)\mathrm{Cos}(\phi) & \mathrm{Cos}(\omega)\mathrm{Cos}(\phi) \end{bmatrix} \qquad (2.8)$$

This is the general form of the matrix that maps the georectangular coordinates to the image plane. [Ref. 3, pp. 597-600]

The general form of the transformation matrix is used to initially map the elevation terrain model into the original reference image. The ω, Φ, and k were supplied with the original DMA photograph and represents the rotation of the reference image plane with respect to the georectangular coordinates at the time the picture was taken. When we generate the synthesized view the image plane must be rotated to the desired viewing angle of the observer (northerly direction in particular). This means that a new ω, Φ, and k must be calculated, or by defining new image plane coordinate axes in terms of the georectangular coordinates, we can calculate the terms of the M matrix directly using the preceding equations. This is discussed further in the next chapter.

III. ALGORITHM CONSIDERATIONS

In this chapter an in-depth analysis of the original image and terrain data is presented. This includes how the data was referenced, the size of the data files, how the data was used, and how the elevation and image data compared with one another. The process of translating from the object space coordinates to image space coordinates and then to screen coordinates is considered, and the equations used are given.

The referencing, fill, and z-buffer algorithms also are discussed in detail. How the data generated from these algorithms is used and put together to produce the synthesized view will be presented. Any problems that were encountered and the eventual solutions will be discussed in the appropriate section to which they pertain.

A. REFERENCE IMAGE DATA

The picture of Moffett Field supplied by the Defense Mapping Agency (DMA), was 4999 by 4997 pixels in size and came with both a left and right image. Only the left image was used to generate the perspective views in this study. Each individual pixel within the original image is designated by coordinates I and J, where I is the pixel column and J is the scan row. The geographic northeast corner has the I, J coordinates of (0,0), and the southwest corner (4997,4999).

The image display devices used were capable of displaying images that were only 512 by 512 pixels in size, therefore, the original image was divided into appropriate blocks suitable for viewing called frames. Each frame contains an image that is 512 by 512 pixels. The coordinates of each frame has a four digit I_Frame and J_Frame value, and is further identified as a left or right (L or R) image. The I_Frame and J_Frame coordinates are designated in multiples of 512 which define the column and row location of each frame. There are 10 frame columns and 10 frame rows with assigned coordinates of 0000 through 4608. To identify a particular frame of the DMA image one first designates whether it is a Left or Right image and then give the I_Frame and J_Frame coordinates. As an example L05121024 would designate a Left image from the second column and third row. The first frame L00000000 starts in the southeast corner and the last frame L46084608 is in the northwest corner. The disparity between the starting location of the frame coordinates and the I and J coordinates of the original image must be compensated for in the equations that are used to determine the location of individual pixels within a frame image as shown later.

1. Object to Reference Image Transformation

Every object point is converted from its 3D X, Y, and Z georectangular coordinates into the 2D x and y image coordinates using the **A** = [M]**B** equation discussed earlier. The vector from the perspective center to each object point is defined by (XA-XL), (YA-YL), and (ZA-ZL). This vector is mapped into the reference image plane coordinates of x, y, and z. Since every vector in the image plane is directed from the perspective center or focal point to the image point (a), the z coordinate value is constant and equal to the negative of the focal point (-f) of the camera. Using these parameters the equation becomes

$$\begin{bmatrix} x-xo \\ y-yo \\ -f \end{bmatrix} = K \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} XA-XL \\ YA-YL \\ ZA-ZL \end{bmatrix} \quad (3.1)$$

where K is a scale factor. From this transformation the following equations are obtained

$$x-xo = K\,[a\,(XA-XL) + b\,(YA-YL) + c\,(ZA-ZL)] \quad (3.2)$$

$$y-yo = K\,[d\,(XA-XL) + e\,(YA-YL) + f\,(ZA-ZL)] \quad (3.3)$$

$$-f = K\,[g\,(XA-XL) + h\,(YA-YL) + i\,(ZA-ZL)] \quad (3.4)$$

Dividing the last equation into the first two and rearranging yields

$$x = xo - f \left[\frac{a\,(XA-XL) + b\,(YA-YL) + c\,(ZA-ZL)}{g\,(XA-XL) + h\,(YA-YL) + i\,(ZA-ZL)} \right] \quad (3.5)$$

$$y = yo - f \left[\frac{d\,(XA-XL) + e\,(YA-YL) + c\,(ZA-ZL)}{g\,(XA-XL) + h\,(YA-YL) + i\,(ZA-ZL)} \right] \quad (3.6)$$

where x and y are the 2D image plane coordinates.[Ref. 3, pp. 141-142]

The original parameters of the M matrix were calculated from the ω, Φ, and k that were given by the DMA with the original photograph. These represent the physical position of the image plane in relation to the georectangular coordinate system at the time the picture was taken. The focal point was also supplied and is particular to the camera that was used to take the original photograph. When the synthesized image is generated, the image plane is oriented to a position for the desired viewing angle, which means that the parameters of the M matrix will change and must be recalculated. The steps used to determine the desired orientation of the image plane coordinates will be discussed later in this chapter.

2. <u>Reference Image to Screen Coordinate Transformation</u>

Once the x and y image coordinates have been calculated they are translated to I and J values of the original image. This is accomplished using the affine transform which represents a 2D into 2D coordinate transformation. The equation that accomplishes this is derived from

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} j & k \\ l & m \end{bmatrix} \begin{bmatrix} I \\ J \end{bmatrix} + \begin{bmatrix} C1 \\ C2 \end{bmatrix} \qquad (3.7)$$

where C1 and C2 are the values that translate the image plane origin to the I and J coordinate system origin. They are calculated by setting I and J to 0 and solving for x and y. To get the desired transformation of the image coordinates to I and J coordinates the inverse transform is taken. [Ref.3, p.593]

$$\begin{bmatrix} I \\ J \end{bmatrix} = \frac{1}{jm - kl} \begin{bmatrix} m & -k \\ -l & j \end{bmatrix} \begin{bmatrix} x - C1 \\ y - C2 \end{bmatrix} \qquad (3.8)$$

Again the original j, k, l, m, C1, and C2 were supplied by the DMA with the original image. Equation 3.8 represents any general 2D into 2D transformation, and it is used both to translate the original image plane coordinates into the I and J coordinates of the reference image and to transform the image plane coordinates of the synthesized view into screen coordinates of IA, and JA.

The screen coordinates were assigned the parameters IA, which represents the columns, and JA, which represents the rows. The point IA,JA(1,1) is mapped to the upper left corner of the screen and IA,JA(512,512) is the lower right corner.

The screen coordinates represent the location of a pixel within a frame. To convert I and J original coordinates to IA and JA screen coordinates one needs to know the particular frame one is working in and compensate for the difference in the starting location of the frame coordinates and the original image coordinates. This is accomplished by using the following equations,

$$IA = (I - I\_Frame) \qquad (3.9)$$

$$JA = (4999 - J\_Frame - J) \qquad (3.10)$$

The I_Frame and J_Frame values must therefore be given to determine the screen coordinates within a desired frame. To allow flexibility in determining which frame image would be used to extract the reference image, an interactive input of the I_Frame and J_Frame coordinates was appropriate.

3. <u>Synthesized Image Plane Rotation</u>

For the synthesized views that were generated, the affine transform parameters C1, C2, j, k, l, and m that would map the newly rotated image plane into the screen coordinate system were selected. Since the image planes in the synthesized views were oriented for an observer looking north, the image plane coordinates were generated by calculating the z-axis, which points south, from two georectangular coordinate vectors calculated from two different terrain data points along the same longitude line. By taking the difference between the X, Y, and Z coordinates a third vector was formed that defined the image plane z-axis. The image plane y-axis was calculated by using only one of the terrain data points used to calculate the z-axis. By taking the negative of the georectangular coordinates of the terrain data point, a vector is produced that points downward through the center of the earth. This was the image plane y-axis. Once the z and y axes are calculated, the cross product of y cross z was used to calculate the x-axis. Figure 3.1 demonstrates the resulting image plane.

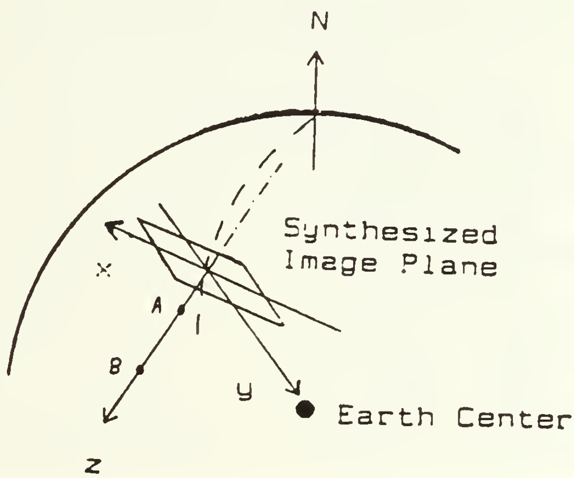


Fig. 3.1 Synthesized Image Plane Coordinates

This image plane coordinate system, from the observers perspective at (0,0,-f), would have the x-axis pointing left, the y-axis pointing down, and the z-axis pointing directly toward the observer. The screen coordinates have the IA axis pointing right and the JA axis pointing down. Therefore, the new values of C1 and C2 with IA, and JA equal to 0 were as follows:

$$C1 = \text{Maximum assigned x-image value} \quad (3.11)$$

$$C2 = 0 \quad (3.12)$$

which aligned the image plane x,y(o,o) to the screen coordinates of IA,JA(0,0). The j, k, l, and m values of the transformation matrix were selected to scale the image plane to the screen.

Having determined the three synthesized image plane coordinate vectors in terms of georectangular vectors, it is relatively easy to generate the M matrix parameters using

$$M = \begin{bmatrix} \dfrac{Sx}{|S|} & \dfrac{Sy}{|S|} & \dfrac{Sz}{|S|} \\ \dfrac{Rx}{|R|} & \dfrac{Ry}{|R|} & \dfrac{Rz}{|R|} \\ \dfrac{Tx}{|T|} & \dfrac{Ty}{|T|} & \dfrac{Tz}{|T|} \end{bmatrix} \quad (3.13)$$

were **S**, **R**, and **T** are the georectangular coordinate vectors of the x, y, and z axes of the rotated image plane. This defines the new transformation matrix that will be used to generate the synthesized view by mapping the georectangular vectors of the terrain data into the new image plane.

B. TERRAIN DATA

The Digital Terrain Elevation Data (DTED) supplied by the DMA came as a rectangular grid of elevation data points. Each elevation data point was recorded as an integer value in meters above sea level. If a particular elevation point was unknown, it was assigned a value of -32767 to assure that it would not be confused with any valid elevation data points. The rectangular terrain grid listed elevation points every one second of a degree change in latitude and longitude. The southwest corner of the terrain grid was defined as being located at 37° 22' 47" N. latitude and -122° 05' 03" W. longitude. From this reference point the elevation data was laid out in 210 rows by 239 columns. The rows represented lines of constant latitude and the columns lines of constant longitude. With this information the northeast corner of the terrain grid was calculated as being located at 37° 26' 17" N. latitude and -122 01'04" W. longitude.

1. Data Verification

The first problem was to compare how accurately the elevation data matched up with the original image data. This required taking specific elevation points that were known to match specific image points, then translating the georectangler coordinates of those elevation points to the I and J coordinates for comparison to the original image. The ω, Φ, and k for the M matrix to transform georectangular to image plane coordinates and the parameters for the affine transform from image plane to original image I and J coordinates were supplied by the DMA with the original image data.

To select specific elevation points for comparison required finding a method of distinguishing unique elevation patterns that could match specific objects in the image. The technique used was to visually display the elevation data as an image. The elevation image file was produced by assigning grey scale values to each elevation data point with the lower elevations receiving the darker shades and the higher elevations the lighter shades. When the elevation image was displayed as shown in Figure 3.2, a distinct highway pattern emerged from which three intersections could reasonably be distinguished. The three intersection points (shown as a, b, and c in Fig. 3.2) correspond to elevated roads that crossed one another in the original image.

Once the elevation points were selected for comparison, the approximate row and column of the elevation data corresponding to the center of the intersections was determined. From the reference point of the terrain grid there is a 1 second change in latitude and longitude for each row and column which allowed the calculation of the latitude and longitude for each of the three reference elevation points. The latitude and longitude for each point was converted to georectangular coordinates using a conversion program, then transformed to I and J coordinates using the provided DMA parameters as explained earlier. Each of the three elevation points mapped to within 10 pixels of the original image in both the I and J coordinate directions. This equates to less than 1 second error in latitude and longitude which was deemed precise enough to establish the correlation between the terrain and image data.

