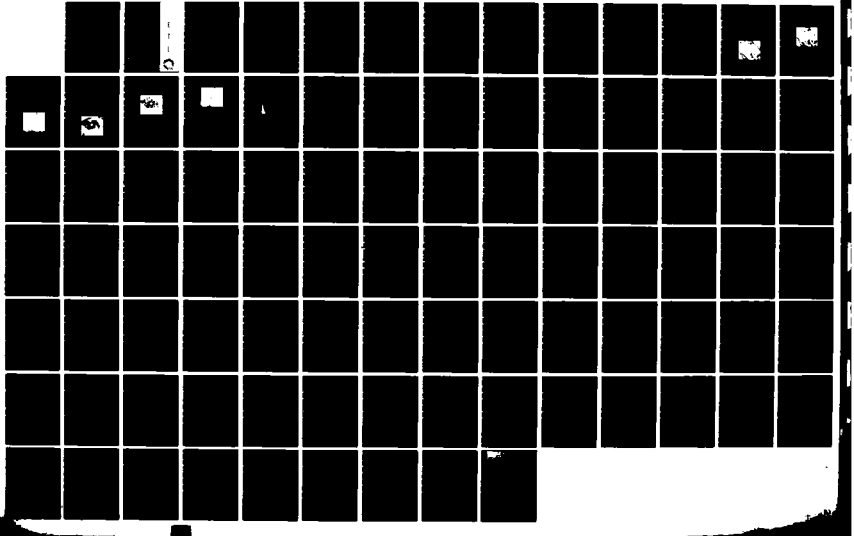
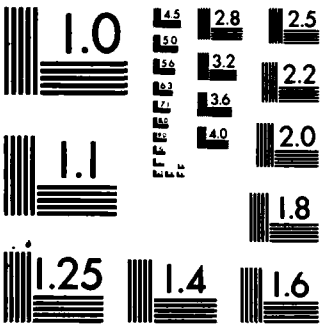
AD-A140 197 INTERACTIVE DIGITAL IMAGE PROCESSING FOR TERRAIN DATA EXTRACTION PHASE 4(U) GENERAL ELECTRIC CO LANHAM MD SPACE SYSTEMS DIV H HEYDT ET AL. NOV 83 ETL-0348 1/1

UNCLASSIFIED DAAK70-79-C-0153 F/G 8/2 NL





ETL-0348

AD-A140197

Interactive digital image processing for terrain data extraction, phase 4

Howard Heydt

Vijay Karkhanis

Thomas Wescott

General Electric Company
Space Systems Division
4701 Forbes Blvd.
Lanham, MD 20706

NOVEMBER 1983

Prepared for

U.S. ARMY CORPS OF ENGINEERS
ENGINEER TOPOGRAPHIC LABORATORIES
FORT BELVOIR, VIRGINIA 22060

84 04 18 023

E T L

U.S. ARMY ENGINEER TOPOGRAPHIC LABORATORIES
ETL

Destroy this report when no longer needed.
Do not return it to the originator.

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

The citation in this report of trade names of commercially available products does not constitute official endorsement or approval of the use of such products.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ETL-0348	2. GOVT ACCESSION NO. AD-A140197	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) INTERACTIVE DIGITAL IMAGE PROCESSING FOR TERRAIN DATA EXTRACTION, PHASE 4		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report Sept 82 - Nov 83 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Mr. Howard Heydt Mr. Vijay Karkhanis Mr. Thomas Wescott		8. CONTRACT OR GRANT NUMBER(s) DAAK70-79-C-0153
9. PERFORMING ORGANIZATION NAME AND ADDRESS General Electric Company Space Systems Division 4701 Forbes Blvd., Lanham, MD 20706		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 4A162707A855 06
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Engineer Topographic Laboratories Fort Belvoir, VA 22060		12. REPORT DATE Nov 1983 13. NUMBER OF PAGES 88
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Digital Image Processing
Terrain Analysis
Feature Classification
Raster to Polygon Conversion
Digital Mosaicking

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This phase of the study concentrated on refinement of vegetation extraction techniques developed in phases 1 through 3. Software was also generated for scaling and mosaicking vegetation data derived from separate adjacent scenes, as appropriate for covering a given map area. The raster format classification files were converted to polygon vector files in the Standard Interface Format.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SUMMARY

The overall Terrain Data Extraction Study is concerned with the development of techniques for the extraction of terrain features from digital aerial imagery. Typical features in the vegetation category are vegetation boundaries and tree stem spacing in forested areas. Extracted data pertaining to features like these are an input to decision-making processes associated with military operations such as cross-country movement. Phase 4 effort, reported here, has built on the base of vegetation feature extraction techniques developed in the previous phases.

Automated delineation (boundary outlines), in a selected digitized aerial panchromatic photograph, has been performed for vegetation/landcover classes, intervals of percent canopy closure, and intervals of tree stem spacing. Images have been produced showing the land-cover class and interval themes, and the theme boundaries have been extracted and recorded on magnetic tape in the Standard Interface Format.

To have utility, terrain data boundaries must be produced for specified map areas normally encompassing multiple frames of adjacent aerial imagery. Algorithms have been developed, therefore, for digital scaling and mosaicking of terrain data derived from separate, adjacent scenes. Software to implement the algorithms has also been produced.

Within a mosaic of multiple aerial images, the terrain data boundaries for any given category are likely to be quite complex. Multiple polygons, polygons with holes, polygons within polygons, etc. are involved, with each polygon containing a large number of vertices. Algorithms have been developed to automatically and systematically examine the terrain data binary themes and to produce the appropriate single-pixel-width outlines (polygons). Procedures also have been developed

to convert the outline data to the Standard Interface Format suitable for input to the ETL Interactive Graphic Design System. These procedures and algorithms have been implemented with appropriate software.

A detailed design for an interactive system capable of performing the above tasks was prepared early in Phase 4 and documented in a separate report entitled Interactive Image Analysis System Design, dated and submitted to USAETL January 1983.

PREFACE

This document has been generated under Contract DAAK70-79-C-0153 for the U.S. Army Engineer Topographic Laboratories, Fort Belvoir, Virginia 22060 by the General Electric Company, Space Systems Division, Lanham, Maryland 20706. The Contracting Officer's Technical Representative was Mr. R Tynes.

A-1

TABLE OF CONTENTS

ILLUSTRATIONS

INTERACTIVE DIGITAL IMAGE PROCESSING FOR TERRAIN DATA EXTRACTION, PHASE 4

1 INTRODUCTION

This report documents the results of Phase 4 of the Terrain Data Extraction study. Phase 4 effort was performed during the period September 1982 to November 1983. The overall study addresses the problem of extraction, from digital aerial imagery, of terrain features essential for decisions associated with military operations such as cross-country movement. Also addressed in the study are some aspects of the generation of terrain analysis products (e.g., maps) necessary for the decision-making associated with these military operations. Primary effort in the overall study prior to Phase 4 has focused on the extraction, from panchromatic aerial photography, of vegetation and other land-cover boundaries, and on the extraction of forest features such as canopy closure and stem spacing. See the Phase 1 report ETL-0241, November 1980 for a more complete description.

Phase 4 involved four tasks:

(1) Produce automated delineation (boundary outlines), based on extraction techniques developed in Phases 1-3, of:

- o Vegetation classes
- o Percent canopy closure intervals
- o Tree stem spacing intervals

(2) Develop algorithms and provide software for scaling and mosaicking vegetation data derived from separate adjacent scenes, as appropriate for covering a given map area.

(3) Develop algorithms and provide software for converting area boundary (outline) data to a format suitable for input to the ETL Interactive Graphic Design System for use in polygon manipulation programs.

(4) Provide a detailed design for an interactive system capable of performing the tasks (1), (2) and (3), above.

Task (4) was completed early in Phase 4 and documented in a report submitted to USAETL dated January 1983 and entitled Interactive Image Analysis System Design. The results of Tasks (1), (2) and (3) are documented in this report in Sections 2, 3 and 4 which follow. Software developed in these tasks is not included in this report but has been delivered separately to USAETL. In Task (3) not only was the polygon conversion problem addressed, but attention was given also to the automatic extraction of polygon boundaries for a complex theme representing, for example, all forest areas in a large scene (i.e., multiple polygons, polygons with holes, polygons within polygons, etc.).

2 DELINEATION OF VEGETATION CLASSES, CANOPY CLOSURE AND STEM SPACING

Image products depicting automated delineation of the following have been produced using extraction techniques which have been developed in this study:

- Boundary Outlines of Vegetation Classes
- Boundary Outline of Forest Canopy Theme According to Percent Canopy Closure
- Boundary Outline of Forest Areas According to Average Tree Stem Spacing.

Figure 2-1 shows the basic digital aerial image from which the above vegetation feature outlines were extracted.

Figure 2-1. Digital Aerial Image

Vegetation themes were produced from the basic image using techniques developed in this project and described previously. In this particular scene only three broad classes exist: Forest, Grass and Bare (includes roads and buildings). These themes have been color coded in blue, cyan and white respectively, and superimposed on the basic image. See Figure 2-2.

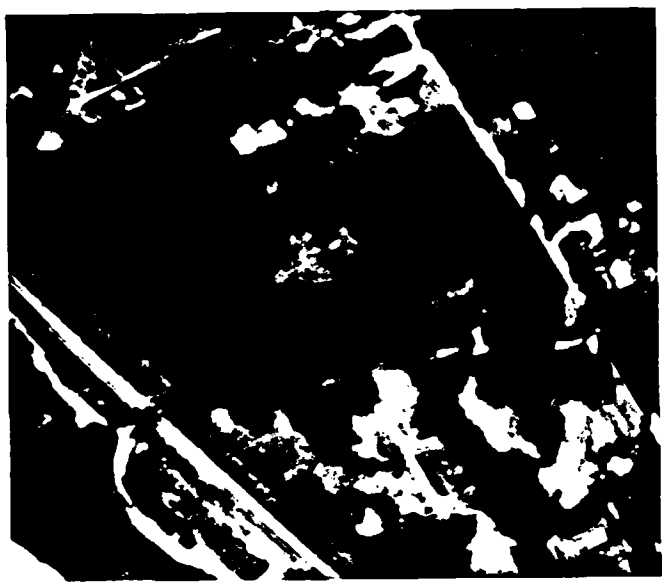


Figure 2-2. Scene With Vegetation Themes Superimposed

A forest canopy theme was extracted from the basic digital image and produced using techniques previously developed and described in this study. A wide-area moving average is applied to this two-level canopy map. Averaging is performed over approximately 0.1 hectare.

The result is a gray-level canopy closure map in which pixel brightness is proportional to percent canopy closure. In turn, that map is level sliced into five equal pixel brightness intervals, and a color-coded theme is produced for each interval. Figure 2-3 shows these themes superimposed on the original digital image. Color coding is as follows:

Canopy Closure Interval	Color Code
0 - 20%	Red (Reddish-gray when superimposed on image)
20 - 40%	Cyan
40 - 60%	Blue
60 - 80%	Green
80 - 100%	Purple

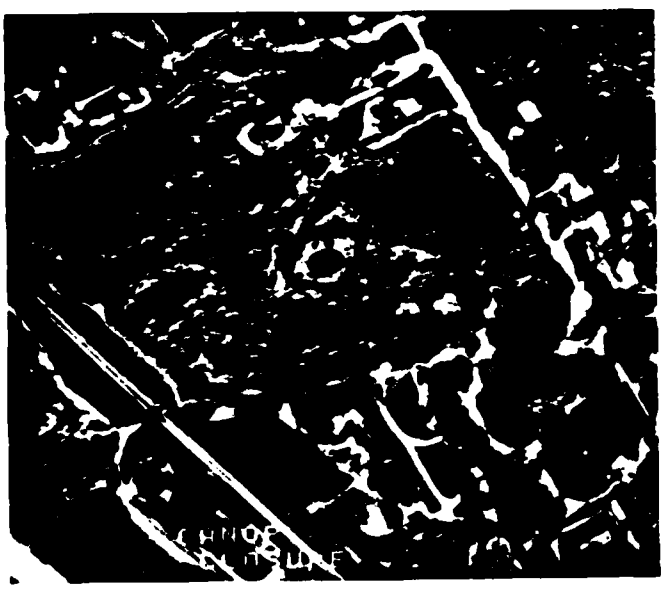


Figure 2-3. Scene With Canopy Closure Themes Superimposed

Tree stem density themes are produced from the basic digital image by first producing a binary map in which single pixels representing tree crown locations are given a value of "1", with "0" for the remainder of the image. This crown location theme is extracted from the basic image using the crown template and processing procedure developed and described previously in this study. Next, the crown (stem) theme is modified such that each crown location pixel becomes a 3 x 3 pixel array, with each pixel having an intensity value of 163. This modified theme is now subjected to a 27 x 27 pixel (729 pixels) moving average. A gray level image results (see Figure 2-4) in which pixel brightness is proportional to stem or crown density. Since 27 x 27 pixels correspond to 22.1 x 22.1 meters^2 in this image, a pixel brightness of $\frac{9 \text{ pixels} \times 163}{27 \times 27 \text{ Pixels}}$ (or ≈ 2) corresponds to an area per tree of 22.1 x 22.1 = 488 meters^2.

Figure 2-4. Moving Average Applied to Tree Crown Theme

The gray level image resulting from the 27 x 27 pixel moving average is level sliced at appropriate digital levels so as to produce themes representing intervals of average tree area or stem spacing. These themes, plus the 3 x 3 pixel crown locations, are shown in Figure 2-5.Theme color codes, and the stem

Figure 2-5. Average Stem Spacing Themes With Crown Location Theme

spacing intervals which they represent, are as follows:

Theme Color Code	Stem Spacing Interval
Yellow	6.7 - 8.2 Meters
Blue	8.2 - 10.1
Green	10.1 - 15.0
Red	15.0 - 33.6
Black	> 33.6

The stem spacing interval themes (except for black) are shown superimposed on the original image in Figure 2-6. To arrive at the calibration of average stem

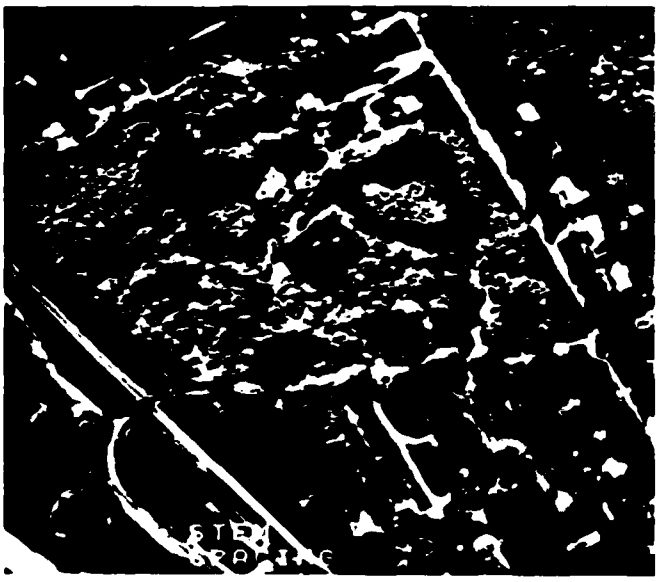
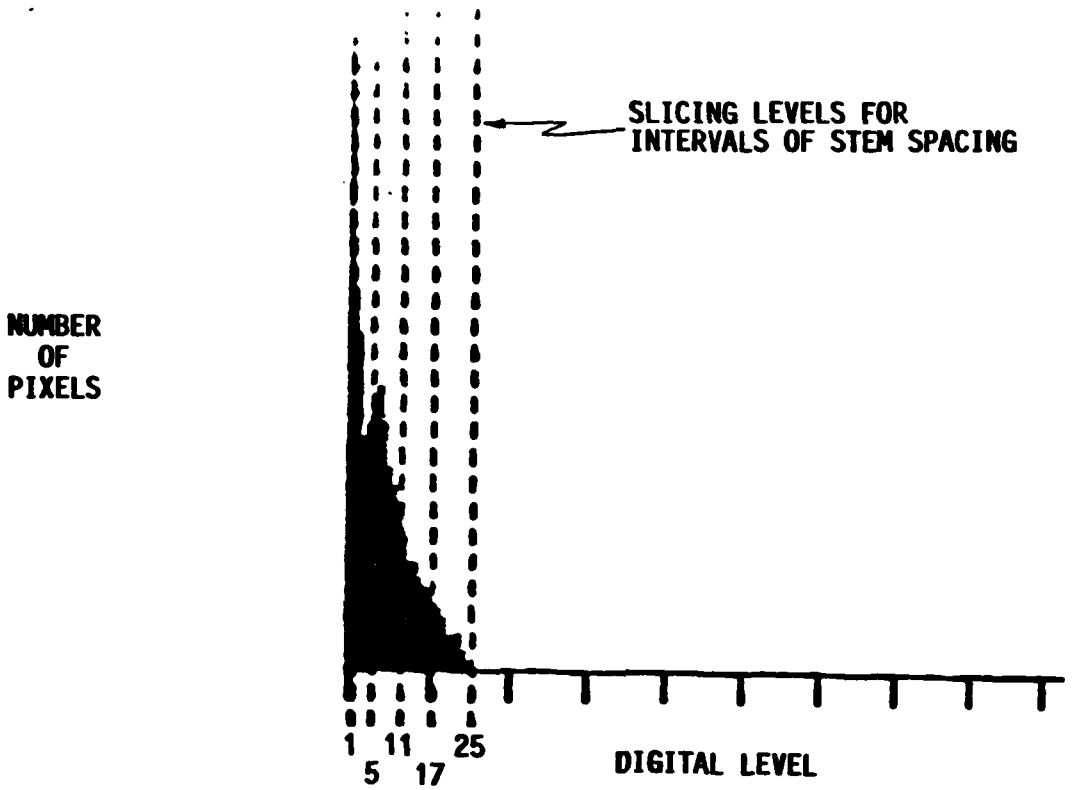


Figure 2-6. Scene With Stem Spacing Interval Themes Superimposed

spacing vs digital level number in the gray level image of the averaged crown location theme, it is assumed that average tree area is inversely proportional to digital number, or $A = \frac{k}{DN}$. Since DN = 2 corresponds to 488M², $A = \frac{977}{DN}$. Using a hexagonal tree model (see Figure 2-7) it can be shown that stem spacing $S = 2h = 1.075A^{1/2}$. Thus $S = 33.6\ (DN)^{-1/2}$. Average stem spacing for five selected digital level numbers are tabulated in Figure 2-7. The figure also shows the locations of those digital levels in the histogram of the image of the averaged crown location theme.

