ALTERNATIVE METHODS OF DETERMINING THE ORIENTATION OF A STELLAR CAMERA

by

Scott M. Rae

Report submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING

in

CIVIL ENGINEERING

APPROVED:

S. D. Johnson, Chairman

James R. Baker

Kathleen Mohr Smith

November, 1987

Blacksburg, Virginia

ALTERNATIVE METHODS OF DETERMINING THE ORIENTATION A STELLAR CAMERA

by

Scott M. Rae

Committee Chairman: S. D. Johnson
Civil Engineering

(ABSTRACT)

Stellar photography has found use in the calibration of mapping cameras and in determining the orientation of the stellar cameras serving as attitude sensors,e.g. the lunar mapping cameras on the NASA Apollo 15, 16, and 17 missions. The equations used to relate the image measurements to the attitude of the camera are the collinearity equations of photogrammetry. These require initial approximations to the unkown attitude angles and the use of an iterative Gaussian correction algorithm. Simpler methods are described which do not require initial approximations to the unknown angles. Two of the methods are modifications of the Church method of space resection and the third is based on the mathematical formulation of a rotation transformation about a single axis. The methods require only two images for a solution. Modifications are given for using the methods with aerial and terrestrial photographs when the camera position is known and two or more control points are imaged on a photograph.

ACKNOWLEDGEMENTS

I would like to thank Dr. Steven D. Johnson, Dr. James R. Baker, and for generously giving their time in reviewing this paper. Their many suggestions, both technical and editorial, were extremely helpful. I would also like to thank the Defense Mapping Agency for allowing me the opportunity to participate in its long term training program.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Background

Stellar photographs have found use in photogrammetry in the calibration of aerial cameras and in determining the orientation of stellar cameras which served as attitude sensors for the lunar mapping camera on the NASA Apollo 15,16, and 17 missions. The reasons for these two uses are that the positions of the stars are accurately known and stars closely approximate point light sources at infinity. This paper addresses the problem of determining the orientation of the camera itself, but some of the findings may have broader implications. From the photogrammetrist's standpoint, the problem of determining the angular orientation of a stellar camera is a special case of space resection or absolute orientation. It is special in the sense that the position of the camera is known. This eliminates three of the six orientation parmeters from the unknowns list. The techniques illustrated for stellar photographs can be applied , with slight modifications, to aerial and terrestrial photographs. For each type of photograph the elements of the problem are the known position coordinates of the taking camera, the known coordinates of two or more control points, and three unknown orientation angles of the camera.

1.2 Problem Definition

The equations which have been used to determine the attitude of a stellar camera are the collinearity equations of photogrammetry. While the collinearity equations are mathematically rigorous and yield accurate solutions to a variety of problems, their solution requires initial approximations to the unknown attitude angles and uses a Gaussian differential correction algorithm which typically requires from three to five iterations for convergence. Even though the initial approximations to the orientation angles need not be very accurate for convergence, accurate approximations do reduce the number of iterations needed. Also, the task of finding approximations is complex enough to require a trained analyst and as with any human task it is a possible source of error. Additionally, the solution of the collinearity equations themselves is no trivial matter. If simpler equations were available, they might be used where speed or ease of computation is more important than extreme accuracy, such as attitude determination of terrestrial cameras. So the primary problem addressed in this report is to find equations or methods which do not require initial approximations to the orientation angles and which simpler in their solution. Two such methods were identified and evaluated.

1.3 Objective

The objective of this paper is to describe and briefly evaluate three alternative methods of determinining the attitude of a stellar camera and show how they can be applied to aerial and terrestrial photographs. The three methods are: 1) the collinearity equation method 2) modifications of the Church method of resection and 3) a single rotation method.

1.4 Scope

This paper presents the equations and terminology for the methods discussed. FORTRAN computer routines for each method are given in the appendices. A sample fictitious data set used to validate the routines is also included along with the validation results. The computer routine used to generate the sample data set is listed in the appendices as well. It computes right ascension and declination for simulated star positions based on input image coordinates and the principal distance and attitude of the taking camera. There is a brief discussion of the merits and limitations of each method. Not covered are the subjects of stellar reduction and the rotation transformation which relates the celestial coordinate system to a local coordinate system.