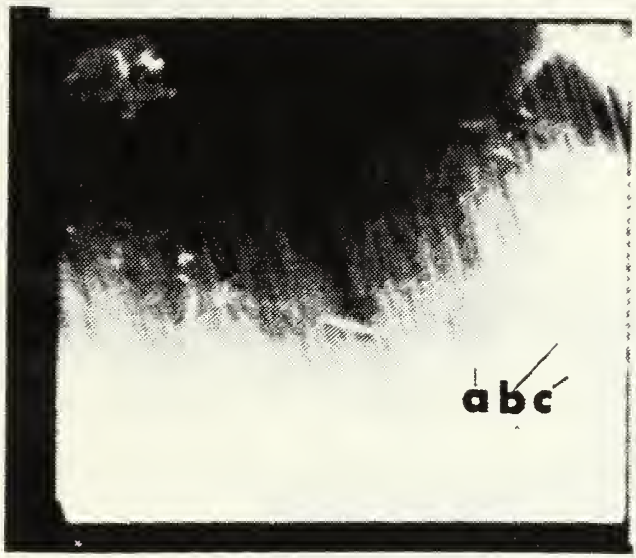


Fig. 3.2 Elevation Image

2. Elevation Line Drawing

By selecting a smaller area of the DTED data, a more distinct picture could be studied. An intersection used to verify image and elevation correlation was selected to be used as the reference image. A smaller rectangular set of terrain data that would map to the intersection, plus a small section of the surrounding area, was extracted from the reference terrain grid. This smaller set of terrain data was taken from rows 71 through 79 and columns 172 through 183 of the terrain model.

To verify that this set of elevation points would appear like the desired reference image, a line drawing illustrated in Figure 3.3 was created using a commercial graphics program called MOVIE.BYU (Ref. 4). This program can generate a connected line drawing from a set of elevation data and allows rotation as well as magnification of the drawing. To assure a sharp visual contrast, the elevation data was magnified by a factor of 10 and the drawing rotated to a useful viewing angle.

The results were not as desired which gave an early indication that the resolution of the terrain data may not be adequate enough to generate a synthesized view that would be a close approximation of the reference image. This was further verified when the synthesized view was produced at a later time. An artificial set of image and elevation data was used to verify that the transformation program functioned as desired before generating synthesized views of the original reference image. The artificial elevation points mapped into the artificial image exactly way as the original elevation data would map to the original reference image. The artificial reference image, shown in Figure 3.4,

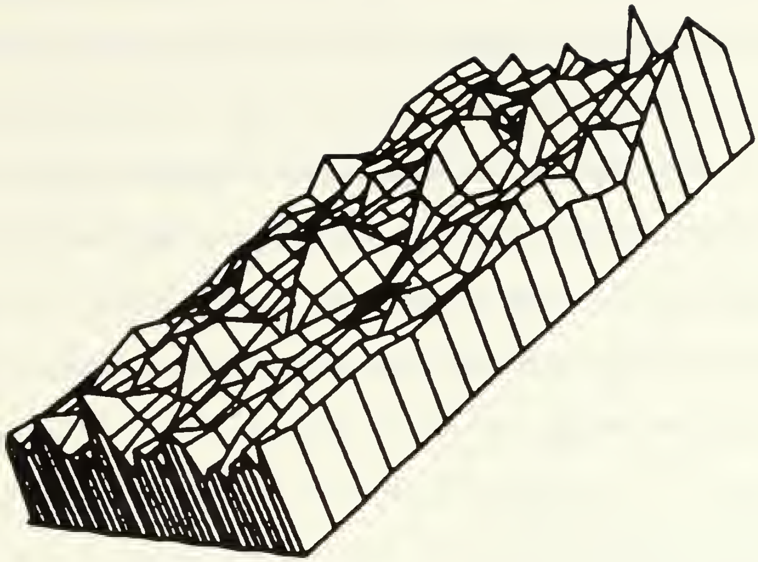


Fig. 3.3 Elevation Line Drawing

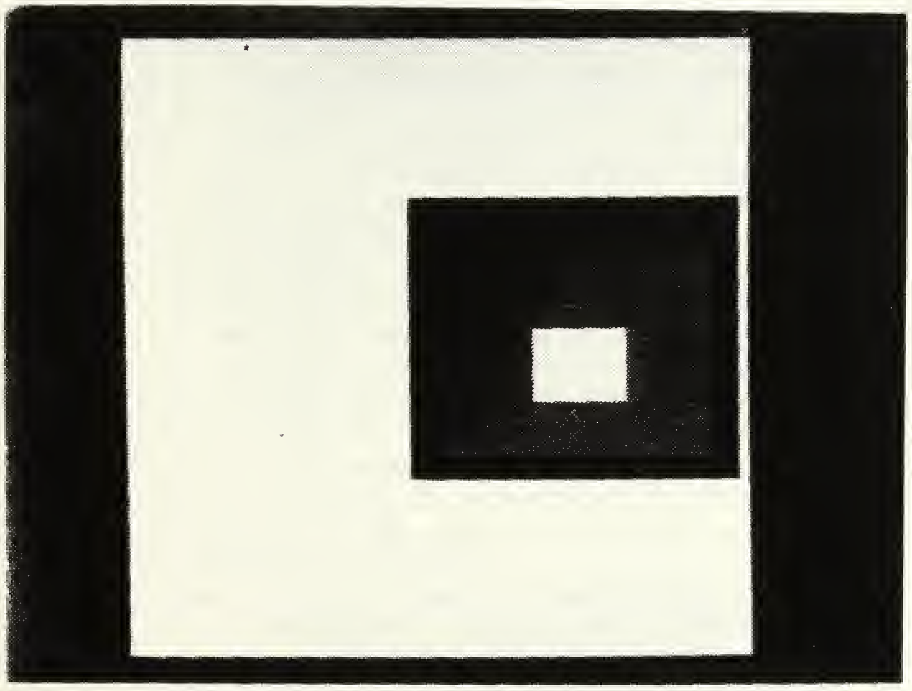


Fig. 3.4 Artificial Reference Image

was of a small square structure resting on top of a much larger square structure. Selecting a large object for an artificial reference image would minimize the effects of the low resolution of the elevation data. This produced more reasonable synthesized images that demonstrated the perspective transformation more accurately. The results of the transformation will be discussed further on in this chapter after considering some of the algorithms used to generate the synthesized view.

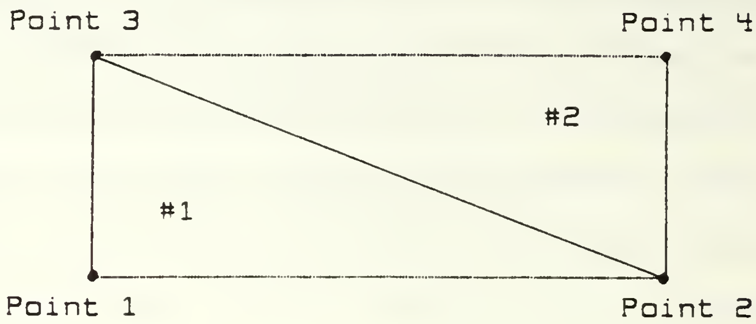
C. SPECIFIC ALGORITHMS

1. Image Referencing Algorithm

Due to the resolution mismatch between the reference image and the terrain data, a smaller subset of the terrain model was selected. This would allow the transformation of smaller and more distinct images that could show the effects of the program better. The desired terrain elevation points are extracted from the larger reference terrain grid by interactively selecting the proper rows and columns that define those particular points. The program allows up to a 50 by 50 array of elevation points to be extracted. This size was chosen to limit the time required to generate a synthesized view. From the rows and columns, the latitude and longitude is tabulated for each point, and is then converted to georectangular coordinates. The georectangular coordinates are written to a file in row order from left to right and from bottom to top. The first set of coordinates match the southwest elevation point, and the last set of coordinates the northeast elevation point.

The georectangular coordinates of each terrain data point is translated to I, and J original image coordinates using the camera parameters supplied by the DMA. They are further processed to IA and JA screen coordinates using the I_Frame and J_Frame of the desired reference image. The IA and JA coordinates derived from the elevation points will now correspond to the IA and JA coordinates of the reference image. Any four adjacent elevation points will define a rectangle which is divided into two triangular panels by defining a line connecting two opposite corners. Once the triangular panels of the elevation points are defined in terms of IA and JA coordinates, the corresponding pixel grey scale values of the reference image that fall within the same screen coordinates of the triangular panel are collected and averaged. As seen in the example of Figure 3.5, the calculated average grey scale value is placed into a file along with the three specific elevation points that make up the triangular panel. This procedure is repeated until the entire terrain grid has been processed. Each triangular panel represents a sample or average grey scale value of the reference image.

When the latitude, longitude, and elevation of the new observation point is input into the program, a new XL, YL, and ZL is calculated that corresponds to the new perspective center. As explained earlier, the image plane is rotated to the desired viewing angle, which changes the M matrix parameters as well.



Triangular Panel	Points	Average Grey Scale
#1	1 2 3	51
#2	2 4 3	62
.	. . .	.
.	. . .	.
.	. . .	.

Fig. 3.5 Triangular Panel Data File Structure

With these new parameters for the perspective transformation equations, the georectangular coordinates of the terrain grid are once again run through the perspective transformation, and the new IA and JA coordinates are calculated for each point. These new IA and JA coordinates represent the transformed elevation points as seen from the new observer location. The next step is to take the same three elevation points that formed a triangular panel in the original transformation, map them into the synthesized image file using the new IA and JA coordinates, and then fill them in with the assigned grey scale value. The mapping and filling process is explained in the next section.

2. Polygon Fill and Hidden Surface Elimination

In the perspective transformation of the terrain grid into the synthesized image plane, some of the triangular panels become partially or entirely hidden by other panels. To determine which pixels will be visible and therefore written to the synthesized image and which pixels are hidden, the z-buffer algorithm was used.

When the new observer location is input into the program, the XL, YL, and ZL georectangular coordinates are tabulated. Using the georectangular coordinates previously calculated for each of the terrain grid elevation points, the distance or depth from the observer location to the elevation data points can be calculated by the following equation

$$\text{Depth} = \text{Square root}((XL-X)^2 + (YL-Y)^2 + (ZL-Z)^2) \quad (3.14)$$

The depth of the pixels within a triangular panel were calculated by using the normalized plane equation that defines the plane of the triangular panel. The normalized plane equation in 3D space is given by

$$ax + by + cz = -1 \quad (3.15)$$

To solve for the coefficients a, b, and c, the three elevation points that specify a triangular panel are used, and the x, y, and z are replaced with IA, JA, and the Depth of each elevation point. If the three points are (IA1, JA1, Depth1), (IA2, JA2, Depth2), and (IA3, JA3, Depth3), then in matrix form we have the following

$$\begin{bmatrix} IA1 & JA1 & Depth1 \\ IA2 & JA2 & Depth2 \\ IA3 & JA3 & Depth3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \qquad (3.16)$$

Solving for the coefficients a, b, and c we have [Ref. 2, p. 208]

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} IA1 & JA1 & Depth1 \\ IA2 & JA2 & Depth2 \\ IA3 & JA3 & Depth3 \end{bmatrix}^{-1} \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \qquad (3.17)$$

The inverse of the elevation point matrix is determined by calculating the adjoint, which is the transpose of the cofactor matrix, and multiplying each term by the reciprocal of the determinant [Ref. 5, p. A-15]. The depth of each pixel within a tranformed triangular panel can now be determined by using the IA and JA of the pixel in the following equation.

$$Depth = -(1 + a(IA) + b(JA))/c \qquad (3.18)$$

Each triangular panel will have different a, b, and c coefficients, therefore the pixels within each triangular panel will have a different depth than those from another panel.

The z-buffer is a two dimensional array that is the same size as the synthesized image of 512 by 512 pixels.