Finally, the three digital images of themes for vegetation classes, canopy closure intervals and stem spacing intervals were processed to produce new themes representing the boundaries of the original polygon themes. These polygon boundary themes have been converted to the SIF format and loaded onto three digital tapes.



DN DIGITAL LEVEL NO.	S STEM SPACING
1	33.6M
5	15.0
11	10.1
17	8.2
25	6.7

$$S = 33.6\ (DN)^{\frac{1}{2}}$$

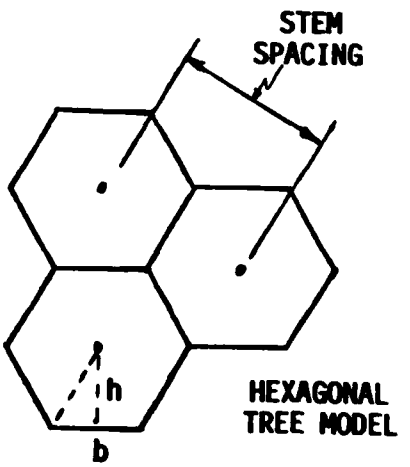


Figure 2-7. STEM SPACING

3 IMAGE MOSAICKING

3.1 INTRODUCTION

A mosaic is an assembly of individual photographs fitted together systematically to form a composite view of the entire area covered by the photographs. The mosaic gives the appearance of a single photograph, producing a complete record of the area photographed Perfect matching of the images of adjoining photographs is virtually impossible because of variations in altitude of the airplane, large differences in elevation of the ground, divergence of the camera axis from the vertical at the time of exposure, and errors inherent in the photographic system of camera, lens, film, print, paper,etc. By the use of proper techniques, however, these errors can be held to a minimum and not permitted to accumulate.

To construct an accurate mosaic, all individual photographs must be reduced to a common scale, must be rectified to remove the distorting effects of camera tilt, and must be assembled to accurate ground control at selected photo-identifiable points for control of azimuth, scale, and position.

Mosaics are very useful in all types of planning activities and have the advantage of showing as a single picture a wealth of detail of the entire area under study.

3.2 DISPLACEMENT OF IMAGES DUE TO TILT

As a result of tilt alone, the images appear to be displaced radially toward the isocenter on the upper side of the photograph and radially outward or away from the isocenter on the lower side. Along the isometric parallel (line through the isocenter perpendicular to the direction of tilt) there is no displacement relative to an equivalent untilted photograph.

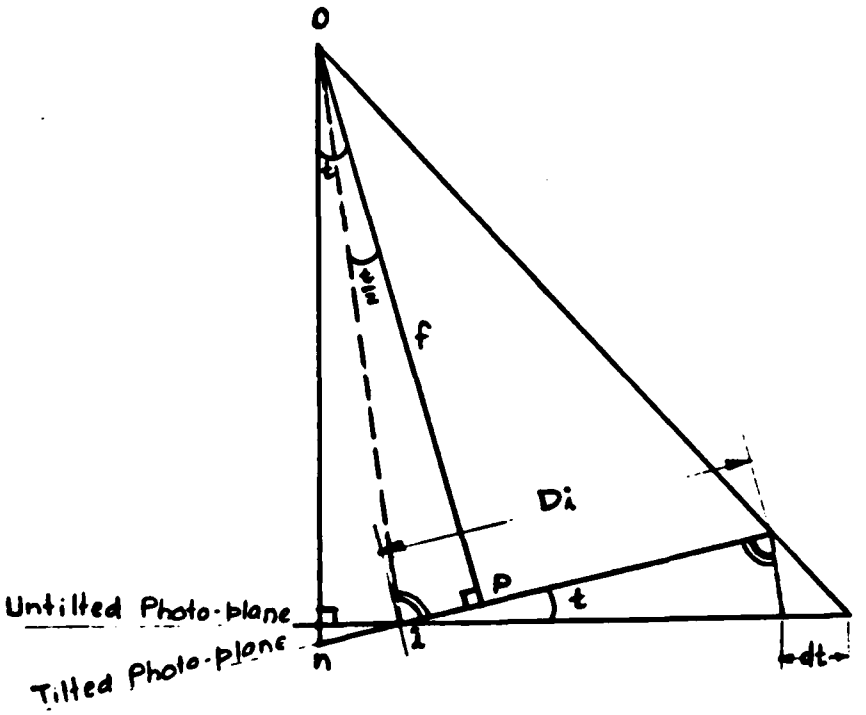


Figure 3-1. Image displacement due to tilt

The Figure 3-1 illustrates the geometry of an aerial photograph and the symbols have the following interpretation:

O = Location of camera lens

OP = Optical axis which is perpendicular to the plane of the photograph at the principal point P (geometric center). This is equal to the focal length of the camera.

t = Tilt angle

i = Isocenter

n = Nadir

Di = Distance of image from the isocenter along the principal line

f = Camera focal length

dt = Displacement of image relative to the equivalent untilted photograph

Then the magnitude of displacement dt is given by the equation

$$dt = \frac{Di^2}{(f/\sin(t) - Di)}$$

$$\approx Di^2 \cdot t/f$$

when t is very small and expressed in radians.

3.3 PROJECTIVE TRANSFORMATION EQUATIONS

Assuming that light rays travel in straight lines, that all the rays entering a camera lens system pass through a single point and that the lens system is distortionless, then a projective relationship exists between the photographic coordinates of the image points and the ground coordinates of the corresponding object points as illustrated in Figure 3-2.

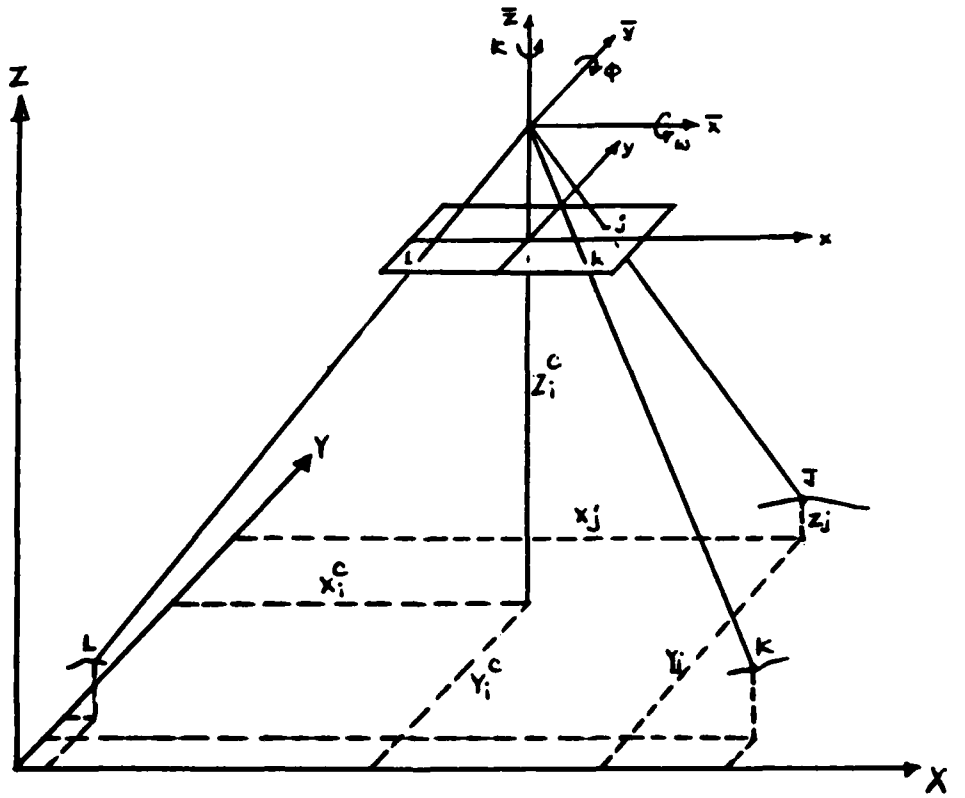


Figure 3-2. Exterior Orientation

where

X, Y, Z = Three-dimensional rectangular coordinate system for the object space

X_j, Y_j, Z_j = The coordinates of any point j in the object space

X_i^c, Y_i^c, Z_i^c = The position of the exposure center of photograph i

x, y, z = The photo-coordinate system

w, ϕ, k = Rotation angles about $\bar{x}, \bar{y}, \bar{z}$ respectively which define the direction of the optical axis. The angles are positive in counter-clockwise direction. When $w=\phi=k=0$, then $\bar{x}, \bar{y}, \bar{z}$ frame is parallel to the X, Y, Z frame and the optical axis is perpendicular to the X-Y plane.

Then, the projective transformation equations can be written down as

$$X_j - X_i^c = \lambda_{ij}\left[m_{11}\cdot(x_{ij}-x_p) + m_{21}\cdot(y_{ij}-y_p) + m_{31}(-f)\right]$$

$$Y_j - Y_i^c = \lambda_{ij}\left[m_{12}\cdot(x_{ij}-x_p) + m_{22}\cdot(y_{ij}-y_p) + m_{32}(-f)\right]$$

$$Z_j - Z_i^c = \lambda_{ij}\left[m_{13}\cdot(x_{ij}-x_p) + m_{23}\cdot(y_{ij}-y_p) + m_{33}(-f)\right]$$

where

$$m_{11} = \cos(\phi_i)\cdot\cos(k_i)$$

$$m_{12} = \cos(\omega_i)\sin(k_i) + \sin(\omega_i)\sin(\phi_i)\cos(k_i)$$

$$m_{13} = \sin(\omega_i) \cdot \sin(k_i) - \cos(\omega_i) \cdot \sin(\phi_i) \cdot \cos(k_i)$$

$$m_{21} = -\cos(\phi_i) \cdot \sin(k_i)$$

$$m_{22} = \cos(\omega_i) \cdot \cos(k_i) - \sin(\omega_i) \cdot \sin(\phi_i) \cdot \sin(k_i)$$

$$m_{23} = \sin(\omega_i) \cdot \cos(k_i) - \cos(\omega_i) \cdot \sin(\phi_i) \cdot \sin(k_i)$$

$$m_{31} = \sin(\phi_i)$$

$$m_{32} = -\sin(\omega_i) \cdot \cos(\phi_i)$$

$$m_{33} = \cos(\omega_i) \cdot \cos(\phi_i)$$

x_{ij}, y_{ij} = Image coordinates of an object point J on photo i

λ_{ij} = The photo scale factor at image point J on photo i
= in the newly rotated photo-coordinate system about
= the $\bar{x}, \bar{y}, \bar{z}$ axis of the photo.

When $\omega=\phi=k=0$, then referring to photo-coordinate system as $\bar{x}_o, \bar{y}_o, \bar{z}_o$ the following relationship exists:

$$X_j - X_i^c = \lambda_{ij} \bar{x}_o$$

$$Y_j - Y_i^c = \lambda_{ij} \bar{y}_o$$

$$Z_j - Z_i^c = \lambda_{ij} \bar{z}_o$$

In this case, $\bar{x}_o$, $\bar{y}_o$, and $\bar{z}_o$ are parallel to X, Y, and Z respectively.

3.3.1 TILT, SWING, AND AZIMUTH

The orientation of a photograph can also be defined by the three rotation angles called tilt(t), swing(s), and azimuth(α) as illustrated in Figure 3-3.

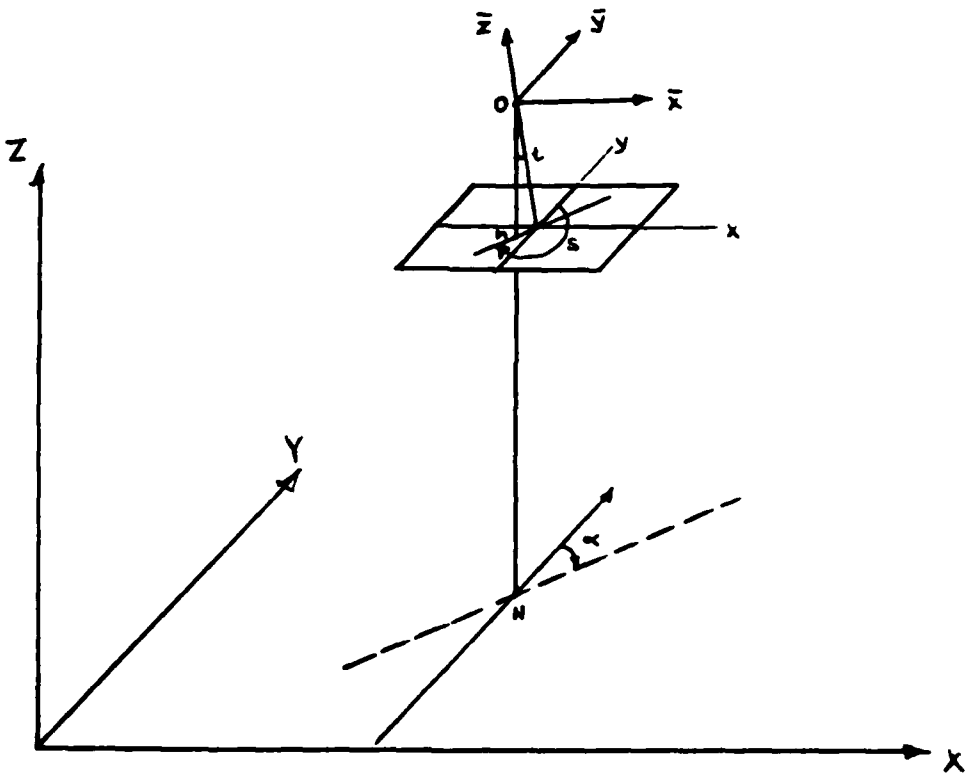


Figure 3-3. Orientation in terms of tilt, swing, and azimuth

The tilt angle (t) is measured in the principal plane from the optical axis of the camera to the plumb line and always has a positive sign. The direction of tilt with respect to the photographic axes is defined by the swing angle (s), which is measured clockwise from the positive y-axis of the photograph to

the lower part of the principal line which passes through the nadir point (n). The direction of tilt with respect to the ground reference coordinate system (X, Y, Z) is defined by the azimuth angle (α) which is measured in the X-Y plane clockwise from the positive Y axis to the projection of the X-Y plane.

The elements in the projective transformation equation may be expressed in terms of t, s, and α as follows:

$$m_{11} = -\cos(s)\cdot\cos(\alpha) - \sin(s)\cdot\cos(t)\cdot\sin(\alpha)$$

$$m_{12} = \sin(s)\cdot\cos(\alpha) - \cos(s)\cdot\cos(t)\cdot\sin(\alpha)$$

$$m_{13} = -\sin(\alpha)\cdot\sin(t)$$

$$m_{21} = \cos(s)\cdot\sin(\alpha) - \sin(s)\cdot\cos(t)\cdot\cos(\alpha)$$

$$m_{22} = -\sin(s)\cdot\sin(\alpha) - \cos(s)\cdot\cos(t)\cdot\cos(\alpha)$$

$$m_{23} = -\cos(\alpha)\cdot\sin(t)$$

$$m_{31} = -\sin(s)\cdot\sin(t)$$

$$m_{32} = -\cos(s)\cdot\sin(t)$$

$$m_{33} = \cos(t)$$

A major disadvantage of this system of defining orientation is that it breaks down for the ideal aerial photograph which is truly vertical. When there is no tilt, the two planes are either parallel or coincident, and there is no line of intersection (or principal line); thus, the angles of swing and azimuth are undefined.