2. The Collinearity Equation Method

The collinearity equations of photogrammetry were used to determine the attitude of the stellar camera on the NASA Apollo missions. These equations yield a very accurate estimate of the stellar camera orientation, but the equations and their solution are somewhat complex. To give a complete picture of the stellar resection problem so that meaningful comparisons of the various methods may be made, a brief development of the collinearity method is presented here. Using the notation of Junkins (1978) and the Manual of Photogrammetry (1980) the collinearity equations have the form

$$x_i = g_x(\alpha_i, \delta_i, \omega, \phi, k)$$

$$= -f \frac{m_{11}U_i + m_{12}V_i + m_{13}W_i}{m_{31}U_i + m_{32}V_i + m_{33}W_i} \tag{2.1}$$

$$y_i = g_y(\alpha_i, \delta_i, \omega, \phi, k)$$

$$= -f \frac{m_{12}U_i + m_{22}V_i + m_{23}W_i}{m_{31}U_i + m_{32}V_i + m_{33}W_i} \tag{2.2}$$

where

x_i, y_i = photographic coordinates of the i^{th} star image

f = stellar camera principal distance (focal length)

α_i = the right ascension of the i^{th} star

δ_i = the declination of the i^{th} star

The angles ω, ϕ, and k are sequential rotation angles that relate the

camera axes to the axes of the celestial sphere in the following manner. It is assumed that the camera axes and the celestial axes are initially coincident. The camera axes are rotated first through the ω angle, then through the ϕ angle, and finally through the κ angle to bring the axes to their position at the time of exposure. Specific definitions for the ω, ϕ, and κ angles are:

ω = an angle of rotation about the x-axis. Positive ω rotates the positive y-axis toward the positive z-axis

ϕ = an angle of rotation about the y-axis. Positive ϕ rotates the positive z-axis toward the positive x-axis

κ = an angle of rotation about the z-axis. Positive κ rotates the positive x-axis toward the positive y-axis

The direction cosine matrix is

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

$$= \begin{bmatrix} \cos\kappa & \sin\kappa & 0 \\ -\sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & \sin\omega \\ 0 & -\sin\omega & \cos\omega \end{bmatrix} \quad (2.3)$$

so that

$$m_{11} = \cos\kappa \cos\phi$$

$$m_{12} = \sin\omega \sin\phi \cos\kappa + \cos\omega \sin\kappa$$

$$m_{13} = -\cos\omega \sin\phi \cos\kappa + \sin\omega \sin\kappa$$

$$m_{21} = -\sin k \cos \phi$$

$$m_{22} = -\sin\omega \sin \phi \sin k + \cos \omega \cos k$$

$$m_{23} = \cos \omega \sin \phi \sin k + \sin \omega \cos k$$

$$m_{31} = \sin \phi$$

$$m_{32} = -\sin \omega \cos \phi$$

$$m_{33} = \cos \omega \cos \phi \qquad (2.4)$$

U_i, V_i, and W_i = the direction cosines of the i^{th} star in the celestial cordinate system

These are also the components of the line of sight vector and are related to the right ascension and declination by

$$U_i = \cos \alpha_i \cos \delta_i$$

$$V_i = \sin \alpha_i \cos \delta_i$$

$$W_i = \sin \delta_i \qquad (2.5)$$

The relationship of the line of sight vector to right ascension and declination of a star is illustrated in Figure 2.1. The collinearity condition relating a star, its photographic image, and the perspective center is illustrated in Figure 2.2.

To solve for the orientation angles a differential correction algorithm is used. The first step in the algorithm is to linearize the nonlinear equations 2.1 and 2.2 using a Taylor's series approximation that retains only the zero and first order terms so that for the i^{th} star the following equations are written

$$x_i \simeq x_i^{o} + \frac{\partial x_i}{\partial \omega} \Delta\omega + \frac{\partial x_i}{\partial \phi} \Delta\phi + \frac{\partial x_i}{\partial k} \Delta k \qquad (2.6)$$