When the IA and JA coordinates of a pixel is determined, the depth is calculated and then compared to any previously written depth in the z-buffer at that same coordinate location. If the depth is smaller, it means that the pixel is closer to the observer and would cover the previously written pixel. The depth of the new pixel will then replace that of the previous pixel. The assigned grey scale value, which is determined from the projected triangular panel in which the new pixel is contained, is written to the synthesized image at the calculated IA and JA coordinate location. If the new pixel depth is larger then the previous pixel depth, that means the new pixel is farther away from the observer and would be covered by the previous pixel. In this case no action is taken and the next pixel is processed. This procedure is continued until all the pixels within each translated triangular panel has been processed. [Ref. 2, pp. 265-267]

The IA and JA coordinates of the pixels in the synthesized image are calculated from each transformed triangular panel using an active edge list (AEL), J_Bucket, and frame buffer. The IA and JA screen coordinates of the three elevation points that define a triangular panel are used to generate the parameters of the AEL. Three lines are formed that connect each of the three translated elevation points and are identified as line 1, 2, and 3. A line is defined between any two points from which you can determine the IA associated with the maximum JA of the two points by

IA_INCPT = (IA value of the maximum JA) (3.19)

How much IA changes for a one step change in JA is given by

$$DELTA\_IA = \left[\frac{IA(Point\ 2) - IA(Point\ 1)}{JA(Point\ 2) - JA(Point\ 1)} \right] \quad (3.20)$$

which is the inverse slope of the line between the two points. The span of JA between two points is given by

DELTA_JA = JA(Point2) - JA(Point 1) (3.21)

For each line, the IA coordinate that corresponds to the maximum JA value of the line, the amount IA changes for each one unit step of JA, and the total span of JA is determined and stored into the AEL. After the line parameters are placed into the AEL, the line identification number is put into the J_Bucket, which is a one dimensional array of size 512, at the same location as the maximum JA of the line. In this way the J_Bucket acts as a pointer to the lines stored in the AEL. If a previous line number has already been written to that location, then the line identification number of the new line is placed into the AEL and is linked to that line already residing in the J_Bucket. An example is shown in Figure 3.6 where line #1 is referenced by the J_Bucket and line #2 is linked to line #1. The IA and JA coordinates of the pixel locations to be mapped to the synthesized image are contained within the three lines whose parameters are contained in the AEL. If a line is horizontal, the IA and JA coordinates of that line are located between the other two lines and therefore does not need to be written to the AEL. [Ref. 2, pp.75-78]

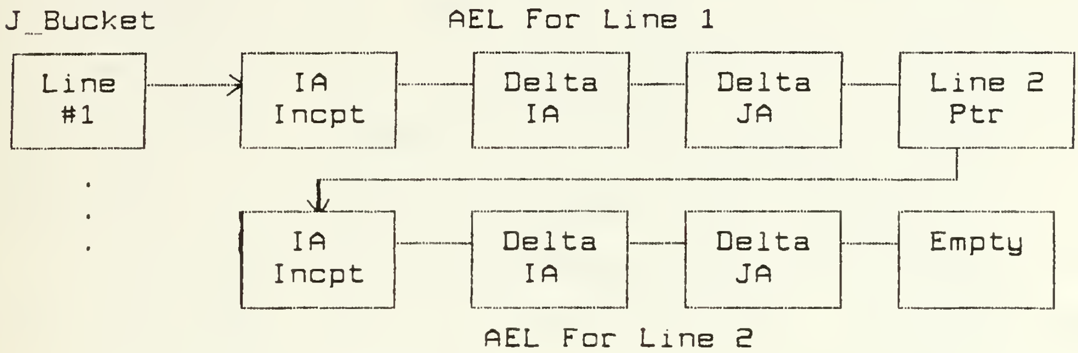


Fig. 3.6 Active Edge List Storage

The J_Bucket is scanned from its maximum to minimum coordinate value. If the J_Bucket contains a line pointer, the parameters for that line are retrieved from the AEL as well as those of any line it is linked to. Since we have defined a triangle there will always be two lines to work with. From the parameters of the two lines, the IA coordinates that fall between them is calculated for each JA scan line. A scan line is all the columns (IA coordinates) along a particular row (JA coordinate). As the JA coordinate is decremented by one, from maximum to minimum, the J_Bucket is checked to see if a new line has been added, and then the corresponding IA for each line is determined for that JA value. Those IA values and any that fall between them are mapped to the frame buffer.

The frame buffer is a 512 by 512 array that is initialized to 0. As the IA and JA values are mapped into the frame buffer the coordinate locations matching the IA and JA values are changed from 0 to 1. This process continues, and each time the JA scan line is decremented, the DELTA_JA parameters for each of the two lines are also decremented and the J_Bucket is checked. If the J_Bucket contains another line pointer, then the line it references will replace the line whose DELTA_JA has decreased to 0. When finished, the frame buffer will have recorded the IA and JA coordinates for every pixel location within the translated triangular panel. The frame buffer is then scanned, and if a particular IA and JA coordinate location contains a value of 1, then the depth for a pixel located at those same coordinates is calculated and compared to previously tabulated depths in the z-buffer as explained earlier.

This process is known as a polygon fill routine that utilizes the z-buffer algorithm for hidden surface elimination. If any part of a triangular panel, after going through the perspective transformation, should map outside the synthesized image coordinate boundary, the entire panel is discarded and the next panel is processed. This is not a satisfactory solution and could have been corrected by implementing a clipping routine that would allow partial triangular panels to be mapped to the synthesized image.

However, due to time constraints, a clipping routine was not incorporated into the 3D transformation program.

D. SUMMARY

Using the artificial reference image from Figure 3.4 and an artificial terrain grid that mapped to the same image, two synthesized views were generated. The first synthesized view shown in Figure 3.7 was from an observation point located at 37° 22' 30" N. latitude, -122° 01' 59" W. longitude, and an elevation of 110 meters. Figure 3.8 exhibits the next synthesized view that was produced from an observation point of 37° 20' 0" N. latitude while maintaining the same longitude and observer height as before. This simulates viewing the object from further away. As expected of a perspective view the object appears smaller. The near view also demonstrates the perspective relationship by the apparent taper from front to back. A third perspective view is depicted in Figure 3.9 from close in and at a higher elevation. The observer location was from 37° 23' 30'' N. Latitude, -122° 01' 59'' W. Longitude, and a height of 160 meters. This demonstrates the visual effect of not displaying partial triangular panels in the image and why a clipping routine is needed.

The actual reference image is shown in Figure 3.10 from which the synthesized view displayed in Figure 3.11 was generated. In comparing the reference image to the synthesized view there is little resemblance. A closer synthesized view is seen in Figure 3.12 which gives a partial representation of the reference image. The lack of detail in both shading and the depiction of features can be attributed to the poor resolution of the terrain data (approximately 25 meter resolution) as compared to that of the reference image data (1 meter resolution).

To improve the quality of the synthesized view the terrain data must be of a higher resolution. Having more elevation data points over a given area, the fewer number of reference image pixels one must collect and average for a triangular panel. The synthesized image will then maintain a closer approximation to the various shades of the reference image. The physical shape of an object also suffers from poor terrain data resolution. The perspective transformation forms straight lines between translated elevation points. This causes object distortion if the elevation points do not fall exactly along the boundaries of the object. As an example, if a square box in the reference image had only one elevation point defined on its surface at the center of the box, then the synthesized image can not reproduce the corners and edges of that box, and it would appear distorted. For these reasons the closer the resolution of the terrain data to the resolution of the reference image, the closer the synthesized view resembles the reference image in shading and shape.

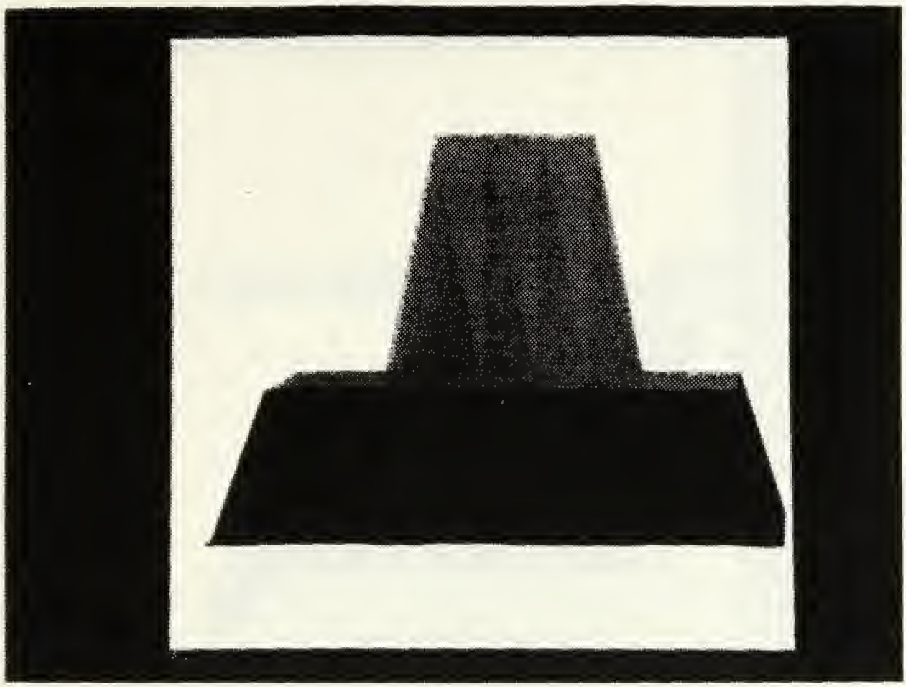


Fig. 3.7 Artificial Synthesized Image 1

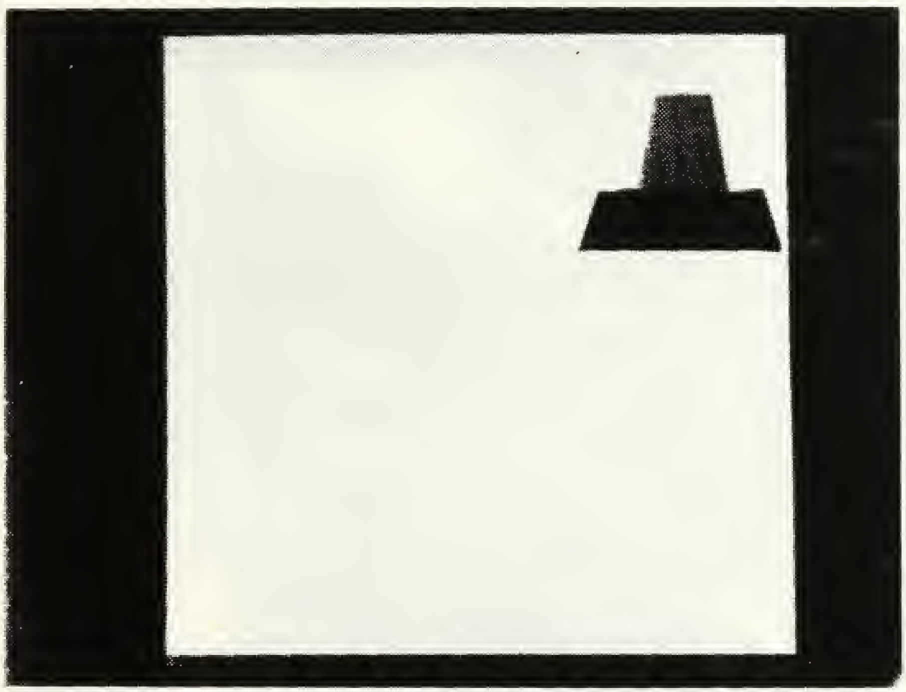


Fig. 3.8 Artificial Synthesized Image 2

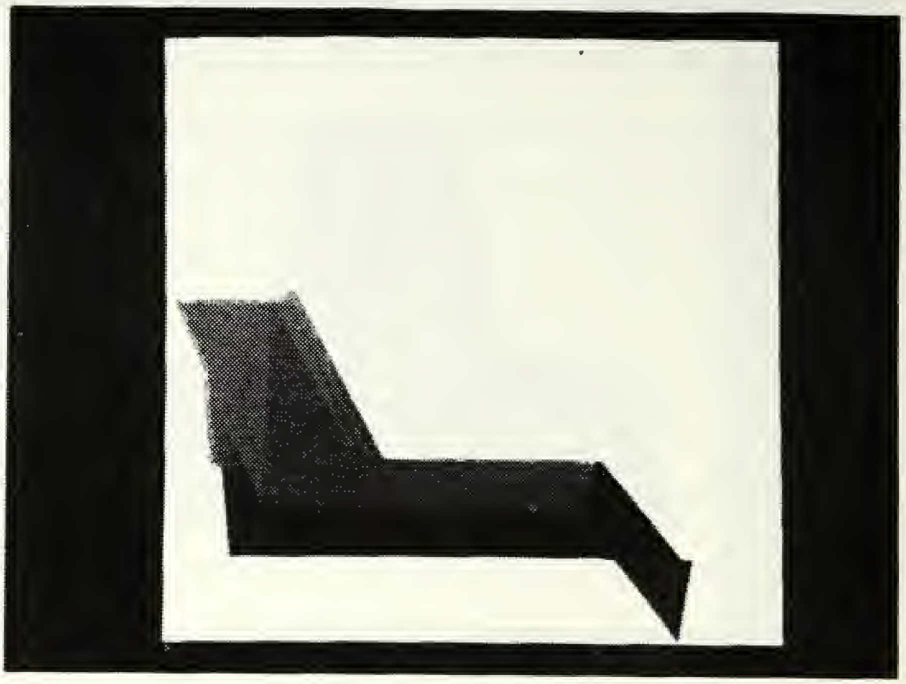


Fig. 3.9 Artificial Synthesized Image 3

Fig. 3.10 Reference Image

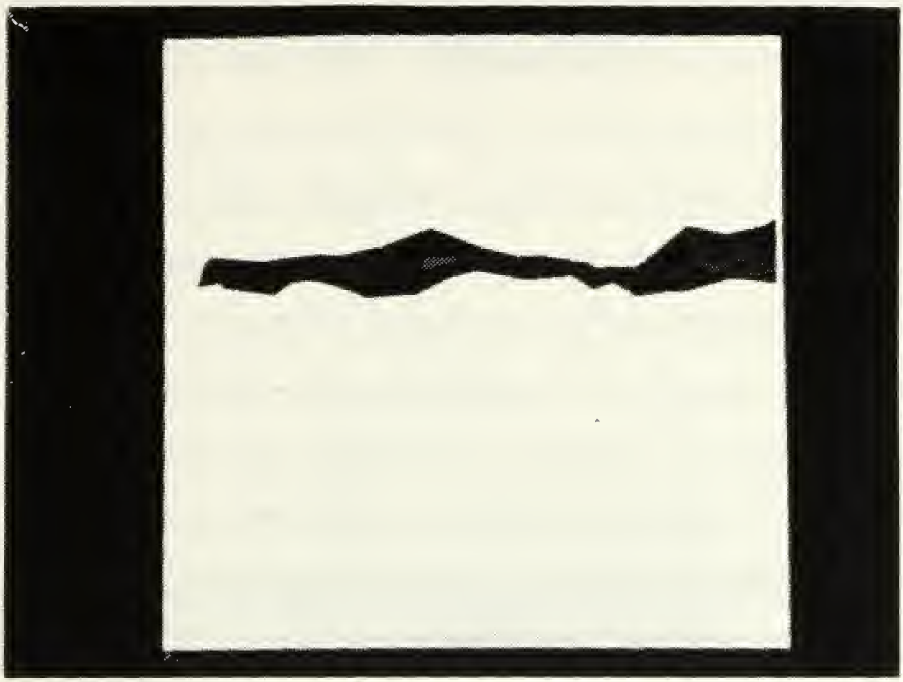


Fig 3.11 Synthesized Image 1

Fig. 3.12 Synthesized Image 2

The 3D transformation program was developed to allow tracing the flow of data easily. Much of the data was written to files so that it could be printed out and studied. This method of data storage required certain files to be read many times as the synthesized image was generated. The program execution would have been faster if the data had been stored in arrays that are easily passed between the various modules. Another consideration that would increase speed would be to decrease or eliminate the interactive input required. This could be accomplished by extracting the desired terrain data into a file and separating the associated reference image before program execution. This would require longer set up time but would decrease the amount of data manipulation required by the program and increase its overall speed.