3.3.2TRANSFORMATION MATRIX

The projective transformation equations can be written in matrix form as follows:

$$\begin{bmatrix} X_j - X_i^c \\ Y_j - Y_i^c \\ Z_j - Z_i^c \end{bmatrix}^T = \lambda_{ij} \begin{bmatrix} x_{ij} - x_p \\ y_{ij} - y_p \\ -f \end{bmatrix}^T \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

Then,

$$[X_j \ Y_j \ Z_j] = [X_i^c \ Y_i^c \ Z_i^c] - \lambda_{ij} \cdot [x_p \ y_p \ f] \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} + \lambda_{ij} [x_{ij} \ y_{ij} \ 0] \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

or

$$\mathbf{X}^T = \mathbf{T}^T + \lambda_{ij} \mathbf{x}^T [R] \ ,$$

Where

$\mathbf{X}$ = a vector of points in the object space

$\mathbf{T}$ = a translation vector for the principal point

$\mathbf{x}$ = a vector for image points in the photo-coordinate system

[R] = a rotation matrix

λ_{ij} = an overall scale factor due to camera altitude

Using the homogeneous coordinate system, this can be written as

$$[X \quad Y \quad Z \quad 1] = \lambda_{ij} [x \quad y \quad -f \quad 1] \begin{bmatrix} R & \vdots & 0 \\ & \vdots & 0 \\ & \vdots & 0 \\ \cdots\cdots & + & \cdots \\ T_x \quad T_y \quad T_z & \vdots & \frac{1}{\lambda_{ij}} \end{bmatrix}$$

3.3.3 ESTIMATING THE TRANSFORMATION MATRIX [TRF]

From the transformation matrix derived above, the equations relating an object space point to an image point are

$$x = \lambda_{ij} \left(m_{11} x + m_{21} y + (T_x - m_{31} f) \right)$$

$$Y = \lambda_{ij} \left(m_{12} x + m_{22} y + (T_y - m_{32} f) \right)$$

$$z = \lambda_{ij} \left(m_{13} x + m_{23} y + (T_z - m_{33} f) \right)$$

The overall scale factor λij is altitude dependent and is generally known. Then, the remaining nine unknowns can be determined by using three control points. If, however, more than three control points are available, then it becomes an overdetermined system and a least squares technique may have to be used. The least square problem is formulated as follows: These equations have a general form

$$W = a + bV_1 + cV_2$$

where

W is a dependent variable

V_1, V_2 are independent variables

a, b, and c are the parameters to be determined

Then the normal equations to be solved for the least squares estimates of a, b, and c are

$$\sum W = n \cdot a + b \sum V_1 + c \sum V_2$$

$$\sum WV_1 = a \sum V_1 + b \sum V_1^2 + c \sum V_1 V_2$$

$$\sum WV_2 = a \sum V_2 + b \sum V_1 V_2 + c \sum V_2^2$$

where all the summations are over the number of observations (n).

Thus, the resulting nine equations when solved provide estimates for the nine elements of the transformation matrix [TRF]. Using this matrix [TRF], a point operation can be performed for each

image point to generate its object space equivalence. However, this way of doing rectification is very time consuming and expensive.

Shortly, an analytical method will be outlined which consists of a conceptually simple model of the exterior orientation of an aerial photograph. The necessity of such a model is evident from the fact that a set of fast and efficient operations are needed to carry out image rectification.

The algorithm that performs a least squares fit to the control points is implemented in the program MOSAIC.

3.4 INPUT IMAGE WINDOW FOR MOSAICKING

Generally, the mosaicking requirements consist of producing the user specified output window by selecting appropriate input image data and rectifying it. The input image data to be selected depends upon the orientation of the aerial photograph with respect to the map.

The transformation matrix [TRF] obtained by performing the least squares fit to the control points, can be used to accomplish the extraction of the input image segment to generate the user specified output window. This is illustrated in Figure 3-4.

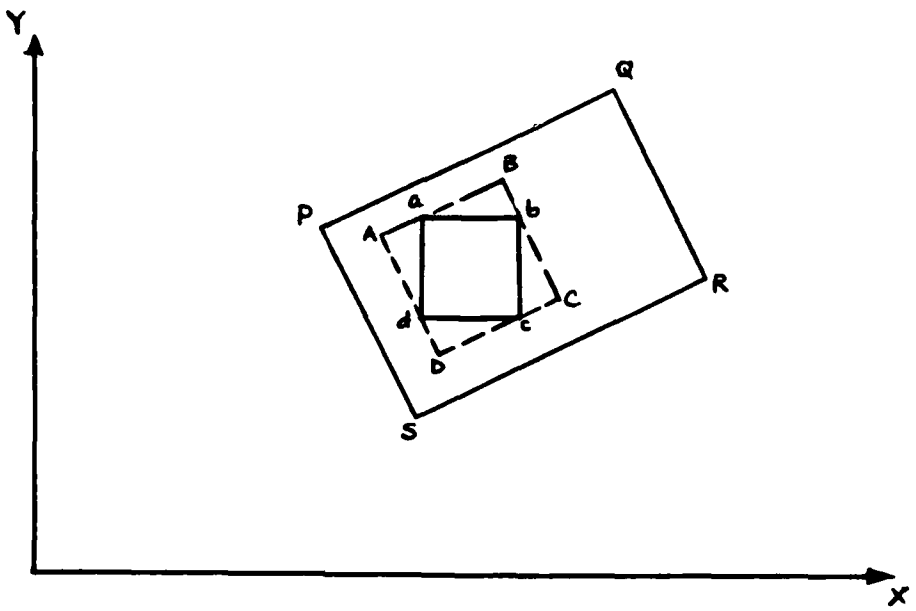


Figure 3-4. Image window for mosaicking

where

X, Y is the rectangular coordinate system for map

PQRS is the aerial photograph

abcd is the user specified output window

ABCD is the input image window to be extracted

Now, the transformation matrix [TRF] maps image points to the corresponding object space points as follows:

$$[X\ Y\ Z] = \lambda_{ij}\ [x\ y\ 1][TRF]$$

Then,

$$[x\ y\ 1] = \frac{1}{\lambda_{ij}} \cdot [X\ Y\ Z][TRF]^{-1}$$

This enables computation of image coordinates corresponding to the user specified output window coordinates.

Now, if the user specified output window "abcd" has the map coordinates (Xa,Ya), (Xb,Yb), (Xc,Yc), and (Xd,Yd), then the corresponding image coordinates given by the above equation are (x_a, y_a), (x_b, y_b), (x_c, y_c), and (x_d, y_d) respectively. However, the input image window "ABCD" to be extracted for mosaicking is given by

$$\left.\begin{aligned} x_A &= \text{MIN}\ (x_a, x_b, x_c, x_d) \\ y_A &= \text{MIN}\ (y_a, y_b, y_c, y_d) \end{aligned}\right\} \quad \text{For upper left corner of window}$$

and

$$\left.\begin{aligned} x_C &= \text{MAX}\ (x_a, x_b, x_c, x_d) \\ y_C &= \text{MAX}\ (y_a, y_b, y_c, y_d) \end{aligned}\right\} \quad \text{For lower right corner of window}$$

This algorithm is implemented in the routine IOWNDO.

3.5 RELATIONSHIP OF AN AERIAL PHOTOGRAPH TO THE MAP

A simple two dimensional model illustrating the relationship of an aerial photograph to the corresponding map is depicted in Figure 3-5.

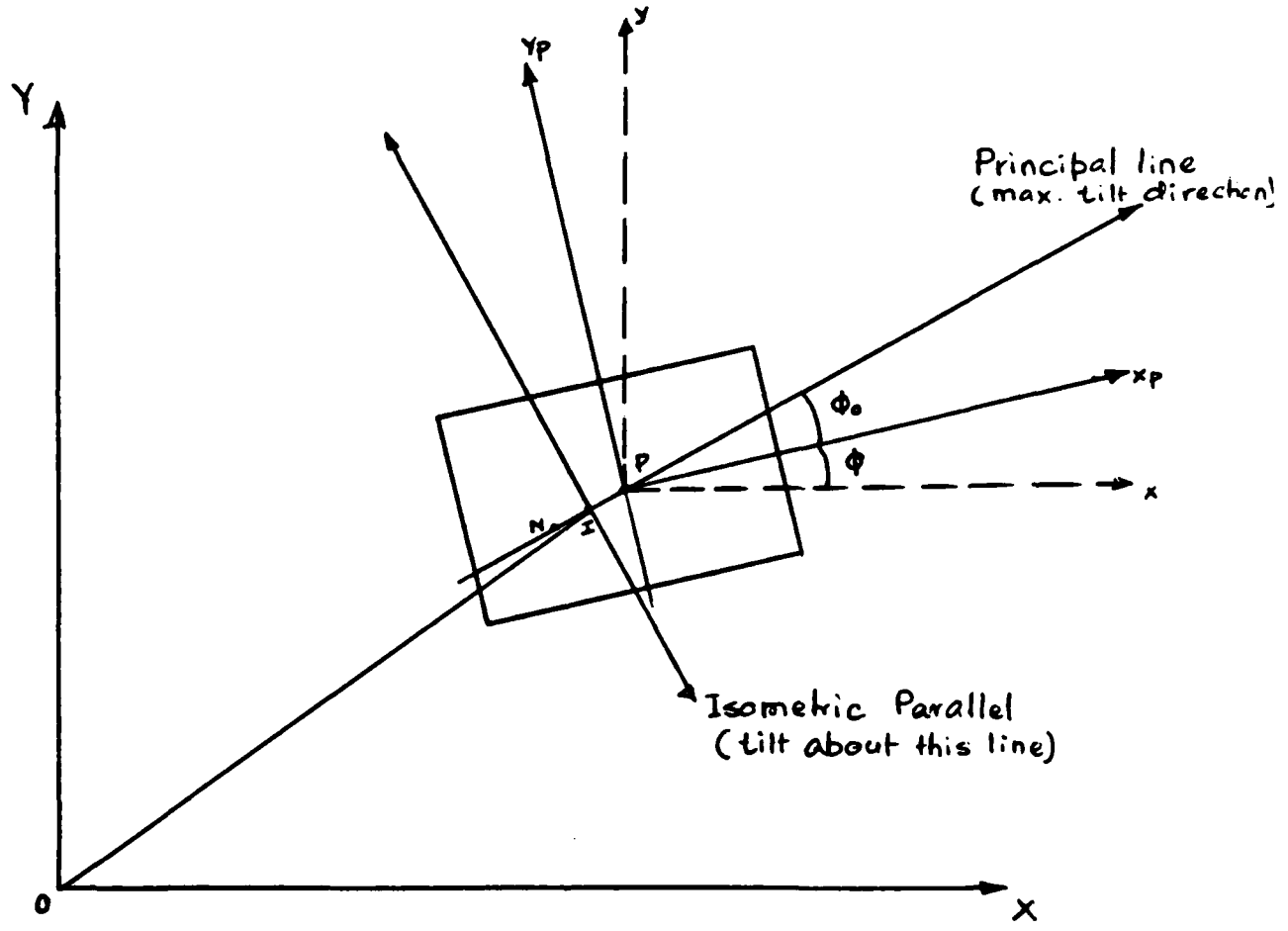


Figure 3-5 Relationship of an aerial photograph to the map

where

X, Y = Map coordinate system

x_p, y_p = Photo-coordinate system

P = Principal point

I = Isocenter

N = Nadir

x, y = Orthographic projection of X, Y on a plane parallel to X–Y plane with origin at P

A = Translation vector with components $[a_x, a_y]$

ϕ = Angle between the photo coordinate system $[x_p, y_p]$ and the rectangular system (x, y)

ϕo = Angle between the pricipal line and x_p.

Then,

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} a_x \\ a_y \end{bmatrix} + s \begin{bmatrix} R^{-1}(\phi) \end{bmatrix} \begin{bmatrix} M \end{bmatrix} \begin{bmatrix} x_p - \Delta_x \\ y_p - \Delta_y \end{bmatrix}$$

where

$R^{-1}(\phi)$ = Rotation of image data from (x, y) system to (x, y) system

M = Scale matrix which removes the tilt effect

s = Overall scale factor due to altitude

$\Delta x, \Delta y$ = Coordinates of the isocenter with respect to the principal point

It should be recognized that $(\Delta x, \Delta y)$ are dependent upon the tilt and swing angles of the photograph which are not known to begin with. Hence, as explained later, they are assumed to be negligible initially and this model enables their determination by using photo-coordinates. Later, an iterative improvement is made

to the predicted values of tilt and swing

Then, writing in terms of x_i, y_i (i.e., image coordinates with respect to isocenter (I) as origin), the model is stated as

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} a_x \\ a_y \end{bmatrix} + s \cdot \begin{bmatrix} R^{-1}(\phi) \end{bmatrix} \begin{bmatrix} M \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

Now

$$\begin{bmatrix} R^{-1}(\phi) \end{bmatrix} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix}$$

$$M = \begin{bmatrix} M_x & 0 \\ 0 & M_y \end{bmatrix}$$

Therefore,

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} a_x \\ a_y \end{bmatrix} + s \cdot \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} M_x & 0 \\ 0 & M_y \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

Since, [M] is a diagonal matrix, $[R^{-1}(\phi)]$ and [M] are commutative; that is

$$[R^{-1}(\phi)][M] = [M][R^{-1}(\phi)]$$

therefore, one can write

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} a_x \\ a_y \end{bmatrix} + s \cdot \begin{bmatrix} M_x & 0 \\ 0 & M_y \end{bmatrix} \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

solving

$$X = a_x + s \cdot M_x \cdot (x_i \cos(\phi) - y_i \sin(\phi)) \qquad (3.1)$$

$$Y = a_y + s \cdot M_y \cdot (x_i \sin(\phi) + y_i \cos(\phi)) \qquad (3.2)$$

3.5.1 SCALE MATRIX [M]

The scale matrix [M] has components in both x and y directions of the photograph and it accounts for image displacement due to tilt. Consider Figure 3-6. which illustrates the effect of tilt on the image acquired.

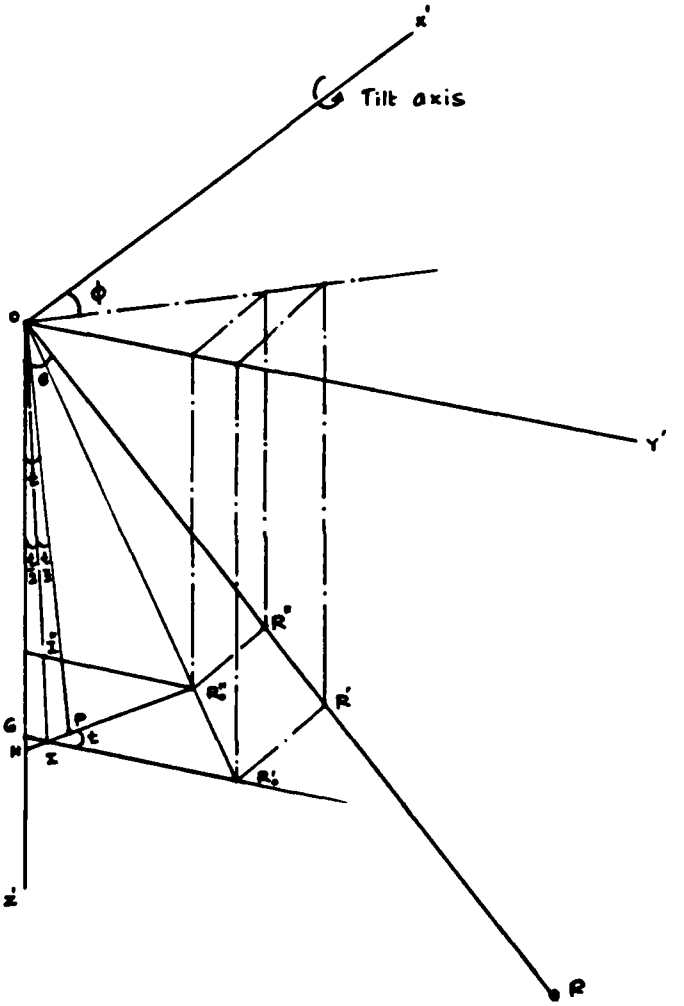


Figure 3-6. Effect of tilt on a photograph

where

X', Y', Z' is a rectangular coordinate system with origin at the center of the camera lens.

R	is a point in the object space with spherical coordinates (r, ϕ, θ)
f	is the focal length of camera
t	is tilt angle and it is around X' axis (i.e., in the $Y'-Z'$ plane)
R'	is the projection of R on the horizontal plane (i.e., parallel to $X'-Y'$ plane) at distance f from the origin. The spherical coordinates of R' are denoted as (r', ϕ, θ).
R''	is the projection of R on the tilted photo plane. Its spherical coordinates are denoted as (r'', ϕ, θ).
P, I, N	are the principal point, isocenter, and nadir respectively.