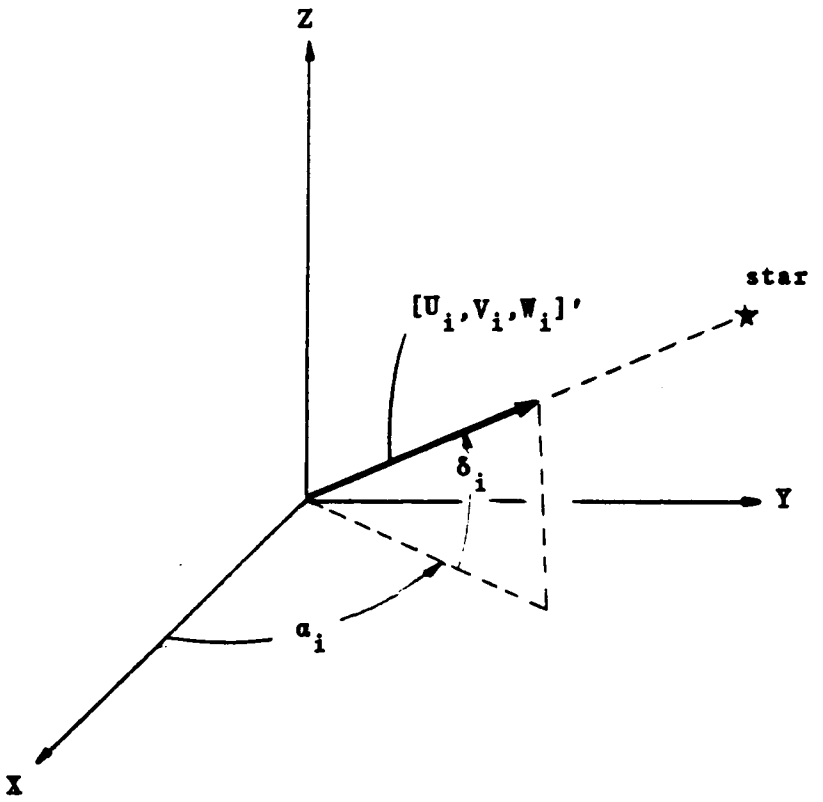

Figure 2.1 Spherical coordinates α_i and δ_i orienting the line of sight vector $[U_i,V_i,W_i]'$ in the celestial coordinate system.

(After Junkins (1978))