IV. CONCLUSIONS

A. GENERAL

The 3D computer image transformation from a photographic image was a difficult task to achieve satisfactorily. The main objective was to develop a program that takes a grey scale photographic image, a set of elevation data points defined over that image, and generates a rotated synthesized perspective view. This goal was realized, but some areas still need improvement. The quality of the synthesized view is judged on how well the grey scale values of pixels match those of the original image and how closely the translated objects resemble the desired structure.

The resolution of the terrain model as compared to the resolution of the reference image data, extensively affects the synthesized image quality. Depending on the contents of the reference image, the required resolution of the terrain model will vary. If the image is of open country, the distance between elevation points may be large and the result may not suffer unacceptable degradation in the synthesized view. If the image is of a city that has many small distinct objects such as buildings, then the resolution of the elevation points must be higher. Given a set of elevation data points, the question to be resolved is how to make the synthesized pixels relate to the reference image better, and thus improve the quality of the synthesized views? This question and alternative ways to improve speed and program flexibility will be discussed.

B. GREY SCALE CORRELATION

There are a few possible methods to improve the shading of the synthesized view to better match that of the original image. The translated triangular panels form very distinctive lines or boundaries between areas of different shades. It may be desirable to blend these boundaries by sampling the pixels along both sides, replacing those pixels with an averaged value. Another possibility would be to implement a Gouraud or Phong shading algoritm [Ref. 3, pp. 323-330]. This would help smooth the shade transition across the boundary and result in a more smooth appearance.

Another method to improve grey scale correlation would be to develop a way to divide the triangular panels into smaller triangular areas before referencing the original image. By having smaller triangular panels, smaller areas of the reference image are sampled resulting in a better approximation in the synthesized view.

Only the left image of a stereo photographic pair of images was used in this study. The right image may contain grey scale information of surfaces hidden in the left image view or vice versa. By sampling both left and right images and complementing them, some ambiguities may be resolved. These suggestions will increase the number of calculations required to generate the synthesized view but may improve the quality of the synthesized view.

C. PROGRAM SPEED AND FLEXIBILITY

Although speed was not a prime consideration in this study, it would be desirable to generate synthesized views as quickly as possible. To determine which subroutines consumed the most CPU time the program was monitored while running the artificial reference data set. Table 1 shows the results of the CPU time used and the percentage of the total time for each subroutine called by the main program. If a subroutine calls another subroutine that time is included in the calling routines CPU time. The subroutines that required interactive input were not measured. Therefore, the total CPU time cannot be measured precisely. However, the results obtained represent reasonable estimates and demonstrate where the focus should be on improving overall speed.

Some suggestions have already been discussed on how to improve the speed of the program, but other methods also exist. Preconditioning the input data to eleminate interactive input and using arrays instead of files were two methods already presented. The z-buffer is accessed several times during synthesized view generation. If the z-buffer could be implemented in hardware, the time required for processing of polygon fill and hidden surface elimination routines would be improved.

TABLE 1

CPU TIME CONSUMPTION (IN SECONDS)

SUBROUTINE	CPU TIME	PERCENTAGE(%)
TER_CROP	14.11	9.051
READIMAGE	3.23	2.072
TER_INTRP	0.04	0.026
REAL_EL	0.07	0.045
TER_DMS	0.69	0.443
IM_REFIJ	0.39	0.250
IM_REFAVG	1.10	0.706
AFFIN	0.04	0.026
NEW_IJ	0.80	0.513
NODE_DEPTH	0.63	0.404
FILL	134.80	86.467
TOTAL	155.9	100.0

The frame buffer is used and accessed in two different areas of the program for each triangular panel translated. It may be possible to eliminate this buffer by processing each pixel as its IA and JA coordinates are determined, instead of storing that information into the frame buffer for later processing. Other polygon fill routines such as seed fill algorithms or using fence registers may be faster than the edge fill routine used in this program [Ref. 2, pp. 80-86].

For program flexibility it would be desirable to be able to adjust the image plane of the synthesized view to any desired angle. The program limits the image plane to a northern direction. To make the synthesized image plane adjustable would require developing a method for rotating the image plane coordinates in terms of the georectangular coordinates. This would improve the 3D transformation program and extend its usefulness. Another program improvement would be the incorporation of a clipping routine to improve the appearance of partial synthesized views as discussed in previous sections.

The ideas used to develop this program were contrived from fundamental concepts. Many areas were discussed that could improve or enhance the basic implementation of the program. The results that were obtained are encouraging and could easily be used as the basis for further study.

APPENDIX A

PROGRAM SUMMARY

1. PROGRAM TRAN 3 D

a. Functions performed

Takes an image and elevation data file and extracts the desired region for transformation. Accepts interactive input of a new observer location and generates a synthesized view by making calls to various subroutine modules. This is the main part of the program.

b. Input

Passes parameters between subroutines.

c. Output

An IMAGES file that contains the synthesized view.

d. Calling routines

None.

e. Called routines

INPUT:	Obtains the interactive input of the elevation and image file names and desired rows and columns used to extract elevation data points from the reference elevation grid.
TER CROP:	Extracts the desired elevation points of the area to be translated.

READIMAGE:	Reads the reference image into an array called IMAGE.
TER INTRP:	This routine collects and averages the extracted elevation points and assigns that value to any unknown elevation data points.
REAL EL:	Writes the extracted elevation points into a file called ZFIL.DAT as real values in meters above sea level.
TER DMS:	Determines the latitude and longitude of the elevation points then converts them to georectangular coordinates and stores them in a file called XYZ.DAT.
IM REFIJ:	Converts the georectangular coordinates to the reference image IA and JA screen coordinates.
IM_REFAVG:	Constructs the file NODE.DAT that contains the three elevation points and grey scale value that make up a triangular panel.
AFFIN:	Determines the affine transform coefficients utilized in the transform of the synthesized image

plane coordinates to screen coordinates.

OBS LOC:	Accepts the interactive input of the latitude, longitude, and height in meters above sea level of the desired observer location.
NEW IJ:	Computes the new IA and JA screen coordinates of the elevation points for the synthesized view.
NODE DPTH:	Calculates the distance from each elevation point to the observer location.
FILL:	Determines the hidden surfaces and generates the synthesized image in the file IMAGES.DAT.

f. Routine parameters

IENDN:	The number of rows of the extracted elevation data points.
IENDM:	The number of columns of the extracted elevation data points.

2. SUBROUTINE INPUT

a. Functions performed.

Accepts the interactive input of the elevation and image data files, as well as the information needed to extract the desired elevation points.

b. Input (interactive)

IROW:	Minimum row of the elevation area desired.
LROW:	Maximum row of the elevation area desired.
ICOL:	Minimum column value of the desired elevation area.
LCOL:	Maximum column value of the desired elevation area.
ELFILE:	The reference elevation file name.
IMFILE:	The reference image file name.
IFRAME:	The I_Frame value of the reference image.
JFRAME:	The J_Frame value of the reference image.

c. Output

Same as the interactive input.

d. Calling routines

TRAN_3_D.

e. Called routines

None.

f. Routine parameters

None.

3. SUBROUTINE TER_CROP

a. Functions Performed

Reads the original terrain grid and constructs a smaller grid of the desired elevation points in the array IELEV2.

b. Input

IROW:	Minimum row of the desired elevation area.
LROW:	Maximum row of the desired elevation area.
ICOL:	Minimum column value of the desired elevation area.
LCOL:	Maximum column value of the desired elevation area.
ELFIL:	The file containing the reference elevation data.

c. Output

IELEV2:	An array containing the extracted elevation points.

d. Calling routines

TRAN 3 D.

e. Called routines.

None.

f. Routine parameters

IELEV1:	An array containing the entire elevation data set.
M and N:	Counters.

4. SUBROUTINE READIMAGE

a. Functions performed

Reads the reference image into an array called IMAGE.

b. Input

IMFIL:	The file containing the reference image pixel grey scale data.

c. Output

IMAGE:	An array containing the pixel grey scale values of the reference image.

d. Calling routines

TRAN 3 D.

e. Called routines

None.

f. Routine parameters

IR and IC:	Counters.

5. SUBROUTINE TER_INTRP

a. Functions performed

Determines the average elevation over the extracted elevation area and assigns that elevation to any unknown elevation points.

b. Input

IENDN:	Number of extracted elevation rows.
IENDM:	Number of extracted elevation columns.

IELEV2: An array of the extracted elevation data.

c. Output

IELEV2: Array containing the extracted elevation data (any unknown data points have been assigned the average elevation of the area.)

d. Calling routines

TRAN 3 D.

e. Called routines

None:

f. Routine parameters

N and M: Counters.

NEXT: Count of the number of elevation points processed.

IEL: Summation of the elevations.

IAVG: The average elevation of the area.

6. SUBROUTINES REAL EL

a. Functions performed.

Creates a real file called ZFIL.DAT of the extracted elevation points.

b. Input

IENDN: The number of extracted elevation rows.

IENDM: The number of extracted elevation columns.

IELEV2: An array of the extracted elevation data points.

c. Output

ZFIL.DAT. A file containing the real value of the extracted elevation data points.

d. Calling routines

TRAN 3 D.

e. Called routines

None.

f. Routine parameters

AELEV: A real array of the extracted elevation points.

7. SUBROUTINE TER DMS

a. Functions performed

Converts each extracted elevation data point to its DMS equivalent. It uses the fact that each elevation point represents a one second change in latitude or longitude from the next point, starting from the Southwest corner reference elevation located at 37° 22' 47" N. Latitude and -122° 05' 03" W. Longitude. The DMS is converted to X, Y and Z georectangular coordinates and stored in the XYZ.DAT file.

b. Input

IROW: The minimum row of the extracted elevation data points.

LROW: The maximum row of extracted elevation data points.

ICOL: The minimum column value of the extracted elevation data points.

LCOL: The maximum column value of the extracted elevation data points.

IENDM: The total number of rows of the extracted elevation data points.

ZFIL.DAT: The file containing the real values of the extracted elevation data points.

c. Output

XYZ.DAT: A file containing the georectangular coordinates of the extracted elevation data points.

d. Calling routines

TRAN 3 D.

e. Called routines

DMS2XYZ.

f. Routine parameters

IL and JL: Counters.

X, Y, and Z: Georectangular coordinates of an elevation point.

LATD, LATM, and LATS: The latitude in degrees, minutes, and seconds of an elevation point.

LOND, LONM,

and LONS: The longitude in degrees, minutes, and seconds of an elevation point.

HEIGHT: The real elevation in meters above sea level of an elevation point.

RLATM and RLATS: The reference latitude in minutes and seconds of the reference elevation point located at row 1, column 1.

RLONM AND RLONS: The reference longitude in minutes and seconds of the reference elevation point located at row 1, column 1.

8. SUBROUTINES IM_REFIJ

a. Functions performed

Calculates the I and J reference image coordinates and converts them to the IA and JA screen coordinates for each extracted elevation data point and puts them into arrays IA, and JA.

b. Input

IFRAME: The I Frame value of the reference image.

JFRAME: The J Frame value of the reference image.

IENDM: The number of rows in the extracted elevation data.

IENDN: The number of columns in the extracted elevation data.

XYZ.DAT: Data file containing the georectangular coordinates of the extracted elevation data points.

c. Output

IA: An array containing the IA screen coordinate values of the extracted elevation data points.