Then, the vector $\overline{R'}$ can be written down as

$$\overline{R'} = r'\left(\bar{i}\sin(\theta)\cos(\phi) + \bar{j}\sin(\phi)\sin(\theta) + \bar{k}\cos(\theta)\right)$$

and

$$\overline{R''} = r''\left(\bar{i}\sin(\theta)\cos(\phi) + \bar{j}\sin(\phi)\sin(\theta) + \bar{k}\cos(\theta)\right)$$

But

$$\overline{R'}\cdot\bar{k} = f = r'\cos(\phi) \qquad (3.3)$$

Since, the tilt is around X' axis, OP is in the $Y'-Z'$ plane and the principal line is along Y' axis. A unit vector in the direction of OP is

$$\bar{n} = \bar{j}\sin(t) + \bar{k}\cos(t)$$

Also

$$\bar{R}'' \cdot \bar{n} = OP = f = r''\left(\sin(\theta)\cdot\sin(\phi)\cdot\sin(t) + \cos(\theta)\cdot\cos(t)\right) \qquad (3.4)$$

Therefore, equating (3.3) to (3.4)

$$r'\cos(\theta) = r''\left(\sin(\theta)\cdot\sin(\phi)\cdot\sin(t) + \cos(\theta)\cdot\cos(t)\right)$$

$$r' = r''\left(\frac{\sin(\theta)}{\cos(\theta)}\cdot\sin(\phi)\cdot t + \frac{\cos(\theta)}{\cos(\theta)}\cdot\left(1-\frac{t^2}{2}\right)\right)$$

$$\frac{r'-r''}{r''} = \frac{\sin(\theta)}{\cos(\theta)}\cdot\sin(\phi)\cdot t - \frac{t^2}{2}$$

$$= \left(\frac{\sin(\theta)}{f/r'}\cdot\sin(\phi) - \frac{t}{2}\right)\cdot t$$

$$= \left(\frac{r'}{f}\sin(\theta)\cdot\sin(\phi) - \frac{GI}{f}\right)t$$

$$= \left(\frac{GR_o'}{f} - \frac{GI}{f}\right)\cdot t$$

$$= \frac{IR_o'}{f}\cdot t$$

$$\frac{\Delta r}{r''} = \frac{IR_o'}{f}\cdot t$$

$$\frac{\Delta r}{r'} = \frac{r''}{r'}\cdot\frac{IR_o'}{f}\cdot t$$

$$= \frac{I''R_o''}{f}\cdot t$$

But angle $IR_o''I'' = t$ and angle $I''IR_o'' = (\pi - t)/2$

therefore, angle $II''R_o'' = (\pi - t)/2$ and $IR_o'' = I''R_o''$

Hence,

$$\frac{\Delta r}{r'} = \frac{Di\cdot t}{f}$$

which implies

$$\frac{\Delta x}{x'} = \frac{\Delta y}{y'} = \frac{Di\cdot t}{f}$$

Define $(Mx - 1) = \frac{\Delta x}{x'}$ and $(My - 1) = \frac{\Delta y}{y'}$

Then,

$$(Mx - 1) = (My - 1) = Di.\, t/f \quad \text{or}$$

$$Mx = My = (1 + Di.\, t/f)$$

3.5.2 PREDICTION OF PHOTO PARAMETERS

Referring to Figure 3-5, for a point (x_p, y_p), the distance from the isocenter along the principal line is given by

$$Di = x_p.\cos(\phi o) + y_p.\sin(\phi o) + f.\, t/2 \text{ and}$$

$$M_x = M_y = 1 + \frac{(x_p\cos(\phi_o) + y_p\sin(\phi_o) + f\cdot t/2)\, t}{f}$$

$$= 1 + \frac{t^2}{2} + \frac{(x_p\cos(\phi_o) + y_p\sin(\phi_o))\, t}{f}$$

Substituting for Mx and My

$$(X - a_x) = s\cdot\left\{1 + \frac{t^2}{2} + \frac{(x_p\cos(\phi_o) + y_p\sin(\phi_o))\, t}{f}\right\}(x_i\cos(\phi) - y_i\sin(\phi)) \quad (3.5)$$

$$(Y - a_y) = s\cdot\left\{1 + \frac{t^2}{2} + \frac{(x_p\cos(\phi_o) + y_p\sin(\phi_o)\, t}{f}\right\}(x_i\sin(\phi) + y_i\cos(\phi)) \quad (3.6)$$

The unknowns in the above equations are

a_x , a_y , t , ϕo , ϕ as the overall scale factor s and the focal length f are generally known.

3.5.2.1 TRANSLATION VECTOR $\overline{A}$

The two equations in above provide the following relationship

$$\frac{X - a_x}{Y - a_y} = \frac{x_i \cos(\phi) - y_i \sin(\phi)}{x_i \sin(\phi) + y_i \cos(\phi)}$$

$$= \frac{x_i - y_i \tan(\phi)}{x_i \tan(\phi) + y_i}$$

Solving for

$$\tan(\phi) = \frac{(Y - a_y) x_i - (X - a_x) y_i}{(X - a_x) x_i + (Y - a_y) y_i}$$

which involves three unknowns

$$a_x \ , \ a_y \ , \ \phi$$

Using three control points, say CP1, CP2, and CP3, these unknowns can be determined as follows

Denote by (X_1, Y_1), (X_2, Y_2), and (X_3, Y_3) the map coordinates of the three control points and by (x_1, y_1), (x_2, y_2), and (x_3, y_3) the corresponding image points respectively. Then, using two control points CP1 , and CP2

$$\tan(\phi) = \frac{(Y_1 - a_y) x_{1i} - (X_1 - a_x) y_{1i}}{(X_1 - a_x) x_{1i} + (Y_1 - a_y) y_{1i}} = \frac{(Y_2 - a_y) x_{2i} - (X_2 - a_x) y_{2i}}{(X_2 - a_x) x_{2i} + (Y_2 - a_y) y_{2i}}$$

cross-multiplying

$$\left[(Y_1 - a_y) x_{1i} - (X_1 - a_x) y_{1i}\right]\left[(X_2 - a_x) x_{2i} + (Y_2 - a_y) y_{2i}\right] =$$

$$\left[(X_1 - a_x) x_{1i} + (Y_1 - a_y) y_{1i}\right]\left[(Y_2 - a_y) x_{2i} - (X_2 - a_x) y_{2i}\right]$$

Expanding and collecting terms in a_x , a_y together (refer Appendix A for details)

$$a_x^2 + a_y^2 + \gamma_1 a_x + \eta_1 a_y + \delta_1 = 0$$

Similarly, using the two control points CP1 and CP3

$$a_x^2 + a_y^2 + \gamma_2 a_x + \eta_2 a_y + \delta_2 = 0$$

Subtracting

$$(\gamma_2 - \gamma_1) a_x + (\eta_2 - \eta_1) a_y + (\delta_2 - \delta_1) = 0$$

meaning a_x and a_y are linearly dependent Then, solving for a

$$a_y = - \left\{ (\delta_2 - \delta_1) + (\gamma_2 - \gamma_1) a_x \right\} / (\eta_2 - \eta_1)$$

As explained in Appendix A, if the three control points are non-colinear, then

$$\eta_1 \neq \eta_2$$

Substituting for a_y in terms of a_x in the equation

$$a_x^2 + \left\{ \frac{(\delta_2 - \delta_1) + (\gamma_2 - \gamma_1) a_x}{(\eta_2 - \eta_1)} \right\}^2 + \gamma_1 a_x - \eta_1 \left\{ \frac{(\delta_2 - \delta_1) + (\gamma_2 - \gamma_1) a_x}{(\eta_2 - \eta_1)} \right\} + \delta_1 = 0$$

Expanding

$$a_x^2 \left\{ 1 + \left(\frac{\gamma_2 - \gamma_1}{\eta_2 - \eta_1} \right)^2 \right\} + a_x \left\{ \gamma_1 + \frac{2(\delta_2 - \delta_1)(\gamma_2 - \gamma_1)}{(\eta_2 - \eta_1)^2} - \frac{\eta_1 (\gamma_2 - \gamma_1)}{(\eta_2 - \eta_1)} \right\}$$

$$+ \left\{ \delta_1 + \left(\frac{\delta_2 - \delta_1}{\eta_2 - \eta_1} \right)^2 - \eta_1 \left(\frac{\delta_2 - \delta_1}{\eta_2 - \eta_1} \right) \right\} = 0$$

Let

$$\left\{ 1 + \left(\frac{\gamma_2 - \gamma_1}{\eta_2 - \eta_1} \right)^2 \right\} = P$$

$$\left\{ \gamma_1 + 2 \left(\frac{\delta_2 - \delta_1}{\eta_2 - \eta_1} \right) \left(\frac{\gamma_2 - \gamma_1}{\eta_2 - \eta_1} \right) - \eta_1 \left(\frac{\gamma_2 - \gamma_1}{\eta_2 - \eta_1} \right) \right\} = Q$$

$$\left\{ \delta_1 + \left(\frac{\delta_2 - \delta_1}{\eta_2 - \eta_1} \right)^2 - \eta_1 \left(\frac{\delta_2 - \delta_1}{\eta_2 - \eta_1} \right) \right\} = R$$

Then one gets

$$P a_x^2 + Q a_x + R = 0$$

This quadratic in a_x has two roots

$$a_x = \frac{-Q \pm \sqrt{Q^2 - 4PR}}{2P}$$

Since, a_x and a_y are components of a translation vector $\bar{A}$, they should be real numbers. Also, if the origin of the map coordinates system is selected such that the photograph falls in the first quadrant, then both a_x and a_y must be positive and a correct pair can be always be selected.

3.5.2.2 ROTATION ANGLE ϕ

Once a_x and a_y are known, the angle ϕ can easily be determined by evaluating the following equation for control point CP1:

$$\tan(\phi) = \frac{(Y_1 - a_y)\, x_{1i} - (x_1 - a_x)\, y_{1i}}{(X_1 - a_x)\, x_{1i} + (Y_1 - a_x)\, y_{1i}}$$

Then, the rotation angle ϕ is determined such that

$$-\frac{\pi}{2} \leq \phi \leq \frac{\pi}{2}$$

3.5.2.3 DIRECTION OF TILT (ϕ_0)

Equation 3.1 is re-written as

$$X = a_x + s \cdot M_x \cdot \left(x_i \cos(\phi) - y_i \sin(\phi) \right)$$

Substituting for Mx

$$\left[1 + \frac{t^2}{2} + \left\{ \frac{x_p \cos(\phi_0) + y_p \sin(\phi_0)}{f} \right\} t \right] = \frac{X - a_x}{s \cdot (x_i \cos(\phi) - y_i \sin(\phi))}$$

Using control point CP1

$$\left[1 + \frac{t^2}{2} + \left\{ \frac{x_1 \cos(\phi_0) + y_1 \sin(\phi_0)}{f} \right\} t \right] = \frac{X_1 - a_x}{s \cdot (x_{1i} \cos(\phi) - y_{1i} \sin(\phi))} = \beta_1 \quad (3.7)$$

Similary, using control points CP2 and CP3

$$\left[1 + \frac{t^2}{2} + \left\{ \frac{x_2 \cos(\phi_0) + y_2 \sin(\phi_0)}{f} \right\} t \right] = \frac{X_2 - a_x}{s \cdot (x_{2i} \cos(\phi) - y_{2i} \sin(\phi))} = \beta_2 \quad (3.8)$$

and

$$\left[1 + \frac{t^2}{2} + \left\{ \frac{x_3 \cos(\phi_0) + y_3 \sin(\phi_0)}{f} \right\} t \right] = \frac{X_3 - a_x}{s \cdot (x_{3i} \cos(\phi) - y_{3i} \sin(\phi))} = \beta_3 \qquad (3.9)$$

Then, subtracting (3.8) from (3.7)

$$\left\{ \frac{(x_1 - x_2)\cos(\phi_0) + (y_1 - y_2)\sin(\phi_0)}{f} \right\} t = \beta_1 - \beta_2 \qquad (3.10)$$

and subtracting (3.9) from (3.7)

$$\left\{ \frac{(x_1 - x_3)\cos(\phi_0) + (y_1 - y_3)\sin(\phi_0)}{f} \right\} t = \beta_1 - \beta_3 \qquad (3.11)$$

Dividing (3.10) by (3.11)

$$\frac{(x_1 - x_2)\cos(\phi_0) + (y_1 - y_2)\sin(\phi_0)}{(x_1 - x_3)\cos(\phi_0) + (y_1 - y_3)\sin(\phi_0)} = \frac{\beta_1 - \beta_2}{\beta_1 - \beta_3}$$

$$\frac{(x_1 - x_2) + (y_1 - y_2)\tan(\phi_0)}{(x_1 - x_3) + (y_1 - y_3)\tan(\phi_0)} = \frac{\beta_1 - \beta_2}{\beta_1 - \beta_3}$$

Giving

$$\tan(\phi_0) = \frac{\left\{ \frac{\beta_1 - \beta_2}{\beta_1 - \beta_3} \right\}(x_1 - x_3) - (x_1 - x_2)}{(y_1 - y_2) - \left\{ \frac{\beta_1 - \beta_2}{\beta_1 - \beta_3} \right\}(y_1 - y_3)}$$

Solving this equation for ϕ_0 yield

$$-\frac{\pi}{2} \leq \phi_0 \leq \frac{\pi}{2}$$

3.5.2.4 MAGNITUDE OF TILT (t)

Now, equation (3.10) is solved for tilt (t) as follows:

$$t = \frac{f(\beta_1 - \beta_2)}{(x_1 - x_2)\cos(\phi_o) + (y_1 - y_2)\sin(\phi_o)}$$

This provides tilt (t) as a signed number. This signed value of tilt (t) with the signed value of ϕo permits computation of swing angle.

If $t > 0$, then

Swing angle = $3\pi/2 - \phi o$ and,

If $t < 0$, then

Swing angle = $\pi/2 - \phi o$.

This algorithm is implemented in the routine RECTFY.

3.5.3 ITERATIVE IMPROVEMENT OF PHOTO PARAMETERS

The first estimation of photo parameters is made by assuming the separation between the principal point and isocenter as negligible because it depends upon the photo parameters to be predicted. The first prediction of the tilt and swing angles under that assumption allows computation of the isocenter coordinates (Δx, Δy) which can be used for the next iteration. The revised prediction of the photo parameters resulting from such iterations shows a significant improvement. Re-stating the basic model

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} a_x \\ a_y \end{bmatrix} + s \cdot \begin{bmatrix} R^{-1}(\Phi) \end{bmatrix} \begin{bmatrix} M \end{bmatrix} \begin{bmatrix} x_p - \Delta x \\ y_p - \Delta y \end{bmatrix}$$

Using the vector notation it can written as

$$\bar{X} = \bar{A} + s\,[R^{-1}(\phi)][M][\bar{x}_p - \bar{\Delta}]$$

$$\bar{A} = \bar{X} - s\,[R^{-1}(\phi)][M]\,\bar{x}_p + s[R^{-1}(\phi)][M]\,\bar{\Delta}$$

$$= \bar{X} - (\bar{p} - \bar{q})$$

where

$$\bar{p} = s\cdot[R^{-1}(\phi)][M]\,\bar{x}_p$$

$$\bar{q} = s\cdot[R^{-1}(\phi)][M]\,\bar{\Delta}$$

Now, the first predictions are

$$\bar{A}^\circ\ ,\ \phi^\circ\ ,\ \phi_o^\circ\ ,\ t^\circ$$

and they enable computation of $[M^\circ]$, and $\bar{\Delta}^\circ$ such that

$$\bar{p}_o = s\cdot[R^{-1}(\phi^\circ)][M^\circ]\,\bar{x}_p$$

satisfying

$$\bar{A}^\circ = \bar{X} - \bar{p}_o$$

Then, the iterations can be stated as

$$\bar{q}_0 = s \cdot [\bar{R}^{-1}(\phi^0)][M^0]\,\bar{\Delta}^0$$

$$\bar{A}_1 = \bar{X} - (\bar{P}_0 - \bar{q}_0) = \bar{A}^0 + \bar{q}_0$$

Get new values of ϕ^1, ϕ_0^1, t^1, $[M^1]$, $\bar{\Delta}^1$

$$\bar{P}_1 = s \cdot [\bar{R}^{-1}(\phi^1)][M^1]\,\bar{x}_P$$

$$\bar{q}_1 = s \cdot [\bar{R}^{-1}(\phi^1)][M^1]\,\bar{\Delta}^1$$

$$\bar{A}_2 = \bar{X} - (\bar{P}_1 - \bar{q}_1) = \bar{A}^1 - (\bar{P}_1 - \bar{P}_0 - \bar{q}_1)$$

Get new values of ϕ^2, ϕ_0^2, t^2, $[M^2]$, $\bar{\Delta}^2$

and repeat the iteration.

and the convergence to be tested is

$$\left| \bar{\Delta}^n - \bar{\Delta}^{n-1} \right| < 1 \text{ pixel}$$

Thus, once the position of the isocenter is determined within a subpixel accuracy, the iterations are stopped. The resulting values of the photo parameters are used in the subsequent image rectification procedure. During algorithm testing, it has been observed that the convergence is reached within, generally, a couple of iterations.

3.6 IMAGE RECTIFICATION

Assuming that the distortion due to film shrinkage during film processing is negligible, an aerial photograph can be considered essentially rectified when the distortion due to tilt effect is removed.

It is evident that the scale matrix [M] removes the distortion due to tilt. Now,

$$[M] = \begin{bmatrix} M_x & 0 \\ 0 & M_y \end{bmatrix}$$

and $$M_x = M_y = 1 + \frac{t^2}{2} + \left\{\frac{(x_p \cos(\phi_o) + y_p \sin(\phi_o)}{f}\right\} t$$

where

t = tilt

f = camera focal length

ϕ_o = rotation angle between the principal line and the x direction of the photo-coordinate sysytm.

(x_p, y_p) = photo-coordinates of an image point

The quantity $(x_p \cos(\phi_o) + y_p \sin(\phi_o))$ is a distance along the principal line of an image point (x_p, y_p). Since, the photo digitization may occur in some arbitrary direction, it may be inefficient to apply the scale matrix [M] to image data keeping the scan direction unchanged. This is due to the fact that for

each pixel on a scan line the distance along the principal line may be different, implying, that it will have to be computed for each pixel on a line and for each line.