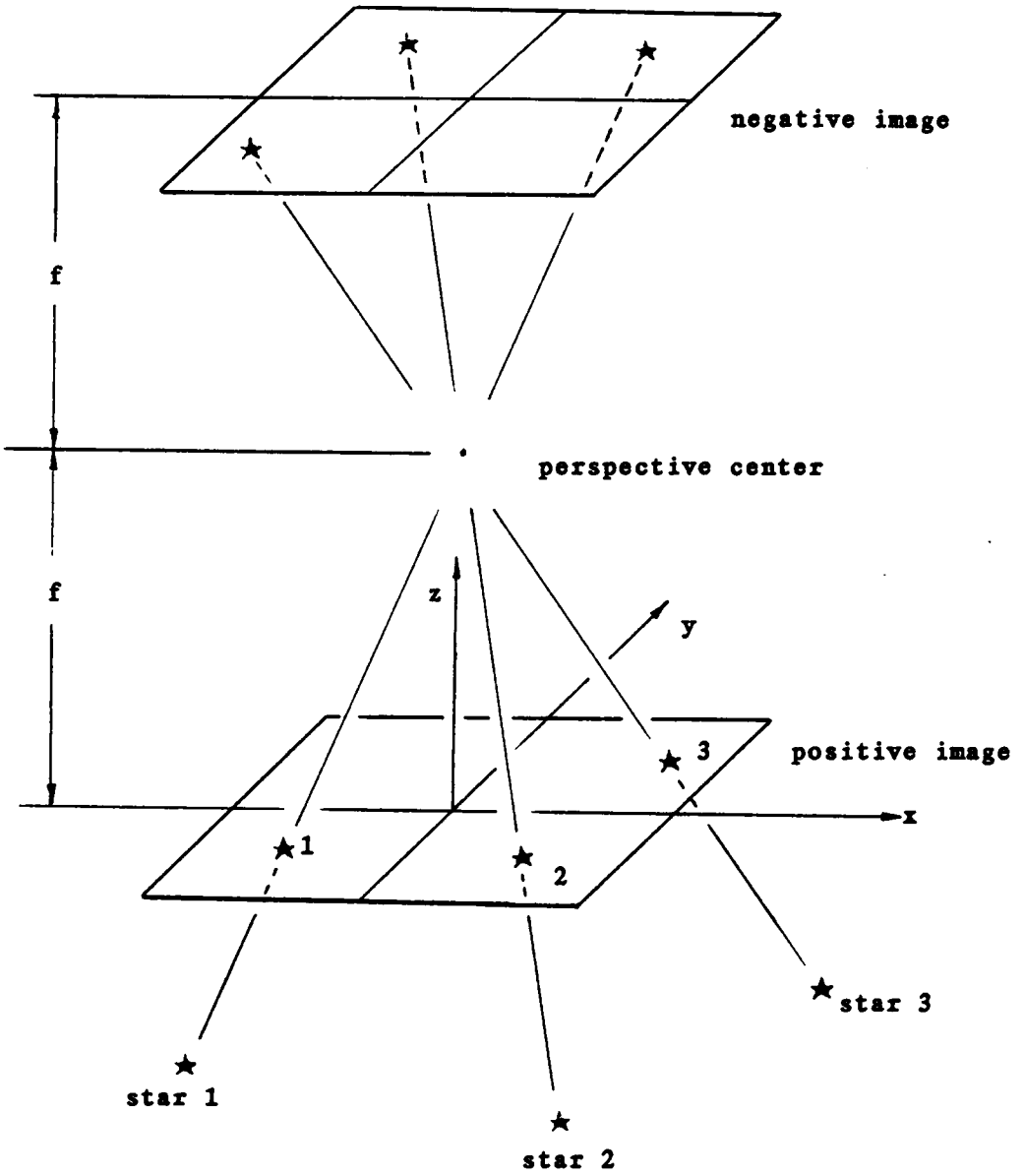

Figure 2.2 Collinearity of the perspective center, image, and object

(After Junkins (1978))

$$y_i \tilde{=} y_i^{o} + \frac{\partial y_i}{\partial \omega}\Delta\omega + \frac{\partial y_i}{\partial \phi}\Delta\phi + \frac{\partial y_i}{\partial k}\Delta k \qquad (2.7)$$

where

$$x_i^{o} = g_x(\alpha_i, \delta_i, \omega^{o}, \phi^{o}, k^{o}) \qquad (2.8)$$

$$y_i^{o} = g_y(\alpha_i, \delta_i, \omega^{o}, \phi^{o}, k^{o}) \qquad (2.9)$$

and the partial derivatives are evaluated at ω^{o}, ϕ^{o}, and k^{o} which are approximations to the orientation angles. Initial approximations are made to the orientation angles and iterative corrections to the approximations are calculated and applied.

The initial approximations can be made once the stellar field has been identified. At that time the photograph can be superimposed on a star chart of approximately the same scale and the relationship between the camera axes and the celestial axes is apparent enough for an analyst to estimate ω, ϕ, and k by visualizing the sequence of rotations necessary to bring the camera axes to there positions at the time of exposure. The approximations can be in error by several degrees and convergence will still occur very rapidly. Junkins (1978) gives an example in which the average error in approximations was seven degrees and convergence occurred within five iterations. On the Apollo missions successive stellar frames had nearly the same orientation so that for most frames the attitude angles from the previous exposure could be

used as the approximations. Corrections to the approximations are calculated using

$$\begin{bmatrix} \omega \\ \phi \\ k \end{bmatrix}^{(j+1)} = \begin{bmatrix} \omega \\ \phi \\ k \end{bmatrix}^{(j)} + ((A^t W A)^{-1} A^t W \, \Delta Y)^{(j)} \tag{2.10}$$

where

$$\Delta Y = \begin{bmatrix} x_1 - g_x(\alpha_1, \delta_1, \omega, \phi, k) \\ y_1 - g_y(\alpha_1, \delta_1, \omega, \phi, k) \\ \vdots \\ x_n - g_x(\alpha_n, \delta_n, \omega, \phi, k) \\ y_n - g_x(\alpha_n, \delta_n, \omega, \phi, k) \end{bmatrix} \tag{2.11}$$