JA: An array containing the JA screen coordinate values of the extracted elevation data points.

d. Calling routines

TRAN 3 D.

e. Called routines

PROJECT, XY2IJ.

f. Routine parameters

OMEGA: Rotation of the original image plane about x-axis in radians.

PHI: Rotation of the original image plane about the y-axis in radians.

KAPPA: Rotation of the original image plane about the z-axis in radians.

XO and YO: Offset of the principal point from the IPP.

X1, Y1, and Z1: The georectangular coordinates of the observer location.

A1, A2, B1, B2, C1, and C2: The given affine transform parameters for the transform of the original image plane coordinates to I and J original image screen coordinates.

FOCUS: The camera focal length.

XIMA: The X-axis value of the elevation point in image plane coordinates.

YIMA: The Y-axis value of the elevation point in image plane coordinates.

I and J: The original reference image screen coordinates.

9. SUBROUTINE DMS2XYZ

a. Functions performed

Converts the DMS latitude and longitude data to X, Y, And Z georectangular coordinates.

b. Input

LATD, LATM, and LATS: The latitude in degrees, minutes and seconds of the extracted elevation data points.

LOND, LONM, and

LONS:	The longitude in degrees, minutes, and seconds of the extracted elevation data points.
HEIGHT:	The height in meters above sea level of the extracted elevation data points.

c. Output

X, Y, and Z:	The georectangular coordinates of the extracted elevation points.

d. Calling routines

TER DMS, OBS LOC.

e. Called routines

None.

f. Routine parameters

PHI:	Angle in radians from the Z-axis of the georectangular coordinate system.
LAMDA:	Angle in radians from the X-axis of the georectangular coordinates system.
N, E.SQUARE, and A:	Given parameters used in calculating the georectangular coordinates of the elevation points.
RADIAN:	Number of radians per degree.
PI:	Number of radians in a half circle.

C1:	Number of degrees in a half circle.
C2:	Number of minutes in a degree.
C3:	Number of seconds in a degree.

10. SUBROUTINE PROJECT

a. Functions performed

Converts the X, Y, and Z georectangular coordinates of the extracted elevation data points to the x-image and y-image coordinates of the image plane.

b. Input

OMEGA:	x-axis rotation of the image plane in radians.
PHI:	y-axis rotation of the image plane in radians.
KAPPA:	z-axis rotation of the image plane in radians.
X0 and Y0:	The offset of the principal point from the IPP.
X1, Y1, and Z1:	The georectangular coordinates of the observer location.
X, Y, and Z:	The georectangular coordinates of the extracted elevation data points.
FOCUS:	The camera focal length.

c. Output

XIMA: The image plane x-axis coordinate value of the extracted elevation data points.

YIMA: The image plane y-axis coordinate value of the extracted elevation points.

d. Calling routines

IM PEFIJ.

e. Called routines

None.

f. Routine parameters

M11, M12, M13,

M21, M22, M23,

M31, M32, M33: The parameters of the M matrix.

DENOM: The denominator of the transformation equations.

11. SUBROUTINE XY2IJ

a. Functions performed

Converts the image plane coordinates of the extracted elevation data points to the I and J original image coordinates.

b. Input

XIMA: The image plane x-axis coordinate values of the extracted elevation data points.

YIMA: The image plane y-axis coordinate values of the extracted elevation data points.

A1, A2, B1, B2,

C1, C2: The affine transform parameters to map the image plane coordinates to the reference image I and J coordinates.

c. Output

I and J: The reference image coordinate values for the extracted elevation data points.

d. Calling routines

IM_REFIJ, NEW IJ.

e. Called routines

None.

f. Routine parameters

DENOM: The denominator of the affine transform.

12. SUBROUTINES IM-REFAVG

a. Functions performed

Constructs the file NODE.DAT that contains the elevation points and reference grey scale value that make up a triangular panel.

b. Input

IA and JA: The arrays that contain the reference image screen coordinates of the extracted elevation data points.

IMAGE: The array that contains the reference image grey scale values.

IENDM: The number of rows in the extracted elevation data.

IENDN: The number of columns in the extracted elevation data.

c. Output

NODE.DAT: The file that contains the elevation points that make up a triangular panel and its associated reference grey scale value.

d. Calling routines

TRAN 3 D.

e. Called routines

None.

f. Routine parameters

SLOPE: The slope of the diagonal line that separates the rectangle, defined by four elevation data points, into two triangular panels.

YINT: The Y intercept of the diagonal line.

IY, M, and N:	Counters.
IGREY1:	The averaged reference grey scale value of the triangular panel below the diagonal line.
IGREY2:	The averaged referenced grey scale value of the triangular panel above the diagonal line.
L, L1, and IR:	Counters.
NODE A, NODE B, NODE C, and NODE D:	The numerical designation of the four elevation points that make up the rectangle that is divided into two triangular panels.
ICOUNT1:	The count of the number of pixels averaged in the triangular panel below the diagonal line.
ITOT1:	The summation of the pixel grey scale values averaged in the triangular panel below the diagonal line.
ICOUNT2:	The count of the number of pixels averaged in the triangular panel above the diagonal line.
ITOT2:	The summation of the pixel grey scale values averaged in the

triangular panel above the diagonal line.

13. SUBROUTINE EXT ORIEN

a. Functions performed

Determines the M matrix parameters used in the rotation of the image plane for the synthesized view. This program defines the three image plane coordinates in terms of georectangular coordinates.

b. Input

IENDM:	The number of rows in the extracted elevation data points.
IENDN	The number of columns in the extracted elevation data points.

c. Output

M11, M12, M13:	First row coefficients of the transformation matrix.
M21, M22, M23:	Second row coefficients of the transformation matrix.
M31, M32, M33:	Third row coefficients of the transformation matrix.

d. Calling routines

New IJ.

e. Called routines

None.

f. Routine parameters

MAGN X: The magnitude of the synthesized view image plane x-axis.

MAGN Y: The magnitude of the synthesized view image plane y-axis.

MAGN Z: The magnitude of the synthesized view image plane z-axis.

X, Y, and Z: Georectangular coordinates of the extracted elevation data points.

X_CORD: Array that stores the georectangular X coordinates of the extracted elevation data points.

Y CORD: Array that stores the georectangular Y coordinates of the extracted elevation points.

Z CORD: Array that stores the georectangular Z coordinates of the extracted elevation data points.

X VECX, X_VECY, and X_VECZ: The X, Y, and Z georectangular coordinates of the synthesized image plane x-axis.

Y_VECX, Y_VECY, and Y_VECZ: The X, Y, and Z georectangular coordinates of the synthesized image plane y-axis.

Z_VECX, Z_VECY,

and Z_VECZ: The X, Y, and Z georectangular coordinates of the synthesized image plane z-axis.

ITOT: The total number of extracted elevation points.

IP and IR: Counters.

14. SUBROUTINE AFFIN

a. Functions performed

Assigns or calculates the coefficients to be utilized in the affine transform from image coordinates to screen coordinates of the synthesized view.

b. Input

None.

c. Output

A1, A2, B1, B2

and C1, C2: The affine transform coefficients for the synthesized view image plane coordinates to screen coordinates.

d. Calling routines

TRAN 3 D.

e. Called routines

None

f. Routine parameters

XIMA MAX: Assigned maximum image plane x coordinate.

YIMA MAX: Assigned maximum image plane y coordinate.

I MAX: Assigned maximum IA screen coordinate.

J MAX: Assigned maximum JA screen coordinate.

15. SUBROUTINE OBS LOC

a. Functions performed

Calculates the new observer location georectangular coordinates from the interactive input of the desired latitude, and longitude, and height. This routine also assigns the focal length for the synthesized view.

b. Input (interactive)

LATM and LATS: The minutes and seconds of the latitude of the new observer location.

LONM and LONS: The minutes and seconds of the longitude of the new observer location.

HEIGHT: The altitude in meters above sea level for the new observer location.

c. Output

X1, Y1, and Z1: The georectangular coordinates of new observer location.

FOCUS: Assigned focal length for the synthesized view.

d. Calling routines

TRAN 3 D.

e. Called routines

DMS2XYZ.

f. Routine parameters

None

16. SUBROUTINE NEW IJ

a. Functions performed

Calculates the new IA, and JA screen coordinates of the extracted elevation data points using the new observer location data.

b. Input

X1, Y1 and Z1: Georectangular coordinates of the new observer location.

FOCUS: Assigned focal length for the synthesized view.

A1, A2, B1, B2, and C1, C2: The affine transform coefficients.

IENDM: The number of rows of the extracted elevation data points.

IENDN: The number of columns of the extracted elevation data points.

c. Output

IA: The array containing the elevation data IA screen coordinates of the synthesized view.

JA: The array containing the elevation data JA screen coordinates of the synthesized view.

d. Calling routines

TRAN 3 D.

e. Called routines

XY2IJ, EXT ORIEN.

f. Routine parameters

M11, M12, and M13: First row coefficients of the transformation matrix.

M21, M22, and M23: Second row coefficients of the transformation matrix.

M31, M32, and M33: Third row coefficients of the transformation matrix.

XO, and YO: The offset of the principal point to the IPP.

XIMA: Image plane x coordinate.

YIMA:	Image plane y coordinate.
DENOM:	Denominator of the transformation matrix.
ITOT:	Total number of extracted elevation data points.
IR:	Counter.

17. SUBROUTINE NODE DPTH

a. Functions performed

Calculates the distance from each transformed elevation data point to the new observer location.

b. Input

X1, Y1, and Z1:	The georectangular coordinates of the new observer location.
IENDM:	The number of rows in the extracted elevation data points.
IENDN:	The number of columns in the extracted elevation data points.

c. Output

DEPTH:	Array of the distances from the transformed elevation data points to the new observer location.

d. Calling routines

TRAN 3 D.

e. Called routines

None.

f. Routine parameters

IR:	Counter.
ITOT:	Total number of extracted elevation data points.

18. SUBROUTINE FILL

a. Functions performed

Determines the hidden surfaces using a z-buffer algorithm. The depth is compared for each pixel at a specified screen coordinate to determine if it is written to the synthesized image.

b. Input

IA:	Array containing the IA screen coordinates of the transformed elevation data points.
JA:	Array containing the JA screen coordinates of the transformed elevation data points.
DEPTH:	Array of the distance from the transformed elevation data points to the new observer location.
IMAGE:	Array used to construct the synthesized image.
IENDM:	The number of rows of the extracted elevation data points.
IENDN:	The number of columns of the extracted elevation data points.

XYZ.DAT: File of the georectangular coordinates of the elevation data points.

c. Output

IMAGES: A file containing the synthesized image.

d. Calling routines

TRAN 3 D.

e. Called routines

FRAME FIL.

f. Routine parameters

Z BUFF: The z-buffer

X1, Y1, Z1, X2, Y2, Z2, X3, Y3, and Z3: The screen coordinates and depth of the three elevation points that define a triangular panel.

Z DPTH: The distance of a pixel within a triangular panel to the new observer location.

C11, C12, C13, C21, C22, C23, C31, C32, C33: The cofactors of the plane equation matrix.

DET: The determinate of the plane equation matrix.

A_COEF, B COEF, and C_COEF:	The coefficients of the plane equation.
IGREY:	Grey scale value of a triangular panel.
NODE A, NODE B, NODE C:	Numerical designation of the three elevation data points that make a triangular panel.
I_MIN:	Minimum IA screen coordinate of the three elevation points.
I MAX:	Maximum IA screen coordinate of the three elevation points.
J MIN:	Minimum JA screen coordinate of the three elevation points.
J MAX:	Maximum JA screen coordinate of the three elevation points.
FRAME:	The frame buffer.
IR, I, J, K, and L:	Counters.
IPLANES:	Total number of triangular panels constructed from the extracted elevation data points.

19. SUBROUTINE FRAME_FIL

a. Functions performed

Constructs an edge list from the three transformed elevation points. These edges form the boundaries for a polygon fill routine. The pixels that are determined to fall within the transformed triangle are marked in the frame buffer.

b. Input

NODE A, NODE B, and NODE C:	The numerical designation of the three elevation data points that make a triangular panel.
IA:	Array that contains the IA screen coordinates of the transformed elevation data points.
JA:	Array that contains the JA screen coordinates of the transformed elevation data points.
J_MAX:	Maximum JA screen coordinate of the three elevation data points.
J_MIN:	Minimum JA screen coordinate of the three elevation data points.

c. Output

FRAME:	The frame buffer array.

d. Calling routine

FILL.

e. Called routine

None.

f. Routine parameters

I INCPT: The matching IA coordinate of the maximum JA point of a line.

DELTA I: The amount IA changes for a one step change in JA.

DELTA J: The total JA span of a line.

AEL: Array containing the parameters of the three lines of transformed triangular panel.

XNODE1 and

XNODE 2: Designates the number of IA coordinates between two lines for a given JA.

DX: Indicator used to determine the direction in which the IA coordinates are counted.

NODE: Array containing the numerical designation of the three elevation data points.

NODE1, and

NODE2: Designators for determining the elevation point that contains the highest JA value between two points.