It is the intent of this section to show that there is a preferred scan direction for each photograph so that the scale matrix [M] can be applied to remove the tilt induced distortion as a fast scan line oriented operation. This is illustrated in Figure 3-7.

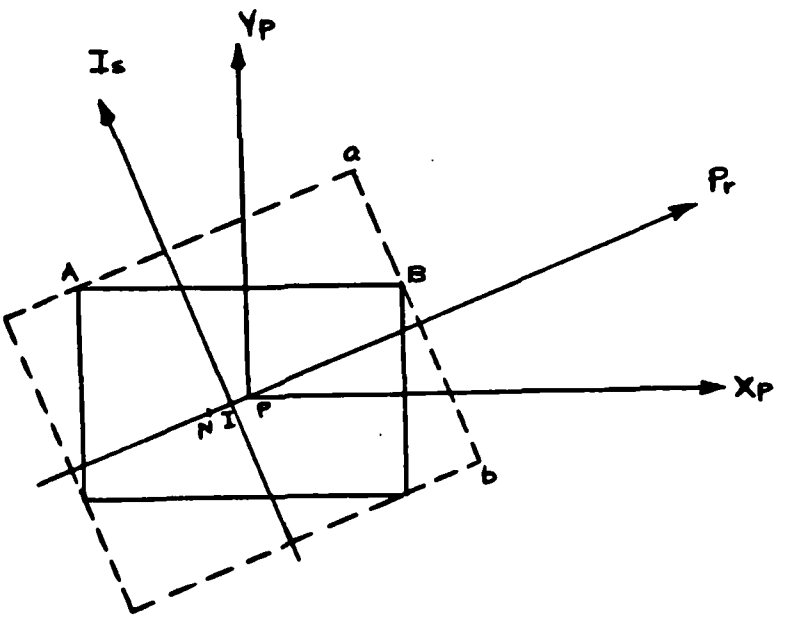


Figure 3-7. Actual and preferred scan directions for an aerial photograph digitization

In the Figure 3-7 the rectangular photo-coordinate system (Xp, Yp) has an origin at P, the principal point, which is also the geometric center of the photograph; the principal line Pr and isometric parallel Is form a rectangular coordinate system with origin at I, the isocenter; and the two coordinate systems are at an angle Φ_o. The photograph is digitized along direction "AB" so that the scan lines are parallel to Xp-axis. The preferred direction of digitization, in this case, is "ab" which is parallel to the isometric parallel line. Then, it is easy to see that for each scan line, all pixels on the line will have the same distance along the pricipal line.

Now

$$1 + \frac{t^2}{2} + \left\{ \frac{x_p \cos(\Phi_o) + y_p \sin(\Phi_o)}{f} \right\} t = 1 + \frac{D_i}{f} t$$

where Di is distance of an image point (x_p, y_p) from the isocenter I along the principal line Pr.

Suppose that the image data is rotated to achieve the preferred scan direction. Then, Figure 3-8 illustrates the rectification process as a line oriented operation.

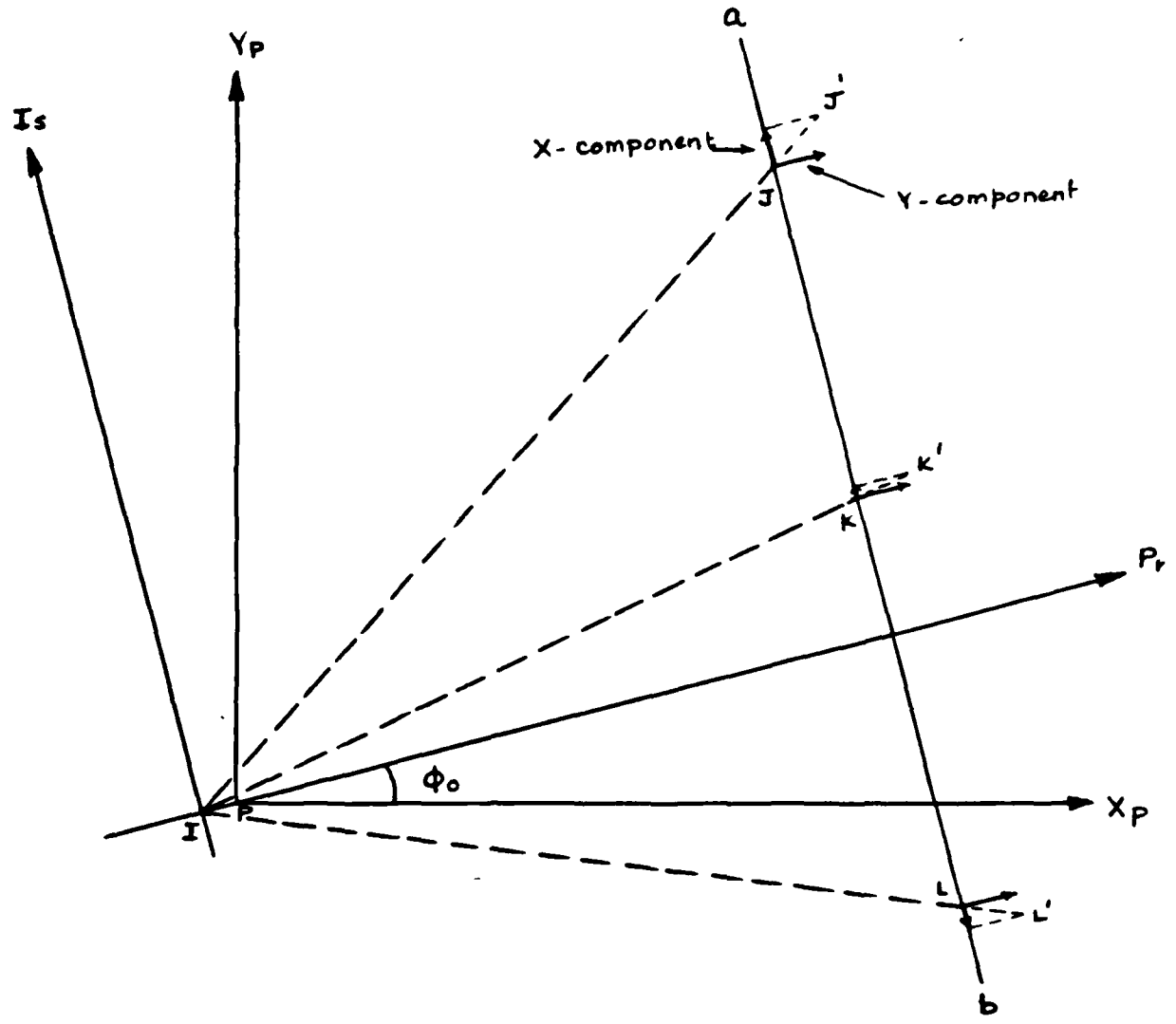


Figure 3-8. Image rectification concept

Let (x_p, y_p) be the coordinates of an image point in Xp-Yp frame with P as the origin; (x_i, y_i) be the coordinates of the same image point when the origin is shifted from P to I.

Clearly,

$$Di = x_i . \cos(\phi o) + y_i . \sin(\phi o)$$

Now, for the image points J, K, and L on the preferred scan direction "ab", their corrected positions are J', K' and L' respectively which are radially outward from the isocenter I. When the vectors JJ' , KK' , and LL' are represented as two orthogonal components, one perpendicuar to Pr and the other parallel to Pr, the latter components turn out to be the same for all points. The component perpendicular to Pr is the x-component in the rotated system and by the same token, the component parallel to Pr is the Y-Component. Also, it is known that

$$X\text{-component} = (M_x)\, x_i = \left(1 + \frac{D_i}{f} t\right) x_i$$

$$Y\text{-component} = (M_y)\, y_i = \left(1 + \frac{D_i}{f} t\right) y_i$$

are coordinates of the corrected positions of image points in the $x_i - y_i$ system. But when the image is rotated to get "ab" as the scan direction, it is in the Is-Pr coordinate system such that each scan line is a constant distance away from the isocenter I, implying the Y-component for all pixels on line "ab" is the same. This means, as a result of rectification, the line "ab" has been shifted from its previous location in the rotated system. The X-components, at the same time, are within scan shifts as illustrated in Figure 3-8. Thus, the rectification is accomplished as a line oriented operation in terms of a constant shift of the input line and a position dependent shift for each pixel of the input line. Also, it should be noted that a position dependent shift for pixels is proportional for all lines, meaning,

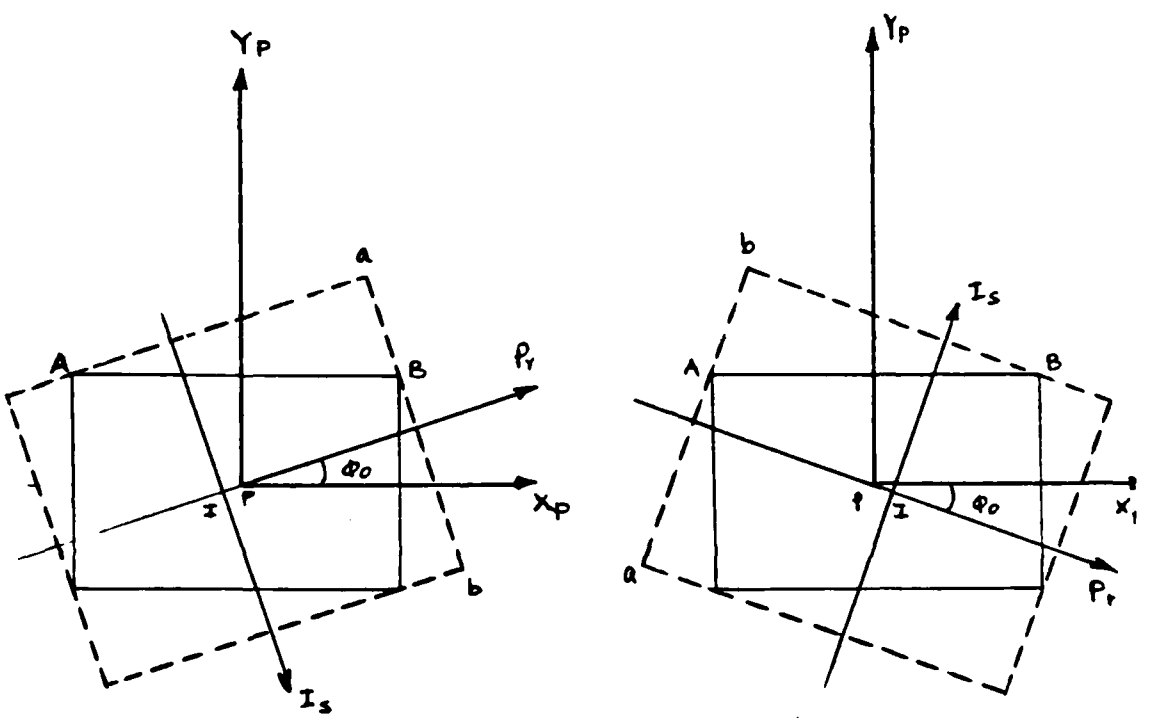
a table can be set up only once and applied to all lines. The advantage of rotating image data to achieve the preferred scan direction is obvious now. It enables implementation of fast line oriented operations to accomplish rectification due to tilt induced distortions. Symbolically, if α and β are image data coordinates in the rotated Is-Pr system, then after rectification, the new image data coordinates will be α_c and β_c such that

$$\alpha_c = (1 + \beta)\alpha$$

$$\beta_c = (1 + \beta)\beta$$

3.6.1 ROTATION FOR THE PREFERRED SCAN DIRECTION

Consider Figure 3-9, which represents the two possible orientations of the principal line Pr.



a) Φ_o is counter-clockwise b) Φ_o is clockwise

Figure 3-9. The preferred scan directions

It is clear that the preferred scan direction "ab" in either case (i.e., when Φ_o is counter-clockwise (positive) or when Φ_o is clockwise (negative)) is at an angle

$$\pm \left(\frac{\pi}{2} - |\Phi_o| \right)$$

where sign is that of the angle ϕ_o.

When image data are rotated to achieve the preferred scan direction, the isocenter (I) becomes the origin of the rotated coordinate system (Is-Pr). The rectification takes place in this rotated coordinate system. The coordinates of (I) in terms of lines and pixels is derived as follows:

If the input image window is m x n (i.e., m pixels and n lines), then the isocenter coordinates with respect to a are given by

$$Ipixel = 0.5(m.\sin(\phi_o) + n.\cos(\phi_o))$$

$$Iline = 0.5(m.\cos(\phi_o) + n.\sin(\phi_o) + f.t)$$

These coordinates permit computation of distance Di for each scan line and pixel in the rotated system.

3.6.2 FACTORING ROTATION INTO TWO SEQUENTIAL SHEARS

A rotation of an image through angle θ implies that the coordinate system is rotated through angle θ. Then, the following relationship exists between the new coordinates (X', Y') and the old coordinates (X, Y):

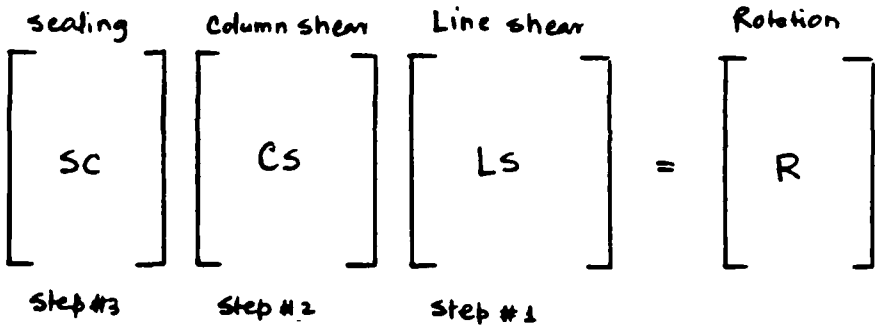
$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

$$= \begin{bmatrix} R \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

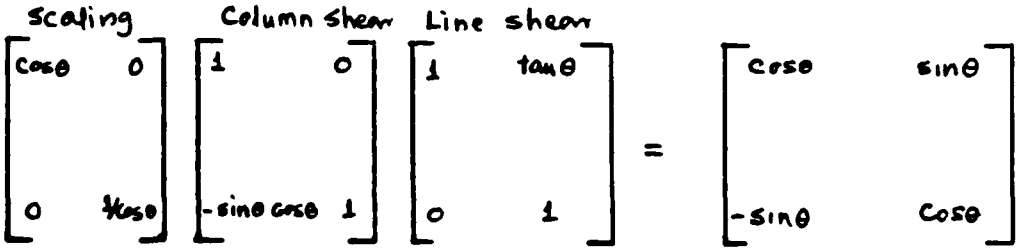
where the matrix [R] indicates the resulting transformation. If this transformation is divided into two sets of operations, one along scan lines and the other along image columns, the rotation can be performed in a very economical way. The "along-line" operations are relatively easy to accomplish. The single required input line can be quickly mapped to the output image through indexing tables computed from the transformation. Efficiencies in the more difficult "along-column" operations are achieved by moving groups of columns as units, for which multiple passes are made and each pass moves groups half as wide as the last. This amounts to performing the rotation by applying two sequential shearing operations, one columnwise and the other rowwise. Sequential shearing directly achieves a satisfactory rotation for small angles. However, for more than a few degrees, the operation

introduces differential scale distortions along lines and columns As with any rotation scheme applied to a rectangular pixel grid, resampling pixels at a compensating rate is required to maintain scale.

The "along-column" operation is termed line shear (LS) and the "along-line" operation is termed column shear (CS). Then the transformation matrix [R] is divided into line shear and column shear as follows:



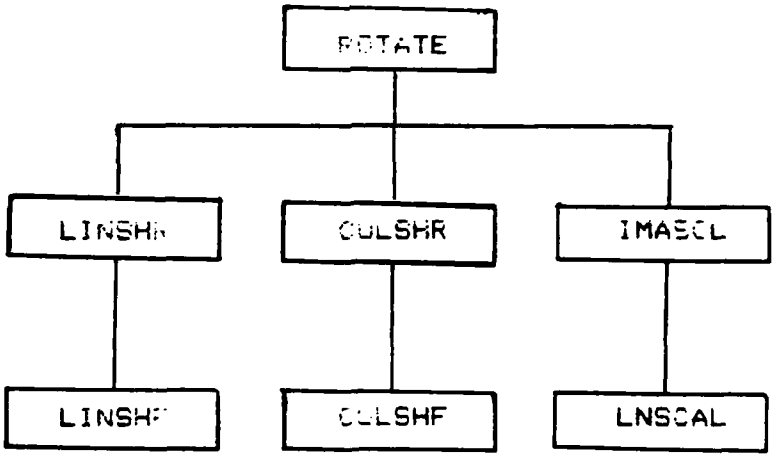
Or



The order of line shear and column shear can be changed to accomplish the same result as evident from the following:

$$\underset{\text{Step \#3}}{\begin{bmatrix} 1/\cos\theta & 0 \\ 0 & \cos\theta \end{bmatrix}} \underset{\text{Step \#2}}{\begin{bmatrix} 1 & \sin\theta/\cos\theta \\ 0 & 1 \end{bmatrix}} \underset{\text{Step \#1}}{\begin{bmatrix} 1 & 0 \\ -\tan\theta & 1 \end{bmatrix}} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

Algorithms to accomplish line shear and column shear are implemented in routines LINSHR and COLSHR respectively. Also, an interactive software system has been implemented to perform rotation using the line shear and column shear technique. The software structure is as follows:



3.6.3 LIMITIMG ROTATION ANGLE USING SEQUENTIAL SHEARS

The distortion introduced by two sequential shears is genarally negligible for small angles of rotation but is quite significant for angles which are more than a few degrees, and scaling must be performed to restore image features and their relationships.

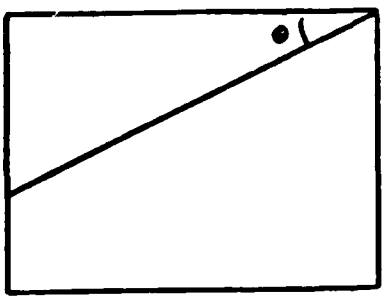
The scaling operation described in the previous section is very nearly equivalent to the nearest neighbor resampling which is considered quite efficient for such operations. Then, it is important to recognize that as long as the sequential shearing operations do not completely destroy the pixel neighborhood, this resampling scheme has some merit.

During the line shear and column shear operations, the pixels are shifted around, and to limit their shifts to one pixel at the most, the rotation angle should be limited to 45 degrees before resampling is performed. An image of 4096 x 4096 pixels requires 12 passes to perform line shear of 45 degrees as illustated in Figure 3-10.

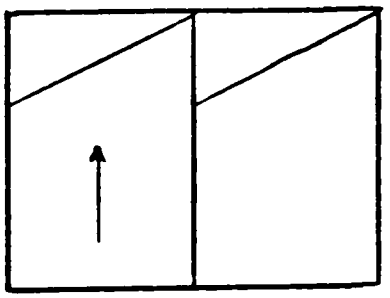
3.6.4 ROTATION ANGLE OF 90 DEGREES

One interesting outcome of having a general purpose line shear and column shear software capability is in being able to rotate images by an angle of 90 degrees fairly easily. This is illustrated in Figure 3-11.

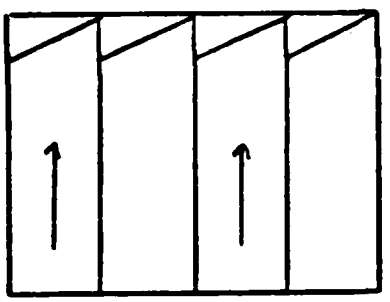
ORIGINAL IMAGE



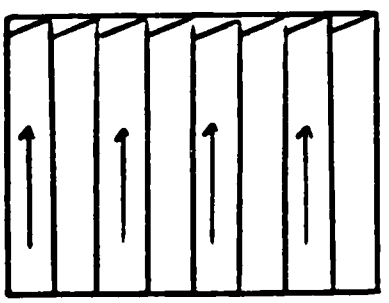
AFTER FIRST PASS HALF OF IMAGE HAS BEEN MOVED UP



AFTER 2nd PASS A DIFFERENT HALF OF IMAGE MOVED UP



AFTER 3rd PASS



```
FOR PASS = 1 to N
   FOR L1 = top to bottom
      SET L2
      READ L1 to buffer 1
      READ L2 to buffer 2
      MOVE half of buffer 2
           into buffer 1
      WRITE buffer 1 to L1
   NEXT L1
NEXT PASS
```

PASS #	MAXIMUM ANGLE FOR 4096 PIXEL LINES
1	
2	
3	
4	
5	.3
6	.6
7	1.1
8	2.2
9	5.5
10	11.
11	22.
12	45.

Figure 3-10 BINARY METHOD OF PERFORMING LINE SHEAR

Now for 90 degrees rotation the transformation matrix [R] would be

$$[R] = \begin{bmatrix} \cos(90) & \sin(90) \\ -\sin(90) & \cos(90) \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

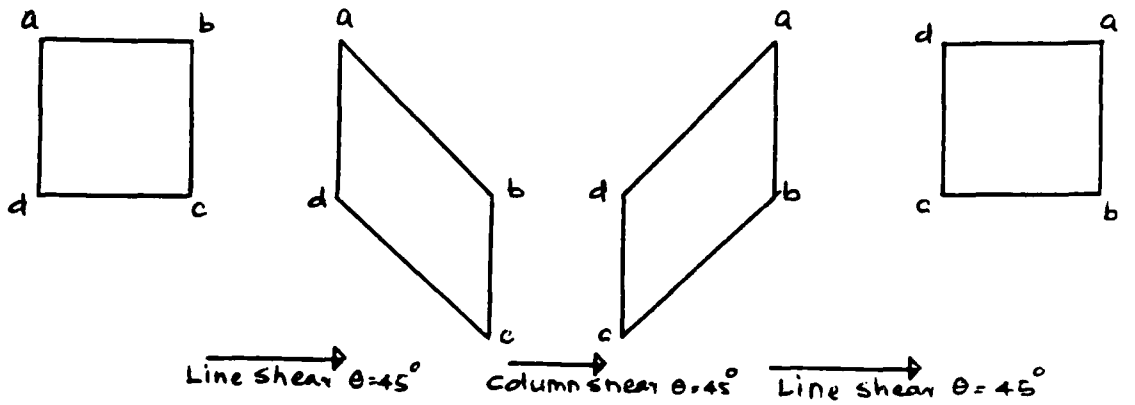


Figure 3-11. 90 Degree Rotation.

$$\begin{bmatrix} 1 & \tan\theta \\ 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 \\ -\tan\theta & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & \tan\theta \\ 0 & 1 \end{bmatrix}$$

Carrying out the matrix product

$$\begin{bmatrix} 1-\tan^2\theta & \tan\theta \\ -\tan\theta & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & \tan\theta \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1-\tan^2\theta & (2-\tan^2\theta)\tan\theta \\ -\tan\theta & -\tan^2\theta+1 \end{bmatrix}$$

Evaluating for θ = 45 degrees

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Thus, the image is rotated by 90 degrees when three sequential shearing operations are performed as follows:

Line shear through 45 degrees

Column shear through 45 degrees

Line shear through 45 degrees

It should be noticed that a scaling operation is not needed in this case and all input data is retained at the end of the operation.

3.6.5 IMAGE ROTATION USING SEQUENTIAL SHEARS:

A pure two dimensional rotation about the origin is performed by the following 2 x 2 transformation matrix:

1) when angle θ is positive,

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

2) when angle θ is negative,

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

It is evident that the maximum rotation required for getting the preferred scan direction in our mosaicking approach is ±90 degrees. To accomplish rotation using sequential shears, the limiting rotation angle requirement of section 3.6.3 should be observed before resampling is performed. The following illustrates the actual implementation of the algorithms to perform image rotation to the preferred scan direction, image rectification, and unrotating the corrected image to original scan direction. The advantage of this approach is that resampling to be performed to restore image features after sequential shears is avoided.

If θ is positive and less than or equal to 45 degrees, then rotation is accomplished by two sequential shears as follows:

$$\begin{bmatrix} \frac{1}{\cos\theta} & 0 \\ 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & \sin\theta\cos\theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\tan\theta & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

scaling step #3 | line shear step # 2 | column shear step # 1 | rotation

Also, rotation in the opposite direction can be implemented as

$$\begin{bmatrix} 1 & 0 \\ \tan\theta & 1 \end{bmatrix} \begin{bmatrix} 1 & -\sin\theta\cos\theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 \\ 0 & \frac{1}{\cos\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

column shear step # 3 | line shear step # 2 | scaling step #1 | rotation

Now, the image rectification process consists of

$[R^{-1}]$ | $[M]$ | $[R]$

UN-rotation to get original image scan direction (operation #3)

Scaling to remove tilt induced distortions (operation #2)

Rotation to get preferred scan direction (operation #1)

$$\begin{bmatrix} \cos\theta_2 & -\sin\theta_2 \\ \sin\theta_2 & \cos\theta_2 \end{bmatrix} \quad \begin{bmatrix} M_x & 0 \\ 0 & M_y \end{bmatrix} \quad \begin{bmatrix} \cos\theta_1 & \sin\theta_1 \\ -\sin\theta_1 & \cos\theta_1 \end{bmatrix}$$

$$\underbrace{[CS][LS][\text{scaling}]}\quad \underbrace{[M]}\quad \underbrace{[\text{scaling}][LS][CS]}$$

where

[CS] is column shear
[LS] is line shear

The intent is to have

$$[\text{ scaling for } R^{-1}\ ]\ [\ M\ ]\ [\text{ scaling for } R\ ]$$

as one operation in image correction software and just perform two sequential shears to rotate and un-rotate image data.

This is equivalent to the following:

$$\begin{bmatrix} \cos\theta_2 & 0 \\ 0 & \frac{1}{\cos\theta_2} \end{bmatrix} \begin{bmatrix} M_x & 0 \\ 0 & M_y \end{bmatrix} \begin{bmatrix} \frac{1}{\cos\theta_1} & 0 \\ 0 & \cos\theta_1 \end{bmatrix}$$

or

$$\begin{bmatrix} M_x \frac{\cos\theta_2}{\cos\theta_1} & 0 \\ 0 & M_y \frac{\cos\theta_1}{\cos\theta_2} \end{bmatrix}$$

and since, $\theta_1 = \theta_2$, finally

$$\begin{bmatrix} M_x & 0 \\ 0 & M_y \end{bmatrix}$$

This is true when θ is negative and greater than or equal to -45 degrees. Thus, resampling resulting from scaling is avoided.

When θ is positive and greater that 45 degrees, then rotation is accomplished by three sequential shears as follows:

$$\begin{bmatrix} \cos\theta+\sin\theta & 0 \\ 0 & \frac{1}{\cos\theta+\sin\theta} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \cos^2\theta-\sin^2\theta & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{\sin\theta}{\cos\theta+\sin\theta} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\tan 45^\circ & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

scaling step #4	column shear step #3	line shear step #2	column shear step #1	rotation

Also, rotation in the opposite direction can be implemented as

$$\begin{bmatrix} 1 & 0 \\ \tan 45^\circ & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{-\sin\theta}{\cos\theta+\sin\theta} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -(\cos^2\theta-\sin^2\theta) & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\cos\theta+\sin\theta} & 0 \\ 0 & \cos\theta+\sin\theta \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

column shear step #4	line shear step #3	column shear step #2	scaling step #1	rotation

Then, the image rectification process consists of

[CS][LS][CS][scaling] [M] [scaling][CS][LS][CS]

with

[scaling $_{-1}$] [M] [scaling]
for R for R

as one operation in image correction software, and just perform three sequential shears to rotate and un-rotate image data.

This is equivalent to

$$\begin{bmatrix} \frac{1}{\cos\theta_2+\sin\theta_2} & 0 \\ 0 & \cos\theta_2+\sin\theta_2 \end{bmatrix} \begin{bmatrix} M_x & 0 \\ 0 & M_y \end{bmatrix} \begin{bmatrix} \cos\theta_1+\sin\theta_1 & 0 \\ 0 & \frac{1}{\cos\theta_1+\sin\theta_1} \end{bmatrix}$$

or

$$\begin{bmatrix} M_x \cdot \dfrac{\cos\theta_1 + \sin\theta_1}{\cos\theta_2 + \sin\theta_2} & 0 \\ 0 & M_y \cdot \dfrac{\cos\theta_2 + \sin\theta_2}{\cos\theta_1 + \sin\theta_1} \end{bmatrix}$$

and since, $\theta_1 = \theta_2$, finally,

$$\begin{bmatrix} M_x & 0 \\ 0 & M_y \end{bmatrix}$$

Thus, resampling resulting from scaling is completely avoided.

4 THEME CONTOURING AND POLYGON VECTOR CONVERSION

4.1 INTRODUCTION

Digital image file structures are inherently spatial in nature, and most operations relating to the extraction of spatial information are done in a raster format file. The use of multiple data planes in image files facilitates not only image categorization, but the interchange and logical manipulation of the resulting thematic maps. Extracted information can be rapidly aggregated by various administrative and physical units or areas defined by map boundaries. There are other advantages to raster format data bases, including the ease of retrieval of spatially registered information, and the ability to perform operations on a pixel basis thus utilizing the finest resolution of the information available.

The obvious disadvantage to raster format data bases is the great amount of redundant storage used to store smoothly structured themes. This makes vector format data bases, where contiguous areas are stored by vertex strings representing outlines, very attractive. Relative storage economy for each format is determined by the ratio between the effective area subtended by a pixel and the average size of uniformly mapped areas. The advantage of vector storage grows exponentially with area size, showing an advantage over raster storage when the average contiguous area is about twenty five pixels. Vector format data bases are generally more economical, and are particularly easy to use when generating chloroplethic maps from stored information. The problem is that the original extraction of information to be mapped is very difficult.

The Standard Interface Format (SIF) provides a common mechanism for the transmittal of graphics and spatial information between dissimilar data bases. Information is passed in vector format on a magnetic tape containing one or more files. Each file is used to describe one drawing or theme.

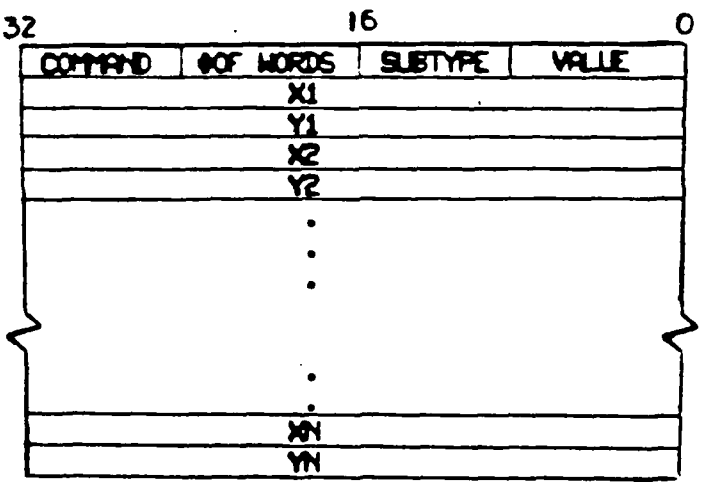
SIF FILE FORMAT

An SIF tape is made up of one or more files. Each file represents one drawing (or, for our purposes, one theme).

Each file contains one or more data blocks, and each data block is 256 32-bit words in length. The first word of each block is always a command header. Data follow the header.

COMMAND AND DATA FORMAT FOR LINE STRINGS

The first word of each command is the command header. It contains four bytes reflecting the type of command, its size, a command modifier, and a one byte value in some cases. The following is the SIF format for line strings:



COMMAND - 40 for a line string
- 80 for continuation of a line string

#OF WORDS - Number of 32-bit words in the command. (255 maximum.)

SUBTYPE - 1 for open line string
- 2 for closed line string representing an area
- 3 for closed line stirng representing a hole

VALUE - 0 for line strings

The mechanics of converting themes to polygons involves an unknown and potentially large number of vertices and presents serious problems concerning computer storage requirements. The number of vertices needed to outline classes in a typical 512 x 512 pixel scene can be grossly estimated at between 5000 and 15000, with the number being highly sensitive to overall scene complexity. In an operational situation where an average map sized area is being processed at high resolution, larger images or image mosaics of 4000x 4000 pixels may be a more representative processing unit size. We clearly need efficient vertex storage, methods of recycling storage during conversion, and a means of dissecting large files into smaller independent segments for processing. Directly storing vertex coordinates in SIF values would use sixty-four bits per vertex, and, if workable, would be very extravagant.

Chain codes are an exceptionally efficient method for storing vertex lists, particularly if the polygons are of complex shapes containing only short straight line segments. A chain code is comprised of a sequence of single digit numbers 0-7, one number per vertex. Based on a rectangular pixel grid, each of the numbers defines one of the eight possible directions pointing to the next vertex. The complete geometric description of any polygon is provided by the coordinates of some starting point and the polygon's chain code. There is some waste associated with storing redundant vertices along extended straight line segments, but this is unimportant in view of the economy gained in storing an average polygon. With the restricted set of direction code possibilities only four bits are required to store each element in the chain. In this respect we might compromise, wasting four bits per vertex by using bytes to decrease execution time and program complexity.

Careful consideration must also be given to the definition of "outline." When dealing with a continuous surface in three space, the concept of a contour is straightforward and can be defined as the intersection of that surface with some

plane within the space. In the discrete world of the pixel grid used with raster format data bases, the concept of a contour is a little more ambiguous. The idea of an intersecting plane is insufficient as the plane no longer intersects a continuum. Actually, when dealing with pixels, a plane is no longer considered continuous, but a series of discrete locations spaced at regular intervals. Any surface intersection in a discrete space must also be discontinuous, as pixel positions represent the limit of available resolution.

Several definitions pertaining to discrete geometry are in order:

DIRECT NEIGHBORS - Pixels pairs that are directly adjacent, sharing a common side in the pixel grid.

INDIRECT NEIGHBORS - Pixel pairs that share a common corner point in the pixel grid.

(N)-NEIGHBOR - The pixel neighbor in direction N from any given position. N ranges from 0 to 7 and increases with counter clockwise angle from east.