N_HIGH and

N_LOW: Used with NODE1 and NODE2 for determining the highest JA value between two points.

IT, IU, IR

and IS: Counters.

J_BUCKET: Array that contains the line number designators referencing the AEL.

I1 and I2: Used to determine the number of IA coordinates to be written to the frame buffer.

ICNT and LCNT: Counters for the J_BUCKET.

APPENDIX B

3D-TRANSFORMATION PROGRAM LISTING

```
            PROGRAM TRAN_3_D
C
C         THIS PROGRAM TAKES AN IMAGE AND ELEVATION FILE AND
C         CONSTRUCTSTHE REFERENCE IMAGE AND ELEVATION FILE.
C         FROM THESE FILES A SYNTHESIZED IMAGE IS PRODUCED.
C
      CHARACTER ELFILE*13,IMFILE*13
      BYTE IMAGE(512,512)
      INTEGER IELEV2(50,50),IA(2500),JA(2500)
      INTEGER IROW,LROW,ICOL,LCOL,IENDN,IENDM,
      INTEGER IFRAME,JFRAME
      REAL X1,Y1,Z1,FOCUS,DEPTH(2500)
      REAL A1,A2,B1,B2,C1,C2,OMEGA,PHI,KAPPA
C
      CALL INPUT(IROW,LROW,ICOL,LCOL,ELFILE,IMFILE,
     .           IFRAME,JFRAME)
      OPEN(UNIT=1,FILE=ELFILE,STATUS='OLD')
      OPEN(UNIT=2,FILE='ZFIL.DAT',STATUS='NEW',
     .     ACCESS='DIRECT',RECORDSIZE=128,MAXREC=512)
      OPEN(UNIT=3,FILE='XYZ.DAT',STATUS='NEW',
     .     ACCESS='SEQUENTIAL',FORM='FORMATTED')
      OPEN(UNIT=4,FILE=IMFILE,STATUS='OLD',ACCESS='DIRECT',
     .     RECORDSIZE=128,MAXREC=512)
      OPEN(UNIT=20,FILE='NODE.DAT',STATUS='NEW',
     .     ACCESS='DIRECT',RECORDSIZE=128,MAXREC=512)
      OPEN(UNIT=21,FILE='IMAGES.DAT',STATUS='NEW',
     .     ACCESS='DIRECT',RECORDSIZE=128,MAXREC=512)
      CALL TER_CROP(IROW,LROW,ICOL,LCOL,IELEV2)
      IENDN=LROW-IROW+1
      IENDM=LCOL-ICOL+1
      CALL READIMAGE(IMAGE)
      CALL TER_INTRP(IENDN,IENDM,IELEV2)
      CALL REAL_EL(IENDN,IENDM,IELEV2)
      CALL TER_DMS(IROW,LROW,ICOL,LCOL,IENDM)
      CALL IM_REFIJ(IA,JA,IFRAME,JFRAME,IENDM,IENDN)
      CALL IM_REFAVG(IA,JA,IMAGE,IENDM,IENDN)
      CALL AFFIN(A1,A2,B1,B2,C1,C2)
      CALL OBS_LOC(X1,Y1,Z1,FOCUS)
      CALL NEW_IJ(X1,Y1,Z1,FOCUS,A1,A2,B1,B2,C1,C2,
     .            IENDM,IENDN,IA,JA)
      CALL NODE_DPTH(X1,Y1,Z1,DEPTH,IENDM,IENDN)
      CALL FILL(IA,JA,DEPTH,IMAGE,IENDM,IENDN)
      CLOSE(1)
      CLOSE(2)
      CLOSE(3)
      CLOSE(4)
      CLOSE(20)
```

```
      CLOSE(21)
      END
C ***********************************************************
        SUBROUTINE INPUT(IROW,LROW,ICOL,LCOL,ELFILE,IMFILE,
     .                   IFRAME,JFRAME)
C
C       IROW : INITIAL ROW OF DESIRED AREA
C       LROW : LAST ROW OF DESIRED AREA
C       ICOL : INITIAL COLUMN OF DESIRED AREA
C       LCOL : LAST COLUMN OF DESIRED AREA
C       ELFILE : ELEVATION DATA FILE NAME
C       IMFILE : IMAGE DATA FILE NAME
C       IFRAME : I FRAME NUMBER
C       JFRAME : J FRAME NUMBER
      CHARACTER ELFILE*13,IMFILE*13
      INTEGER IROW,LROW,ICOL,LCOL,IFRAME,JFRAME
C
      WRITE(6,*)'INPUT ELEVATION AREA DESIRED'
      WRITE(6,*)'ENTER MINIMUM ROW NUMBER : '
      READ(5,35)IROW
      WRITE(6,*)'ENTER MAXIMUM ROW NUMBER : '
      READ(5,35)LROW
      WRITE(6,*)'ENTER MINIMUM COLUMN NUMBER : '
      READ(5,35)ICOL
      WRITE(6,*)'ENTER MAXIMUM COLUMN NUMBER : '
      READ(5,35)LCOL
35    FORMAT(I3)
      WRITE(6,*)'INPUT THE ELEVATION DATA FILE NAME : '
      READ(5,45)ELFILE
      WRITE(6,*)'INPUT THE IMAGE DATA FILE NAME : '
      READ(5,45)IMFILE
45    FORMAT(A13)
      WRITE(6,*)'INPUT THE I FRAME NUMBER OF IMAGE : '
      READ(5,55)IFRAME
      WRITE(6,*)'INPUT THE J FRAME NUMBER OF IMAGE : '
      READ(5,55)JFRAME
55    FORMAT(I4)
      WRITE(6,*)'***** WAIT APPROX. 1 MINUTE FOR SETUP*****'
      RETURN
      END
C
C ***********************************************************
        SUBROUTINE TER_CROP(IROW,LROW,ICOL,LCOL,IELEV2)
C
C       THIS SUBROUTINE READS THE ORIGINAL TERRAIN GRID
C       AND CONSTRUCTS A SMALLER GRID OF THE ELEVATION
C       POINTS DESIRED.
C
      CHARACTER SWLAT*8,SWLON*8,DELLAT*4,DELLON*4,
      CHARACTER COLS*4,ROWS*4
      INTEGER IELEV1(210,239),IELEV2(50,50),IROW,LROW
      INTEGER ICOL,LCOL,JV(239)
C     THE VALUES OF JV,SWLON,SWLAT DELLON,DELLAT,COLS
```

```
C     AND ROWS ARE IGNORED.
      READ(1,5)SWLON,SWLAT,DELLON,DELLAT,COLS,ROWS
5     FORMAT(1X,2(A8,2X),4(A4,2X))
      DO 20 M=1,238
       READ(1,10)JV(M),(IELEV1(N,M),N=1,210)
10     FORMAT(I6,2x,20I6/(8X,20I6))
20    CONTINUE
      DO 40 N=IROW,LROW
       DO 30 M=ICOL,LCOL
        IELEV2(M+1-ICOL,N+1-IROW)=IELEV1(N,M)
30     CONTINUE
40    CONTINUE
      RETURN
      END
C
C *********************************************************
        SUBROUTINE READIMAGE(IMAGE)
C
C       THIS SUBROUTINE READS THE IMAGE DATA INTO AN ARRAY.
C
      BYTE IMAGE(512,512)
C
      DO 10 IR=1,512
       READ(4,REC=IR)(IMAGE(IC,IR),IC=1,512)
10    CONTINUE
      RETURN
      END
C
C *********************************************************
        SUBROUTINE TER_INTRP(IENDN,IENDM,IELEV2)
C
C       THIS SUBROUTINE DETERMINES THE AVERAGE VALUE OF THE
C       ELEVATION DATA AND ASSIGNS THAT VALUE TO ANY UNKNOWN
C       POINTS
C
      INTEGER IENDN,IENDM,IELEV2(50,50)
      DATA NEXT,IEL,IAVG/0,0,0/
C
      DO 20 N=1,IENDN
       DO 10 M=1,IENDM
        IF(IELEV2(M,N).EQ.-32767)THEN
         GOTO 10
        ELSE
         NEXT=NEXT+1
         IEL=IEL+IELEV2(M,N)
        END IF
10     CONTINUE
20    CONTINUE
      IAVG=IEL/NEXT
C       CHANGE UNKNOWN ELEVATION VALUES TO THE
C       CALCULATED AVERAGE.
      DO 40 N=1,IENDN
       DO 30 M=1,IENDM
```

```
        IF(IELEV2(M,N).EQ.-32767)THEN
         IELEV2(M,N)=IAVG
        END IF
30     CONTINUE
40    CONTINUE
      RETURN
      END
C
C ************************************************************
        SUBROUTINE REAL_EL(IENDN,IENDM,IELEV2)
C
C     THIS SUBROUTINE CREATS A REAL FILE OF THE ELEVATIONS
C
      INTEGER IELEV2(50,50),IENDN,IENDM
      REAL AELEV(239)
C
      DO 20 N=1,IENDN
       K=0
       DO 10 M=1,IENDM
        K=K+1
        AELEV(K)=IELEV2(M,N)
10     CONTINUE
       WRITE(2,REC=N)(AELEV(INT),INT=1,IENDM)
20    CONTINUE
      RETURN
      END
C
C *************************************************************
        SUBROUTINE TER_DMS(IROW,LROW,ICOL,LCOL,IENDM)
C
C       THIS SUBROUTINE CONVERTS EACH ELEVATION POINT TO ITS
C       DMS EQUIVALENT.  IT USES THE FACT THAT EACH ELEVATION
C       POINT REPRESENTS A ONE SECOND CHANGE IN LATITUDE
C       OR LONGITUDE FROM THE NEXT POINT.
C       REFERENCE POINT:  LAT.037 DEG.:22 MIN.:47 SEC.  NORTH
C                         LON.-122 DEG.:05 MIN.:03 SEC.  WEST
C       OUR ELEVATION POINTS STAY WITHIN THE BOUNDS OF 037
C       DEGREES NORTH AND -122 DEGREES WEST, SO THESE VALUES
C       WILL BE ASSIGNED.
C
      IMPLICIT DOUBLE PRECISION (A-Z)
      INTEGER IROW,LROW,ICOL,LCOL,IENDM,IL,JL
      REAL X,Y,Z,LATD,LATM,LATS,AELEV(239)
      REAL LOND,LONM,LONS,HEIGHT
      PARAMETER(RLATM=22.,RLATS=47.,RLONM=5.,RLONS=3.)
C
      LATD=37.
      LOND=-122.
      DO 20 IL=IROW,LROW
       K=IL/60
       LATM=RLATM+K
       LATS=RLATS+(IL-K*60)
        IF(LATS.GE.60.0)THEN
```

```
          LATS=LATS-60.0
          LATM=LATM+1.0
         END IF
        I1=IL-(IROW-1)
        READ(2,REC=I1)(AELEV(INT),INT=1,IENDM)
        I=0
         DO 10 JL=ICOL,LCOL
          K=JL/60
          LONM=RLONM-K
          LONS=RLONS-(JL-K*60)
           IF(LONS.LT.0.0)THEN
            LONM=LONM-1.0
            LONS=LONS+60.0
           END IF
          LONM=-LONM
          LONS=-LONS
          I=I+1
          HEIGHT=AELEV(I)
          CALL DMS2XYZ(LATD,LATM,LATS,LOND,LONM,LONS,
     .                 HEIGHT,X,Y,Z)
          WRITE(3,99)X,Y,Z
99        FORMAT(3(1X,F17.7))
10       CONTINUE
20      CONTINUE
        ENDFILE(3)
        REWIND(3)
        RETURN
       END
C
C ***********************************************************
         SUBROUTINE IM_REFIJ(IA,JA,IFRAME,JFRAME,IENDM,IENDN)
C
C        THIS SUBROUTINE CONSTRUCTS THE I AND J COORDINATE
C        DATA FOR EACH ELEVATION POINT AND THEN CONVERTS THEM
C        TO SCREEN COORDINATES AND STORES THEM IN ARRAYS
C        IA AND JA.
C
       IMPLICIT DOUBLE PRECISION (A-Z)
       REAL OMEGA,PHI,KAPPA,X,Y,Z,X1,Y1,Z1,X0,Y0
       REAL XIMA,YIMA A1,A2,B1,B2,C1,C2,FOCUS
       INTEGER I,J,IENDM,IENDN,IA(2500),JA(2500)
       INTEGER IFRAME,JFRAME,ITOT
       DATA OMEGA,PHI,KAPPA/.8341764,-.4563699,3.0761254/
       DATA X0,Y0/0.000002,0.0/
       DATA X1,Y1,Z1/-2693765.9,-4304520.4,3859018.3/
       DATA A1,A2/20.11323959,-6.022849824/