DIRECT PATH and INDIRECT PATH - Sequences of pixels containing direct neighbors, and any neighbors, respectively.

INDIRECTLY CONNECTED REGION - A set of pixels, each pair of which can be connected by a path of direct or indirect neighbors. Similarly a **DIRECTLY CONNECTED REGION** is a set, each pair of which is connected by a path of direct neighbors.

CONTOUR or INDIRECT CONTOUR - The set of all pixels in a region which have at least one direct neighbor not in the region,

DIRECT CONTOUR - The set of all pixels in a region which have at least one neighbor not in the region.

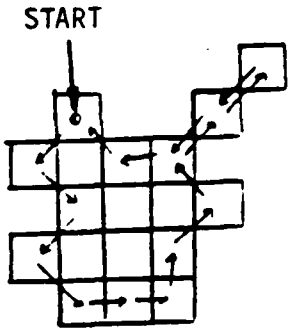
As the list indicates, there are several possible contour definitions usable for theme outlining, including direct and indirect paths outlining direct and indirectly

connected regions. Our choice is dictated by data storage economy. We need to describe themes with the fewest number of vertex strings, and to use the shortest strings possible to provide a complete description. Of the possible methods, the one best suited for outlining noisy or irregular themes is to describe indirectly connected theme areas with indirect contours.

The method for outlining theme holes must also be established. If we are willing to have hole outlines occur within the set defined by a theme region, then the same contour definition can be used for holes. This keeps the software simpler. A single contouring algorithm utilizing an operator which marches along the theme boundary and selects the rightmost available next pixel, can be used for both region types. The types are even classified by the direction of outline traverse. The single algorithm scheme does create some special and potentially troublesome conditions which must be remembered. First, since the hole contours will occur on theme pixels, calculation of hole areas must be different from theme areas to compensate for peripheral difference of one pixel. Also, contours of the two region types have different spatial characteristics. The theme outlines can contain degenerate polygon segments which have no width, and hence no interior area. Examples are single pixel width protrusions, single pixel width bridges between larger areas, and also isolated lines and single pixels. All of these features can be considered degenerate, spanning no interior area in the discrete plane. Hole contours, if they are defined to be in the theme set, are a little easier to work with as they cannot contain degenerate segments. Another commonly occurring special condition arises when a theme area includes a hole that is positioned within one pixel of the region boundary. In these cases the associated contours will be coincident for some fraction of their length, and can complicate processing.

4.2 THEME OUTLINING ALGORITHM

The outlining algorithm is a type of marching operator, and as such must be capable of handling any outline or hole, providing a predictable and repeatable outline for any input theme of any complexity. With the algorithm being used hundreds or even thousands of times on a single complex image, an operator which occasionally gets off track or fails to close outlines in complex situations is unacceptable. Fortunately, such algorithms have been developed and have evolved to a mature state. The one chosen is a contouring algorithm which basically starts at some point on a region contour, determines a starting direction and marches along, finding the rightmost next pixel satisfying the contour definition. The actual pixel testing sequence is interesting and intricate. Directions close to the previous one are the most probable for contour continuation and are tested first. Leftward, but not rightward, acute angle turns are allowed. This results in the generation of an indirect contour around theme areas. Figure 4-1 is a pseudo code algorithm description. Some complicating factors are included to handle discrete plane contours. A loop counter is needed to count the right turns made at any particular operator position in order to identify single, isolated pixels and to avoid continually circling on them. If no new contour pixel is found, the tests used are such that the marching operator will retrace its steps until a new rightward pixel is found. This feature makes it possible to handle single-pixel-width lines or outline segments. Finally, a dual test is used for contour termination. The operator must return to the start point and must be heading in the original starting direction. The reason behind the first test is obvious, and the second is to avoid premature termination when the starting point happens to lie on a section of the polygon where the operator is retracing its steps.



3 2 1
4 P 0
5 6 7

Codes for directions
From position P

The outline of a connected region is completely specified by:

Starting Pixel - (x,y) ,and
Chain Code - 5757082131155 43

The Outline of a connected region R is defined as the set of pixels in R which have at least one direct neighbor (one sharing a common side) not in R.

An outlining operator can be described as something that steps along pixel to pixel on the edge of a region by selecting the rightmost available satisfying the outline definition.

Themes are considered to have a positive area, and are outlined in a counter clockwise direction. Holes are outlined clockwise.

Defining an outline with a chain code

Figure 4-1.

The contour-tracing algorithm is presented in Figure 4-2. However, the completely detailed algorithm is given in the Fortran program listing for the "TRACKR" module which is largely self-documenting. For input, it requires only that a start point be selected which is within the region being contoured while its directly adjacent neighbor to the west is not. Output of the module is a closed contour chain code loaded into a queue. An image I/O subroutine, "UPDATE" is called by TRACKR and does several things. It accepts the just located direction to the next contour pixel and reads the adjacent image line in that direction, using a rotating buffer scheme for efficiency. It returns values for the eight pixels surrounding the next operator position. It also marks the pixel just departed by incrementing its value by one, thus permanently identifying the contour as having been traced. Troublesome single-pixel-width segments are uniquely marked, as their values will be incremented twice. In an effort to minimize the large number of image line reads and writes, a buffering scheme is used which holds the three lines surrounding the operator, reading or writing single lines when the operator moves up or down on the image.

At this point it is appropriate to refer to Figure 4-3 which presents all the contouring and polygon conversion modules and their inter-relationships.

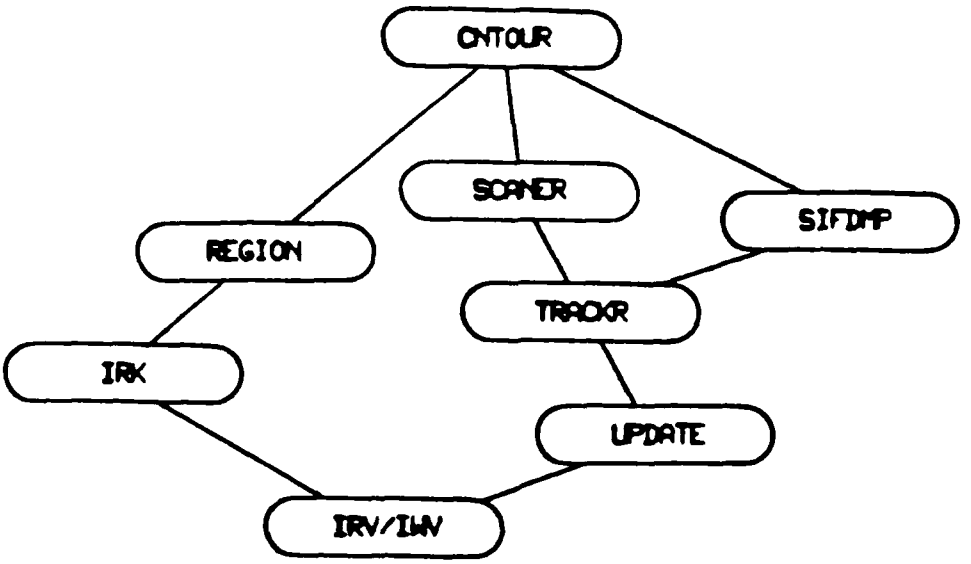
SIFDMP extracts numeric contour coordinates. It is an optional driver for the subroutine TRACKR, with expanded output capabilities, and is used to generate a single theme SIF magnetic tape. SIFDMP is called once for each region in the theme. Given a contour start point in the same manner as TRACKR, this subroutine similary locates the vertices, counts them to insure the SIF line string minimum of three is surpassed, then converts them into strings of thirty-two bit cartesian coordinates in accordance with SIF record structure. Record header information is assigned to prefix the vertex strings and each record is written to tape after it becomes filled. An added feature that was included but not currently used is the capability to linearly transform the polygon vertices into any arbitrary coordinate system.

```
STARTING AT SOME POINT A IN REGION R SUCH THAT ITS 4-NEIGHBOR IS NOT IN R

SET THE CURRENT PIXEL TO THE INITIAL PIXEL
C=A
FIRST = .TRUE.

SET INITIAL SEARCH DIRECTION  S=6

WHILE ((C .NE. A) .AND. (SLOPE .NE. D)) .OR. (FIRST = .TRUE.) DO
   FOUND = .FALSE.
   COUNTER = 1

   WHILE (.NOT. FOUND) .AND. (COUNTER .LE. 3) DO

      IF (B, THE (S-1)-NEIGHBOR OF C IS IN R)
         THEN:
            FOUND = .TRUE.
            C=B
            S=S-2

         ELSE IF (B, THE (S)-NEIGHBOR OF C IS IN R)
            FOUND = .TRUE.
            C=B

         ELSE IF (B, THE (S+1)-NEIGHBOR OF C IS IN R)
            FOUND = .TRUE.
            C=B

         ELSE:  INCREMENT SEARCH DIRECTION BY 90 DEGREES
            S=S+2
            COUNTER = COUNTER +1
      END IF

   END WHILE

   ADD DIRECTION CODE OF LOCATED PIXEL TO CHAIN

   IF (FIRST = .TRUE.)
      THEN:  D = DIRECTION CODE FOR B
      ELSE:  SLOPE = DIRECTION CODE FOR B
   END IF

END WHILE
```

A - Starting point on the contour of region R
C - Current point whose neighborhood is being examined
B - Pixel neighboring C
D - Slope of contour leaving point A
Arithmetic is modulo 8

Contour Tracing Algorithm

Figure 4-2.



CNTOUR - Main program which provides an operator interface and controls the selected processing.

REGION - Defines the scene segment to be processed by inserting a rectangular window into the scene. The window will be searched internally for the location of theme areas.

TRACKR - The contour tracing subroutine. Given some point on the outline of a theme area or hole, this subroutine traces an outline and stores the associated chain code.

SCANER - Region scanning routine which systematically searches the interior of a designated theme area or hole, and returns the location of any internal holes or areas. The interiors of any features located are not scanned.

SIFDMP - A subroutine which generates SIF tape records from a region chain code generated by TRACKR.

UPDATE - Handles image file I/O required by the contour tracing subroutines. After each new pixel position is found by SCANER, this routine marks its position and returns the surrounding pixels.

IRK - G.E. DIAL cursor position reading subroutine.
IRV - G.E. DIAL image line reading subroutine.
IWV - G.E. DIAL image line writing subroutine.

Theme Contouring and Polygon Conversion Program Modules

Figure 4-3.

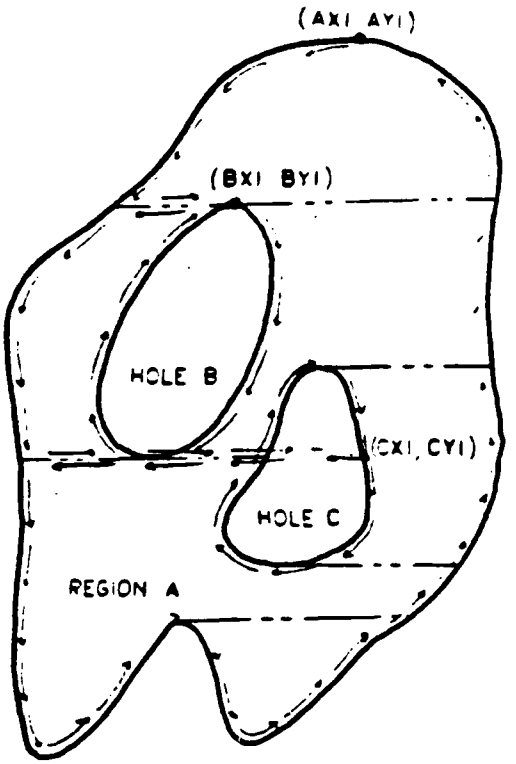
4.3 THEME REGION LOCATING ALGORITHM

The vector conversion software package has a second major module used to locate and tabulate theme areas. It is a region scanner which takes any region in a thematic map and systematically searches the interior for the presence of subregions. The subregions may be either theme areas or holes, depending on the character of the region itself. But in order to keep track of region nesting, it is important that the search area is internal to the region and external to any located subregions. Generations of subregions then can be tracked by the number of times the scanning algorithm must be called to reach a given level of nesting.

Coupled with the previously described outlining algorithm, the region scanner provides a means of locating and extracting all contours associated with a single theme. If desired, a tree structure could be used to keep track of which holes are contained in which areas, etc. Since the outlining algorithm easily regenerates vertex strings completely describing a region from its starting point, the data structure referencing all theme regions need only store a single point for each region.

Figure 4-4 shows the scanning algorithm and sequence used in searching for holes in a theme area. A slightly simplified hole-finding algorithm is presented for clarity. The generalized algorithm is found in the Fortran listing for SCANER. The algorithm requires some point on the outline of some region, draws the complete outline, and then finds all areas of the opposite parity (areas/holes) enclosed in the region. The module returns start points for all subregions in direct contact with the region being scanned.

```
TEMPORY CHAIN        START POINT
CODE QUEUE           LIST

   A1                AX1,AY1  (INPUT)
   A2                BX1,BY1  (OUTPUT)
   A3                CX1,CY1  (OUTPUT)
    .                 .
    .                 .
   AN
   B1
   B2
   B3
    .
    .
   BN
   C1
    .
    .
```




```
GIVEN A POINT ON THE BOUNDARY OF THE REGION TO BE INTERNALLY SCANNED
CALL TRACKR TO MARK THE REGION OUTLINE AND STORE IT IN A QUEUE

WHILE (QUEUE NOT EMPTY) DO
   SET LAST CHAIN CODE TO CURRENT        C0=C
   GET DIRECTION OPPOSITE TO C0          AC0=MOD((C0+4),8)
   GET NEXT CHAIN CODE FROM QUEUE        C=QUEUE(FIRST ELEMENT)

   TEST FOR DESCENDING PATH BEFORE SCANNING LEFT TO RIGHT
   IF ((C.NE.0).AND.(C0.NE.4).AND.(AC0.GE.C))  THEN:
      TERMINATE = .FALSE.

      WHILE (SCAN NOT TERMINATED) DO
         EXTRACT NEXT SEQUENTIAL SET OF THREE PIXELS (ABC) FROM SCAN LINE

         TEST FOR INTERSECTION WITH A PREVIOUSLY UNLOCATED SUBREGION
         IF((A=0).AND.(B=1)) .OR. ((A=0).AND.(B=2).AND.(C=0))
            THEN:
               SAVE NEWFOUND SUBREGION START POINT FOR FUTURE REFERENCE
               CALL TRACKR TO MARK SUBREGION AND APPEND IT TO QUEUE
               TERMINATE =.TRUE.
            ELSE:
               TEST FOR A PREVIOUSLY MARKED OUTLINE
               IF (B.GE.2) TERMINATE =.TRUE.
         END IF
      END WHILE
   END IF
END WHILE
```

—Region Scanning Algorithm To Locate Holes—

Figure 4-4.

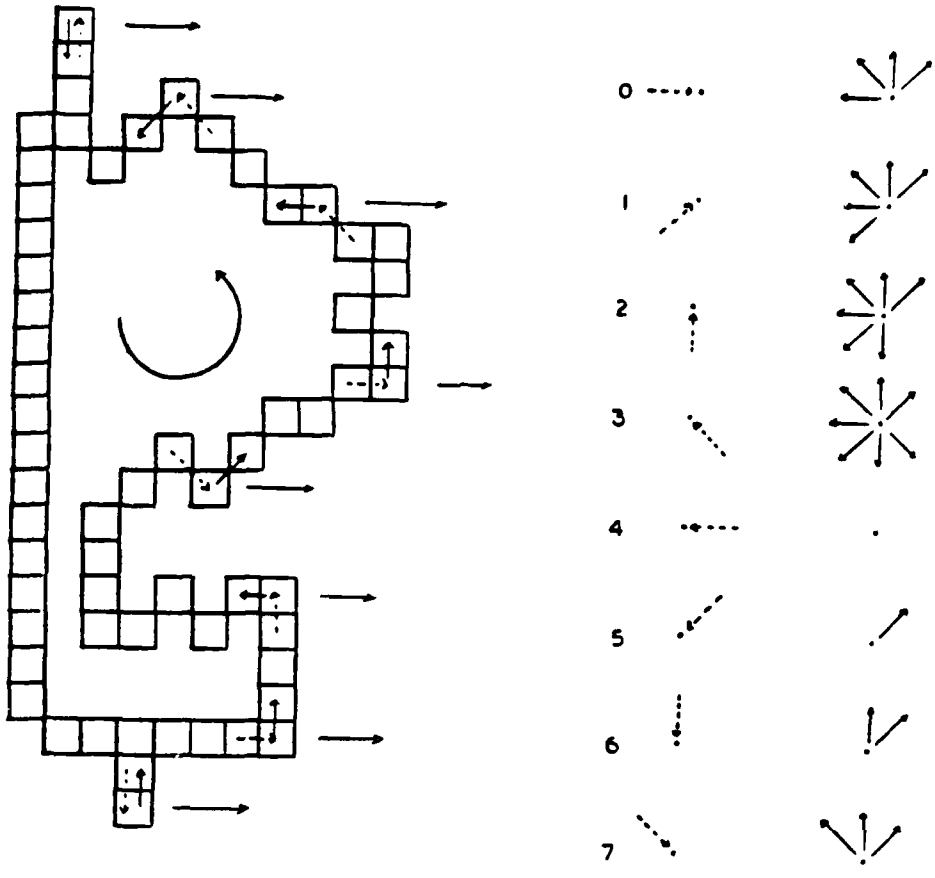
The procedure starts with a binary image in which theme pixels have been assigned a value of **one**, and "not theme" pixels are **zero**. The outline start point is used to call the contour tracing module, which marks each pixel on the outer boundary by adding **one** to the pixel values. Since the outline traced is wholly included in the theme, outline pixels will now have a value of **two** and thus be distinguished from untraced outline pixels. As the outline is traced, the resulting chain code is temporarily placed in a large queue in preparation for scanning the interior of the region.