       DATA B1,B2/6.016207940,20.10938801/
       DATA C1,C2/-34954.59484,-22566.71593/
       DATA FOCUS/0.153197/
C
       ITOT=IENDN*IENDM
       DO 10 IR=1,ITOT
        READ(3,5,END=20)X,Y,Z
```

```
5       FORMAT( 3( 1X,F17.7))
        CALL PROJECT(X,Y,Z,OMEGA,PHI,KAPPA,XO,YO,X1,Y1,Z1,
     .               XIMA,YIMA,FOCUS)
        CALL XY2IJ(XIMA,YIMA,I,J,A1,A2,B1,B2,C1,C2)
        IA( IR)=I-IFRAME
        JA( IR)=4999-JFRAME-J
10    CONTINUE
20    REWIND(3)
      RETURN
      END
C
C ************************************************************
        SUBROUTINE DMS2XYZ(LATD,LATM,LATS,LOND,LONM,LONS,
     .                     LONS,HEIGHT,X,Y,Z)
C
C       THIS SUBROUTINE CONVERTS DMS DATA TO X,Y, AND Z
C       GEORECTANGULAR COORDINATES.
C
      IMPLICIT DOUBLE PRECISION (A-Z)
      REAL PHI,LAMDA,N,X,Y,Z,LATD,LATM,LATS
      REAL LOND,LONM,LONS,HEIGHT
      PARAMETER(PI=3.1415926535 8793238)
      PARAMETER(C1=180.,C2=60.,C3=3600.)
      PARAMETER(E_SQUARE=0.006768658,A=6378206.4)
C
      RADIAN=PI/C1
      PHI=(LATD+LATM/C2+LATS/C3)*RADIAN
      LAMDA=(LOND+LONM/C2+LONS/C3)*RADIAN
      N=A/SQRT(1-E_SQUARE*SIN(PHI)*SIN(PHI))
      X=(N+HEIGHT)*COS(PHI)*COS(LAMDA)
      Y=(N+HEIGHT)*COS(PHI)*SIN(LAMDA)
      Z=(N*(1-E_SQUARE)+HEIGHT)*SIN(PHI)
      RETURN
      END
C
C ************************************************************
        SUBROUTINE PROJECT(X,Y,Z,OMEGA,PHI,KAPPA,XO,YO,X1,
     .                     Y1,Z1,XIMA,YIMA,FOCUS)
C
C       THIS SUBROUTINE CONVERTS THE X,Y,Z GEORECTANGULAR
C       COORDINATES TO XIMA AND YIMA WHICH ARE IMAGE
C       PLANE COORDINATES.
C
      IMPLICIT DOUBLE PRECISION (A-Z)
      REAL M11,M12,M13,M21,M22,M23,M31,M32,M33,DENOM
      REAL XIMA,YIMA,X,Y,Z,OMEGA,PHI,KAPPA
      REAL XO,YO,X1,Y1,Z1,FOCUS
C
      M11=COS(PHI)*COS(KAPPA)
      M12=COS(OMEGA)*SIN(KAPPA)+
     .    SIN(OMEGA)*SIN(PHI)*COS(KAPPA)
      M13=SIN(OMEGA)*SIN(KAPPA)-
     .    COS(OMEGA)*SIN(PHI)*COS(KAPPA)
```

```
      M21=-COS(PHI)*SIN(KAPPA)
      M22=COS(OMEGA)*COS(KAPPA)-
     .    SIN(OMEGA)*SIN(PHI)*SIN(KAPPA)
      M23=SIN(OMEGA)*COS(KAPPA)+
     .    COS(OMEGA)*SIN(PHI)*SIN(KAPPA)
      M31=SIN(PHI)
      M32=-SIN(OMEGA)*COS(PHI)
      M33=COS(OMEGA)*COS(PHI)
C
      DENOM=M31*(X-X1)+M32*(Y-Y1)+M33*(Z-Z1)
      XIMA=X0-FOCUS*(M11*(X-X1)+M12*(Y-Y1)+M13*(Z-Z1))/DENOM
      YIMA=Y0-FOCUS*(M21*(X-X1)+M22*(Y-Y1)+M23*(Z-Z1))/DENOM
      XIMA=XIMA*1000000
      YIMA=YIMA*1000000
      RETURN
      END
C
C *********************************************************
        SUBROUTINE XY2IJ(XIMA,YIMA,I,J,A1,A2,B1,B2,C1,C2)
C
C       THIS SUBROUTINE TAKES THE  IMAGE POINTS XIMA,YIMA
C      AND CONVERTS THEM TO I,J ORIGINAL IMAGE COORDINATES.
C
      IMPLICIT DOUBLE PRECISION (A-Z)
      REAL XIMA,YIMA,A1,A2,B1,B2,C1,C2,DENOM
      INTEGER I,J
C
      DENOM=A1*B2-B1*A2
      I=((XIMA-C1)*B2-(YIMA-C2)*B1)/DENOM
      J=-((XIMA-C1)*A2-(YIMA-C2)*A1)/DENOM
      RETURN
      END
C
C *********************************************************
        SUBROUTINE IM_REFAVG(IA,JA,IMAGE,IENDM,IENDN)
C
C       THIS SUBROUTINE CONSTRUCTS THE FILE THAT IDENTIFIES
C       THE ELEVATION POINTS THAT MAKE UP A TRIANGULAR PANEL
C       AND ITS ASSOCIATED SAMPLED GREY SCALE VALUE.
C
      IMPLICIT DOUBLE PRECISION (A-Z)
      REAL SLOPE,YINT
      BYTE IMAGE(512,512)
      INTEGER IA(2500),JA(2500),IY,M,N,IGREY1,IGREY2,L,L1
      INTEGER IENDM,IENDN,NODE_A,NODE_B,NODE_C,NODE_D,IR
      INTEGER ICOUNT1,ICOUNT2,ITOT1,ITOT2
C
      DO 90 IR=1,IENDN-1
       IL=(IR-1)*IENDM
       DO 80 N=1+IL,IENDM-1+IL
        NODE_A=N
        NODE_B=N+1
        NODE_C=N+IENDM
```

```
          SLOPE=(JA(NODE_C)-JA(NODE_B))*1.0/
     .            (IA(NODE_C)-IA(NODE_B))*1.0
          YINT=1.0*JA(NODE_B)-SLOPE*IA(NODE_B)
          ITOT1=0
          ITOT2=0
          ICOUNT1=0
          ICOUNT2=0
          DO 70 M=IA(NODE_B),IA(NODE_A)
           IY=(SLOPE*M+YINT)
           DO 50 L=JA(NODE_B),IY
            ITOT1=ITOT1+IMAGE(M,L)
            ICOUNT1=ICOUNT1+1
50         CONTINUE
           IF(M.LT.IA(NODE_A))THEN
            DO 60 L=IY+1,JA(NODE_C)
             ITOT2=ITOT2+IMAGE(M,L)
             ICOUNT2=ICOUNT2+1
60          CONTINUE
           END IF
70        CONTINUE
          L1=(N-(IR-1))*2
          IGREY1=ITOT1/ICOUNT1
          IGREY2=ITOT2/ICOUNT2
          NODE_D=NODE_C+1
          WRITE(20,REC=L1-1)NODE_A,NODE_B,NODE_C,IGREY1
          WRITE(20,REC=L1)NODE_B,NODE_D,NODE_C,IGREY2
80       CONTINUE
90      CONTINUE
        RETURN
        END
C
C **********************************************************
          SUBROUTINE EXT_ORIEN(M11,M12,M13,M21,M22,M23,
     .                         M31,M32,M33,IENDM,IENDN)
C
C         THIS ROUTINE DETERMINES THE M MATRIX PARAMETERS
C         FOR ROTATION OF THE IMAGE PLANE TO DESIRED
C         LOCATION FOR VIEWING IN THE SYNTHESIZED IMAGE.
C
        IMPLICIT DOUBLE PRECISION (A-Z)
        REAL MAGN_X,MAGN_Y,MAGN_Z,X,Y,Z
        REAL X_CORD(2500),Y_CORD(2500),Z_CORD(2500)
        REAL X_VECX,X_VECY,X_VECZ,Y_VECX,Y_VECY,Y_VECZ
        REAL Z_VECX,Z_VECY,Z_VECZ
        INTEGER IENDM,IENDN,ITOT,IP,IR
C
        ITOT=IENDM*IENDN
        DO 10 IR=1,ITOT
         READ(3,5,END=20)X,Y,Z
         X_CORD(IR)=X
         Y_CORD(IR)=Y
         Z_CORD(IR)=Z
5        FORMAT(3(1X,F17.7))
```

```
10    CONTINUE
20    REWIND(3)
      Y_VECX=-(X_CORD(1))
      Y_VECY=-(Y_CORD(1))
      Y_VECZ=-(Z_CORD(1))
      IP=ITOT-IENDM+1
      Z_VECX=X_CORD(1)-X_CORD(IP)
      Z_VECY=Y_CORD(1)-Y_CORD(IP)
      Z_VECZ=Z_CORD(1)-Z_CORD(IP)
C       USE THE CROSS PRODUCT OF Y CROSS Z TO OBTAIN
C       THE X VECTOR.
      X_VECX=((Y_VECY*Z_VECZ)-(Y_VECZ*Z_VECY))
      X_VECY=((Y_VECZ*Z_VECX)-(Y_VECX*Z_VECZ))
      X_VECZ=((Y_VECX*Z_VECY)-(Y_VECY*Z_VECX))
      MAGN_Z=SQRT((Z_VECX**2)+(Z_VECY**2)+(Z_VECZ**2))
      MAGN_X=SQRT((X_VECX**2)+(X_VECY**2)+(X_VECZ**2))
      MAGN_Y=SQRT((Y_VECX**2)+(Y_VECY**2)+(Y_VECZ**2))
      M11=X_VECX/MAGN_X
      M12=X_VECY/MAGN_X
      M13=X_VECZ/MAGN_X
      M21=Y_VECX/MAGN_Y
      M22=Y_VECY/MAGN_Y
      M23=Y_VECZ/MAGN_Y
      M31=Z_VECX/MAGN_Z
      M32=Z_VECY/MAGN_Z
      M33=Z_VECZ/MAGN_Z
      RETURN
      END
C
C ************************************************************
        SUBROUTINE AFFIN(A1,A2,B1,B2,C1,C2)
C
C       THIS SUBROUTINE ASSIGNS OR CALCULATES THE
C       COEFFICIENTS TO BE UTILIZED IN THE TRANSFORM FROM
C       IMAGE COORDINATES TO SCREEN COORDINATES.
C
      IMPLICIT DOUBLE PRECISION(A-Z)
      REAL A1,A2,B1,B2,C1,C2,XIMA_MAX,YIMA_MAX,I_MAX,J_MAX
      DATA I_MAX,J_MAX/512.0,512.0/
C
      XIMA_MAX=1600.0
      YIMA_MAX=1600.0
      C1=XIMA_MAX
      C2=0.0
      A2=0.0
      B1=0.0
      A1=-XIMA_MAX/(I_MAX*1.0)
      B2=YIMA_MAX/(J_MAX*1.0)
      RETURN
      END
C
C ************************************************************
        SUBROUTINE OBS_LOC(X1,Y1,Z1,FOCUS)
```

```
C
C        THIS SUBROUTINE CALCULATES THE NEW OBSERVER X1,Y1,Z1
C        LOCATION FROM DESIRED LAT. AND LONG. INPUTS AS WELL
C        AS PROVIDE THE FOCAL LENGTH.
C
      IMPLICIT DOUBLE PRECISION (A-Z)
      REAL LATD,LATM,LATS,LOND,LONM,LONS,HEIGHT
      REAL X1,Y1,Z1,X,Y,Z,FOCUS
C
      LATD=37.0
      WRITE(6,*)'INPUT OBSERVER LATITUDE IN-MINUTES(REAL):'
      READ(5,5)LATM
      WRITE(6,*)'                          -SECONDS(REAL):'
      READ(5,5)LATS
      LOND=-122.0
      WRITE(6,*)'INPUT OBSERVER LONGITUDE IN-MINUTES(REAL):'
      READ(5,5)LONM
      WRITE(6,*)'                           -SECONDS(REAL):'
      READ(5,5)LONS
5     FORMAT(F5.1)
      WRITE(6,*)'INPUT OBSERVER HEIGHT-METERS(REAL): '
      READ(5,10)HEIGHT
10    FORMAT(F6.1)
      FOCUS=0.015
      CALL DMS2XYZ(LATD,LATM,LATS,LOND,LONM,LONS,HEIGHT,
     .             X,Y,Z)
      X1=X
      Y1=Y
      Z1=Z
      RETURN
      END
C
C *************************************************************
        SUBROUTINE NEW_IJ(X1,Y1,Z1,FOCUS,A1,A2,B1,B2,C1,C2,
     .                    IENDM,IENDN,IA,JA)
C
C        THIS SUBROUTINE COMPUTES THE NEW IA AND JA SCREEN
C        COORDINATES FROM THE GIVEN OBSERVER LOCATION.
C
      IMPLICIT DOUBLE PRECISION (A-Z)
      REAL X1,Y1,Z1,FOCUS,M11,M12,M13,M21,M22,M23,M31
      REAL X0,Y0,X,Y,Z,XIMA,YIMA,A1,A2,B1,B2,C1,C2
      REAL M32,M33,DENOM
      INTEGER IENDM,IENDN,ITOT,IR,IA(2500),JA(2500),I,J
      DATA X0,Y0/0.0,0.0/
C
      ITOT=IENDN*IENDM
      DO 10 IR=1,2500
       IA(IR)=0
       JA(IR)=0
10    CONTINUE
      CALL EXT_ORIEN(M11,M12,M13,M21,M22,M23,M31,M32,M33,
     .               IENDM,IENDN)
```

```
      DO 20 IR=1,ITOT
       READ(3,15,END=30)X,Y,Z
15     FORMAT(3(1X,F17.7))
       DENOM=M31*(X-X1)+M32*(Y-Y1)+M33*(Z-Z1)
       XIMA=XO-FOCUS*(M11*(X-X1)+M12*(Y-Y1)+M13*(Z-Z1))/
     .      DENOM
       YIMA=YO-FOCUS*(M21*(X-X1)+M22*(Y-Y1)+M23*(Z-Z1))/
     .      DENOM
       XIMA=XIMA*1000000.0
       YIMA=YIMA*1000000.0
       CALL XY2IJ(XIMA,YIMA,I,J,A1,A2,B1,B2,C1,C2)
       IA(IR)=I
       JA(IR)=J
20    CONTINUE
30    REWIND(3)
      RETURN
      END
C
C ************************************************************
        SUBROUTINE NODE_DPTH(X1,Y1,Z1,DEPTH,IENDM,IENDN)
C
C       THIS SUBROUTINE CALCULATES THE DISTANCE OF THE
C       TRANSLATED ELEVATION POINTS TO THE NEW OBSERVER
C       LOCATION.
C
      IMPLICIT DOUBLE PRECISION(A-Z)
      REAL X1,Y1,Z1,DEPTH(2500)
      INTEGER IENDM,IENDN,IR,ITOT
C
      ITOT=IENDM*IENDN
      DO 10 IR=1,ITOT
       READ(3,5,END=20)X,Y,Z
5      FORMAT(3(1X,F17.7))
       DEPTH(IR)=SQRT(((X1-X)**2)+((Y1-Y)**2)+((Z1-Z)**2))
10    CONTINUE
20    REWIND(3)
      RETURN
      END
C
C ************************************************************
        SUBROUTINE FILL(IA,JA,DEPTH,IMAGE,IENDM,IENDN)
C
C       THIS SUBROUTINE DETERMINES THE HIDDEN SURFACES AND
C       CONSTRUCTS THE TRANSLATED IMAGE. A Z_BUFFER IS USED
C       TO HOLD THE COMPUTED DEPTHS FROM THE OBSERVER TO THE
C       GEOGRAPHIC POSITION. A PLANE EQUATION IS CONSTRUCTED
C       FROM THREE ELEV. POINTS. THIS EQUATION IS USED TO
C       DETERMINE THE DEPTH OF ALL POINTS WITHIN THE PLANE.
C
      IMPLICIT DOUBLE PRECISION(A-Z)
      BYTE IMAGE(512,512)
      REAL DEPTH(2500),Z_BUFF(512,512),X1,X2,X3,Y1,Y2,Y3
      REAL Z1,Z2,Z3,Z_DPTH,C11,C12,C13,C21,C22,C23,C31
```

```
      REAL C32,C33,DET,A_COEF,B_COEF,C_COEF
      INTEGER IGREY,NODE_A,NODE_B,NODE_C,I_MIN,I_MAX,J_MIN
      INTEGER J_MAX,FRAME(512,512),IA(2500),JA(2500),IENDM
      INTEGER IENDN,IR,I,J,K,L,IPLANES
C
C       FIRST DETERMINE THE NUMBER OF PLANES AND INITIALIZE
C       THE Z_BUFFER AND IMAGE TO 0.
C
      IPLANES=((IENDM-1)*2)*(IENDN-1)
      DO 20 K=1,512
       DO 10 L=1,512
        IMAGE(L,K)=0
        Z_BUFF(L,K)=0
10     CONTINUE
20    CONTINUE
C
C       DETERMINE THE COEFFICIENTS OF THE PLANE EQUATION
C       FROM THE THREE ELEVATION POINTS.
C
      DO 70 IR=1,IPLANES
       READ(20,REC=IR)NODE_A,NODE_B,NODE_C,IGREY
       X1=IA(NODE_A)
       Y1=JA(NODE_A)
       Z1=DEPTH(NODE_A)
       X2=IA(NODE_B)
       Y2=JA(NODE_B)
       Z2=DEPTH(NODE_B)
       X3=IA(NODE_C)
       Y3=JA(NODE_C)
       Z3=DEPTH(NODE_C)
C       DETERMINE THE COFACTOR ELEMENTS
       C11=((Y2*Z3)-(Y3*Z2))
       C12=-((X2*Z3)-(X3*Z2))
       C13=((X2*Y3)-(X3*Y2))
       C21=-((Y1*Z3)-(Y3*Z1))
       C22=((X1*Z3)-(X3*Z1))
       C23=-((X1*Y3)-(X3*Y1))
       C31=((Y1*Z2)-(Y2*Z1))
       C32=-((X1*Z2)-(X2*Z1))
       C33=((X1*Y2)-(X2*Y1))
C       CALCULATE THE DETERMINANT
C
      DET=(X1*C11)+(Y1*C12)+(Z1*C13)
C
C       THE COEFFICIENTS ARE DETERMINED FROM MULTIPLYING
C       THE reciprocal of the determenant with the
C       transpose of the cofactors called the adjoint.
C
       A_COEF=-((C11+C21+C31)/DET)
       B_COEF=-((C12+C22+C32)/DET)
       C_COEF=-((C13+C23+C33)/DET)
       IF(C_COEF.EQ.0.0)GOTO 70
C       DETERMINE THE MAXIMUM AND MINIMUM VALUES TO TEST
```

```
C       FOR THE FILL ALGORITHUM.
C
       I_MAX=MAX(IA(NODE_A),IA(NODE_B),IA(NODE_C))
       I_MIN=MIN(IA(NODE_A),IA(NODE_B),IA(NODE_C))
       J_MAX=MAX(JA(NODE_A),JA(NODE_B),JA(NODE_C))
       J_MIN=MIN(JA(NODE_A),JA(NODE_B),JA(NODE_C))
       IF(I_MIN.LT.1.OR.I_MAX.GT.512)GOTO 70
       IF(J_MIN.LT.1.OR.J_MAX.GT.512)GOTO 70
C
C       CLEAR THE REFERENCE FRAME BUFFER AND CALL THE
C       FRAME FILL SUBROUTINE.
C
       DO 40 L=I_MIN,I_MAX
        DO 30 K=J_MIN,J_MAX
         FRAME(L,K)=0
30      CONTINUE
40     CONTINUE
       CALL FRAME_FIL(NODE_A,NODE_B,NODE_C,FRAME,IA,JA,
     .                J_MAX,J_MIN)
       DO 60 J=J_MIN,J_MAX
        DO 50 I=I_MIN,I_MAX
         IF(FRAME(I,J).EQ.1)THEN
         Z_DPTH=-(1+(A_COEF*I)+(B_COEF*J))/C_COEF
         IF(Z_BUFF(I,J).EQ.0.0.OR.Z_DPTH.LT.Z_BUFF(I,J))THEN
           Z_BUFF(I,J)=Z_DPTH
           IMAGE(I,J)=IGREY
          END IF
         END IF
50      CONTINUE
60     CONTINUE
70    CONTINUE
      DO 90 J=1,512
       WRITE(21,REC=J)(IMAGE(I,J),I=1,512)
90    CONTINUE
      RETURN
      END
C
C **********************************************************
        SUBROUTINE FRAME_FIL(NODE_A,NODE_B,NODE_C,FRAME,
     .                       IA,JA,J_MAX,J_MIN)
C
C       THIS SUBROUTINE CONSTRUCTS AN EDGE LIST FROM THE
C       THREE NODES PASSED IN THE ROUTINE.  THESE EDGES
C       ARE USED IN A POLYGON FILL ROUTINE USEING A FRAME
C       BUFFER AND Y_BUCKET
      IMPLICIT DOUBLE PRECISION(A-Z)
      REAL I_INCPT,DELTA_I,DELTA_J,AEL(3,4)
      REAL XNODE1,XNODE2,DX
      INTEGER NODE_A,NODE_B,NODE_C,FRAME(512,512),IA(2500)
      INTEGER JA(2500),J_MAX,J_MIN,NODE(4),IS,NODE1,NODE2
      INTEGER N_HIGH,N_LOW,IT,IU,J_BUCKET(512),IR,I1,I2
      INTEGER ICNT,LCNT
C
```

```
      DO 10 IS=1,512
       J_BUCKET(IS)=0
10    CONTINUE
      DO 30 IS=1,3
       DO 20 IR=1,4
        AEL(IS,IR)=0.0
20     CONTINUE
30    CONTINUE
      NODE(1)=NODE_A
      NODE(2)=NODE_B
      NODE(3)=NODE_C
      NODE(4)=NODE_A
      DO 40 IS=1,3
       NODE1=NODE(IS)
       NODE2=NODE(IS+1)
       IF(JA(NODE1).GE.JA(NODE2))THEN
        N_HIGH=NODE1
        N_LOW=NODE2
       ELSE
        N_HIGH=NODE2
        N_LOW=NODE1
       END IF
       I_INCPT=IA(N_HIGH)
       DELTA_J=(JA(N_HIGH)-JA(N_LOW))
       IF(DELTA_J.EQ.0.0)THEN
        GOTO 40
       ELSE
        DELTA_I=-(IA(N_HIGH)-IA(N_LOW))/DELTA_J
       END IF
       IT=JA(N_HIGH)
       IU=J_BUCKET(IT)
       IF(IU.EQ.0)THEN
        J_BUCKET(IT)=IS
       ELSE
        AEL(IU,4)=IS
       END IF
       AEL(IS,1)=I_INCPT
       AEL(IS,2)=DELTA_I
       AEL(IS,3)=DELTA_J
40    CONTINUE
      IT=J_BUCKET(J_MAX)
      IU=INT(AEL(IT,4))
      XNODE1=AEL(IT,1)
      XNODE2=AEL(IU,1)
      DX=XNODE1-XNODE2
      IF(DX.LE.0.0)THEN
       I1=NINT(XNODE1)
       I2=NINT(XNODE2)
      ELSE
       I1=NINT(XNODE2)
       I2=NINT(XNODE1)
      END IF
      DO 50 IR=I1,I2
```

```
        FRAME(IR,J_MAX)=1
 50    CONTINUE
       AEL(IT,3)=AEL(IT,3)-1.0
       AEL(IU,3)=AEL(IU,3)-1.0
       ICNT=J_MAX-1
       LCNT=J_MIN
       DO 70 IS=ICNT,LCNT,-1
        IF(J_BUCKET(IS).EQ.0)THEN
         XNODE1=XNODE1+AEL(IT,2)
         XNODE2=XNODE2+AEL(IU,2)
        ELSE IF(AEL(IT,3).LE.0.0)THEN
              IT=J_BUCKET(IS)
              XNODE1=AEL(IT,1)
              XNODE2=XNODE2+AEL(IU,2)
             ELSE
              IU=J_BUCKET(IS)
              XNODE1=XNODE1+AEL(IT,2)
              XNODE2=AEL(IU,1)
        END IF
        DX=XNODE1-XNODE2
        IF(DX.LE.0.0)THEN
         I1=NINT(XNODE1)
         I2=NINT(XNODE2)
        ELSE
         I1=NINT(XNODE2)
         I2=NINT(XNODE1)
        END IF
        DO 60 IR=I1,I2
         FRAME(IR,IS)=1
60      CONTINUE
        AEL(IT,3)=AEL(IT,3)-1.0
        AEL(IU,3)=AEL(IU,3)-1.0
70    CONTINUE
      RETURN
      END
C
C ************************************************************
C ************************************************************
```

LIST OF REFERENCES

1. Quam, Lynn H., "The Terrain-Calc System", Proc. SAIC-85/1149, pp.327-330, Dec. 1985.

2. Rogers, David F., Procedural Elements for Computer Graphics, McGraw-Hill, 1985.

3. Moffitt, Francis H., Mikhail, Edward M., Photogrammetry, 3rd Edition, Harper-Row, 1980.

4. Christiansen, H., Stephenson, M., MOVIE.BYU, Community Press, Provo, Utah, 1986.

5. Thomas, G., Finney, R., Calculus and Analytic Geometry, Addison-Wesley Publishing Company, 1984.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Chairman, Code 62 Computer and Electrical Eng. Dept Naval Postgraduate School Monterey, California 93943	1
4.	Professor Chin-Hwa Lee Code 62 Le Computer and Electrical Eng. Dept. Naval Postgraduate School Monterey, California 93943	5
5.	Professor Mitchell L. Cotton Code 62 Cc Computer and Electrical Eng. Dept. Naval Postgraduate School Monterey, California 93943	1
6.	LT. Leland G. Coleman c/o Edward Griffin P.O. 541 Canyon City, Oregon 97820	2

Thesis
C5343 Coleman
c.1 Three-dimensional image generation from an aerial photograph.