Points for starting interior scans are found in the queue. Chain code elements are extracted one by one, and sequential pairs are tested to determine if they lie on ascending or decending segments of the region outline. Ascending/descending path pixels are used as start points for rightward scans searching for areas/holes. Along the scan, location of either type subregion is found by the occurrence of two pixels having a value sequence of **zero**, **one**. This sequence occurs where the scan line intersects the leftmost edge of a theme (climbing on), or the rightmost edge of a hole (climbing out). An additional sequence of **zero**, **two**, **zero** locates holes whose right side is coincident with the right side of the region being scanned. Immediately on location of a subregion, the current scan coordinates are stored for future reference.

The contour tracer is called to mark the subregion outline, and to append the outline to the queue. A particular scan is stopped when a new subregion is located, when the outline of previously traced subregion is found, or when the traced right side of the region is reached. Both the latter cases are identified by pixel values of **two or greater**, as they will have been traced and marked at least once.

The reason for using a queue to store region chain codes now becomes apparent. The region, if it is a theme, will be outlined in a counter-clockwise direction. Scanning from descending segments will locate holes to the right (provided they are not eclipsed by other holes) and immediately enter them into the queue. Eventually, when the region start point is returned to, the region outline will have been completely removed from the queue. The scanning algorithm continues, with the upcoming queue elements belonging to the outline of the first hole. Since holes are outlined in a clockwise manner, descending segments occur on their right sides, conveniently serving as start points for scans into sections of the region shaded from the left by the holes. By the time the queue is finally empty, the region boundary along with all hole outlines will have been checked for descending segments. Consequently all sections of the theme region external to the hole type subregions will have been scanned.

The scanning algorithm presented in Figure 4-4 must be expanded somewhat to find areas within holes in order to become a generalized region scanning tool. The decision as to where on the chain to start a line scan becomes more complex. To preserve the left-to-right scanning convention when searching for theme areas, we modify the logic to scan from ascending chain segments (as opposed to the descending segments used in hole location). Figure 4-5 shows the derivation of a comprehensive scan start test, useful for identifying either ascending or decending segments. The test also locates chain code turns at local tops and bottoms in outlines where a simple ascent/descent test is insufficient to identify start points. In all cases the location of a previously traced outline is used to terminate the scan.



Starting scans from ascending segments of theme outlines

CB cases

Corresponding C's for scan start

Notation:
- C - Current chain code pointing to next position
- CB - Last chain code pointing to current position
- ACB - Direction opposite to CB

Rules:
1. Never scan when CB=0 as the line has already been scanned
2. Never scan when C=4 as a better start position is coming up.
3. For rightward scan from ascending chain segments, C must be included in a counter clockwise arc between NE (1) and ACB.
4. For rightward scan from descending chain segments, C must be included in a clockwise arc between SE (7) and ACB.

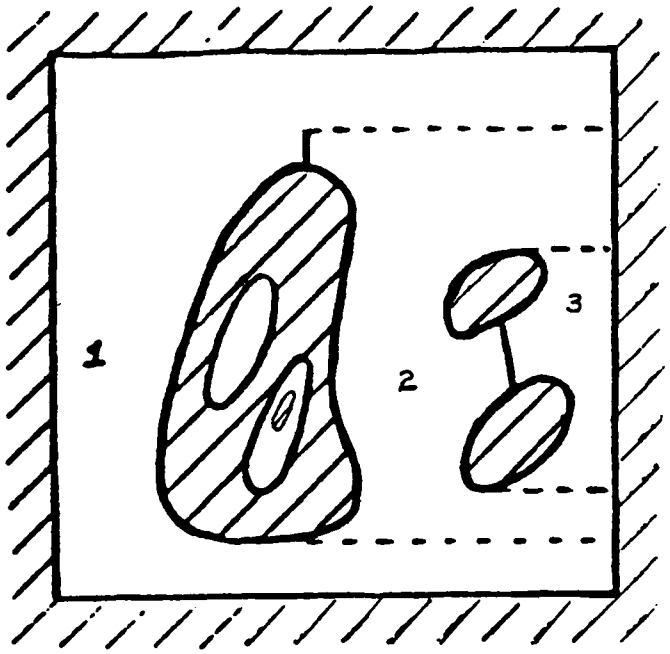
Ascending segment scan start test — (C.NE.0).AND.(CB.NE.4).AND.(ACB.GE.C)
Descending segment scan start test — (C.NE.0).AND.(CB.NE.4).AND.(ACB.LT.C)

Conditions for Starting External Region Scans
Figure 4-5.

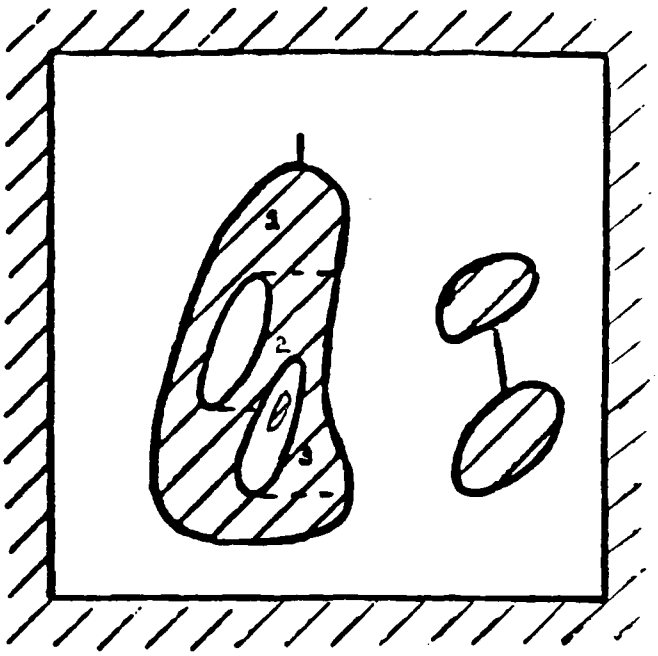
A final complication concerns the handling of degenerate outline segments. When searching for theme are , scanning is internal to a hole type region, hence external to any located heme-type subregions. If the region is to be completely scanned, all ascending egments *including degenerate ones must be used as scan* start points. For noisy themes it is particularly important to scan from singeltons that can otherwise have a combining effect that eclipses large areas. Fortunately the exterior region, being a hole, will not contain degenerate sections and thus allows the entire queue contents to be processed consistently. For the alternate case, the search is internal to a theme area and scanning from degenerate segments must be suppressed or the scan will be outside of the region. Figure 4-6 shows the possibilities.

The scanning algorithm has one known limitation. If SCANER is being used to locate holes (a second level operation for our purposes), and if a hole contains a nested theme subregion, and if all portions of that hole on image lines above the nested theme are scan shaded from the left by other holes, an irregularity occurs. As the lines being scanned descend past the bottom of the shaded zone, the next scan will enter the previously untraced hole, and instead of finding the far side, will prematurely trace around the nested theme. Figure 4-6b illustrates the error producing theme configuration. Tracing direction will still be correct, but without more intricate logic the nested theme will be associated with the wrong external region. For our immediate thematic mapping application this irregularity will probably go unexercised as it can not occur without a rare three-level nesting.

The remaining module "REGION" is used to insert a hole into the stored image to define the processing window. A start point on the window is returned and subsequently passed to SCANER to begin processing of the enclosed theme areas.



a



b

Level 1 region scan from ascending queue segments to locate theme areas.

Scans are internal to hole region and external to located areas.

- Start scans from single pixel width segments.
- Start scans from singletons.

Level 2 scan of decending queue segments to locate holes within theme areas.

Scans are internal to theme area and external to located holes.

- Don't scan from single pixel width segments
- Don't scan from singletons.

Modifications of left-to-right scan start logic for handling degenerate polygons.

Figure 4-6.

APPENDIX A

Derivation of a Quadratic in a_x and a_y:

For two control points CP1 and CP2 we have,

$$\tan \emptyset = \frac{(Y_1 - a_y)\, x_1 - (X_1 - a_x)\, y_1}{(X_1 - a_x)\, x_1 + (Y_1 - a_y)\, y_1} = \frac{(Y_2 - a_y)\, x_2 - (X_2 - a_x)\, y_2}{(X_2 - a_x)\, x_2 + (Y_2 - a_y)\, y_2}$$

Cross-multiplying, we get:

$$\{(Y_1 - a_y)\, x_1 - (X_1 - a_x)\, y_1\}\ \{(X_2 - a_x)\, x_2 + (Y_2 - a_y)\, y_2\} =$$
$$\{(X_1 - a_x)\, x_1 + (Y_1 - a_y)\, y_1\}\ \{(Y_2 - a_y)\, x_2 - (X_2 - a_x)\, y_2\}$$

Expanding,

$$(Y_1 - a_y)(X_2 - a_x)\, x_1 x_2 + (Y_1 - a_y)(Y_2 - a_y)\, x_1 y_2 - (X_1 - a_x)(X_2 - a_x)\, y_1 x_2$$
$$- (X_1 - a_x)(Y_2 - a_y)\, y_1 y_2 = (Y_2 - a_y)(X_1 - a_x)\, x_1 x_2 +$$
$$(Y_2 - a_y)(Y_1 - a_y)\, x_2 y_1 - (X_2 - a_x)(X_1 - a_x)\, y_2 x_1 - (X_2 - a_x)(Y_1 - a_y)\, y_1 y_2$$

Collecting terms in $x_1 x_2$, $x_1 y_2$, $x_2 y_1$, and $y_1 y_2$ together

$$\{(Y_1 - a_y)(X_2 - a_x) - (Y_2 - a_y)(X_1 - a_x)\}\, x_1 x_2 +$$
$$\{(Y_1 - a_y)(Y_2 - a_y) + (X_2 - a_x)(X_1 - a_x)\}\, x_1 y_2 -$$
$$\{(X_1 - a_x)(X_2 - a_x) + (Y_1 - a_y)(Y_2 - a_y)\}\, x_2 y_1 -$$
$$\{(X_1 - a_x)(Y_2 - a_y) - (X_2 - a_x)(Y_1 - a_y)\}\, y_1 y_2 = 0$$

$$\{(Y_1 - a_y)(X_2 - a_x) - (Y_2 - a_y)(X_1 - a_x)\}(x_1 x_2 + y_1 y_2) +$$

$$\{(Y_1 - a_y)(Y_2 - a_y) + (X_1 - a_x)(X_2 - a_x)\}(x_1 y_2 - x_2 y_1) = 0$$

Dividing by $(x_1 y_2 - x_2 y_1)$ throughout, we get

$$\{(Y_1 - a_y)(X_2 - a_x) - (Y_2 - a_y)(X_1 - a_x)\}\left(\frac{x_1 x_2 - y_1 y_2}{x_1 y_2 - x_2 y_1}\right) +$$

$$\{(Y_1 - a_y)(Y_2 - a_y) + (X_1 - a_x)(X_2 - a_x)\} = 0$$

provided $x_1 y_2 \neq x_2 y_1$. If the control points CP1 and CP2 and the origin of the photo-coordinate system are non-colinear, then $\frac{y_1}{x_1} \neq \frac{y_2}{x_2}$ and $x_1 y_2 - x_2 y_1 \neq 0$.

Let us call

$$(x_1 x_2 + y_1 y_2)/(x_1 y_2 - x_2 y_1) = \alpha$$

Then,

$$\{Y_1 X_2 - a_x Y_1 - a_y X_2 + a_x a_y - Y_2 X_1 + a_x Y_2 + a_y X_1 - a_x a_y\}\alpha$$

$$+ \{X_1 X_2 - a_x (X_1 + X_2) + a_x^2 + Y_1 Y_2 - a_y (Y_1 + Y_2) + a_y^2\} = 0.$$

Collecting terms in a_x and a_y,

$$a_x^2 + a_y^2 - a_x \{(Y_1 - Y_2)\alpha + (X_1 + X_2)\} - a_y \{(X_2 - X_1)\alpha + (Y_1 + Y_2)\}$$

$$+ \{X_1 X_2 + Y_1 Y_2 + (Y_1 X_2 - Y_2 X_1)\alpha\} = 0.$$

or

$$a_x^2 + a_y^2 + a_x \left\{(Y_2 - Y_1)\alpha - (X_1 + X_2)\right\} + a_y \left\{(X_1 - X_2)\alpha - (Y_1 + Y_2)\right\}$$
$$+ \left\{X_1 X_2 + Y_1 Y_2 + (Y_1 X_2 - Y_2 X_1)\alpha\right\} = 0.$$

or

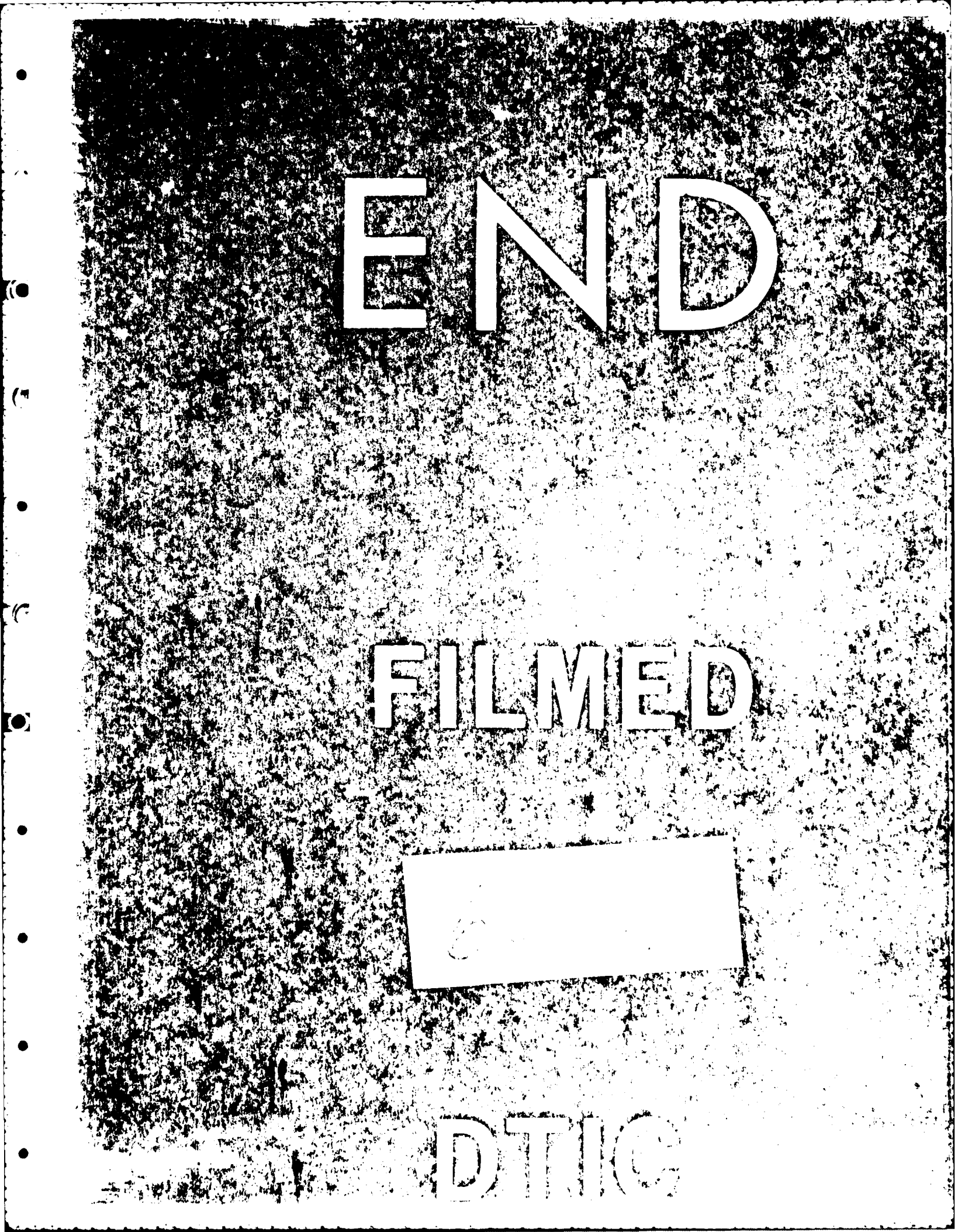
$$a_x^2 + a_y 2 + \gamma a_x + \eta a_y + \delta = 0.$$

where

$$\gamma = (Y_2 - Y_1)\alpha - (X_1 + X_2)$$

$$\eta = (X_1 - X_2)\alpha - (Y_1 + Y_2)$$

$$\delta = X_1 X_2 + Y_1 Y_2 + (Y_1 X_2 - Y_2 X_1)\alpha$$


END
FILMED
DTIC