and

$$A = \begin{bmatrix} \frac{\partial x_1}{\partial \omega} & \frac{\partial x_1}{\partial \phi} & \frac{\partial x_1}{\partial k} \\ \frac{\partial y_1}{\partial \omega} & \frac{\partial y_1}{\partial \phi} & \frac{\partial y_1}{\partial k} \\ \vdots & \vdots & \vdots \\ \frac{\partial x_n}{\partial \omega} & \frac{\partial x_n}{\partial \phi} & \frac{\partial x_n}{\partial k} \\ \frac{\partial y_n}{\partial \omega} & \frac{\partial y_n}{\partial \phi} & \frac{\partial y_n}{\partial k} \end{bmatrix} \tag{2.12}$$

$$W = \begin{bmatrix} w_{x1} & & & & & \\ & w_{y1} & & & & \\ & & \cdot & & & \\ & & & \cdot & & \\ & & & & \cdot & \\ & & & & & w_{xn} \\ & & & & & & w_{yn} \end{bmatrix} \tag{2.13}$$

For the uniformly weighted least squares solution the weights or diagonal elements of the W matrix may be set equal to one.

A rather complete development of the collinearity equations including their linearization can be found in of Wolf (1974). The fact that several paritial derivatives must be evaluated and the fact that more than one iteration may be required almost demands that a computer program be used if this method is chosen.

3. Modifications of the Church Method

3.1 Background for the Church Method and Its Modifications

The Church method of space resection was developed in 1945 for aerial photographs and is well suited for manual calculation since the largest system of simultaneous equations involves only three equations. The method is described in detail in the Manual of Photogrammetry (1980). Basically, the Church method is a two-stage process. The first stage consists of an iterative solution for the coordinates of the camera position. However, this first stage is not applicable to stellar photographs since the position of the camera is already known. The second stage of the Church method of resection consists of solving simultaneous linear equations for the nine elements of the orientation matrix. It is this second stage that is applicable to stellar photographs. While it is possible to adapt the second stage almost unchanged for the case of stellar photographs, there are two differences between stellar photographs and aerial photographs that make it worthwhile to investigate the possibility of modifying the second stage so that it is more closely tailored to the special characteristics of stellar photographs.

One difference between aerial photographs and stellar photographs is in the number of control points that appear on a given photograph. For aerial photographs the number of control points used when applying

the Church method is usually only three, three being the minimum number of points required to solve for the six exterior orientation parameters. In the case of stellar photographs though, the control points are stars and there are usually many more than three star images on a single photograph. Because of the larger number of control point images, modifying the method to accommodate more than three star images would be of interest. On the other hand, with stellar photographs there are only three unknown parameters, the three orientation angles, to be determined and the solution should not require three stars. So another modification of the Church method to be considered is to make changes such that only two stars would be required for a solution. With these two objectives in mind the second stage of the Church method will be illustrated for three or more stars and then for the case of only two stars.

3.2 Church Method for Three or More Stars

As is pointed out in the Manual of Photogrammetry (1980), the following relationship holds for one star:

$$\begin{bmatrix} P \\ Q \\ R \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} U \\ V \\ W \end{bmatrix} \tag{3.1}$$

where the following definitions apply:

x_i, y_i = photographic coordinates of the i^{th} star image

U_i, V_i, W_i = direction cosines of the i^{th} star in the celestial coordinate system

$[M_s]$ = orientation matrix of the stellar camera in the celestial coordinate system

f = stellar camera principal distance (focal length)

P_i, Q_i, R_i = direction cosines of the i^{th} star image in the camera coordinate system

These direction cosines are given by the equations:

$$P_i = \frac{x_i}{(x_i^2 + y_i^2 + f^2)^{1/2}}$$

$$Q_i = \frac{y_i}{(x_{i2} + yi2 + f^2)^{1/2}}$$

$$R_i = \frac{-f}{(x_{i2} + yi2 + f^2)^{1/2}} \qquad (3.2)$$

For three stars whose images do not lie on the same straight line (insuring that the image vectors are linearly independent) the relationships given by equation 3.1 can be expressed by a single matrix equation written in the form:

$$\begin{bmatrix} P_1 & P_2 & P_3 \\ Q_1 & Q_2 & Q_3 \\ R_1 & R_2 & R_3 \end{bmatrix} = [M_s] \begin{bmatrix} U_1 & U_2 & U_3 \\ V_1 & V_2 & V_3 \\ W_1 & W_2 & W_3 \end{bmatrix} \qquad (3.3)$$

which can be solved for $[M_s]$ by postmutiplying the leftmost matrix by

the inverse of the rightmost matrix so that

$$[M_s] = \begin{bmatrix} P_1 & P_2 & P_3 \\ Q_1 & Q_2 & Q_3 \\ R_1 & R_2 & R_3 \end{bmatrix} \begin{bmatrix} U_1 & U_2 & U_3 \\ V_1 & V_2 & V_3 \\ W_1 & W_2 & W_3 \end{bmatrix}^{-1} \tag{3.4}$$

Usually, because of small errors made in measuring the coordinates of the star images, the resulting orientation matrix will not be exactly orthogonal. If orthogonality is of great concern, then a rigorous way of achieving it would be to add constraint equations. Six constraint equations would be required since nine linear equations are used in finding the inverse matrix in equation 3.4, but only three unknowns, the three orientation angles, define the model.

Obvious choices for the constraint equations arise from the properties of an orthogonal matrix. The sums of the squares of the elements in any row of $[M_s]$ should total to one and the dot products of any two rows should equal zero . The equations expressing these relationships are nonlinear so that their solution would require linearization and an iterative correction algorithm such as that used in the solution of the collinearity equations. Furthermore, as stated by Mikhail (1976) in a discussion of constraint equations, if linearization is required, then it must be applied to both the condition equations and the constraint equations and the same approximation vector of the parameters must be used both in the condition equations and in the constraint equations. So, while the addition of constraint equations would yield a more rigorous solution, the techinique loses any advantages over the collin-

earity equation method.

A less rigorous means of achieving orthogonality in the orientation matrix is to calculate a set of orientation angles from the elements of the orientation matrix and generate a second matrix from the angles. For the ω, ϕ, and k angles of equation 2.3, the following equations can be used

$$\tan \omega = \frac{-m_{32}}{m_{33}}$$

$$\sin \phi = m_{31}$$

$$\tan k = \frac{-m_{21}}{m_{11}} \qquad (3.5)$$

These equations will yield two values for each of the orientation angles over the interval -180° to $+180^{\circ}$ for a total of eight possible combinations of the three orientation angles. To ascertain which combination is the correct one it is necessary to select the one that gives correct values for the m_{12}, m_{13}, m_{22}, and m_{23} elements of the orientation matrix. Many texts list only equations 3.5 and do not mention selecting the correct combination. Finding the correct combination is not a difficult task, but it does require some computations and comparisons. Even if orthogonality is not a consideration, this step may still be necessary to extract the orientation angles from the matrix if it is the values of the angles that are needed rather than the matrix itself. A FORTRAN subroutine named TRUOPK that finds the orientation angles in their proper quadrant is listed in Appendix C.

Before proceeding further, it shoud be mentioned that this discussion is not intended to imply that the Church method is not rigorous for aerial photography. According to the Manual of Photogrammetry (1980), it is, but the model for stellar photography is different from the aerial photography model and attempting to adapt the Church method to stellar photographs requires changes if rigor is to be maintained.

Another consideration that arises is the proper weighting of the equations. Weighting, if it is needed, can be readily implemented. Noting first that the relationship expressed by equation 3.1 can also be expressed in the form

$$\begin{bmatrix} P \\ Q \\ R \end{bmatrix} = U \begin{bmatrix} m_{11} \\ m_{21} \\ m_{31} \end{bmatrix} + V \begin{bmatrix} m_{12} \\ m_{22} \\ m_{32} \end{bmatrix} + W \begin{bmatrix} m_{13} \\ m_{23} \\ m_{33} \end{bmatrix} \tag{3.6}$$

or for each star three equations can be written

$$[U_i \; V_i \; W_i] \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \end{bmatrix} = [P_i] \tag{3.7}$$

$$\begin{bmatrix} m_{21} \\ m_{22} \\ m_{23} \end{bmatrix} = [Q_i] \tag{3.8}$$

$$[U_i \; V_i \; W_i] \begin{bmatrix} m_{31} \\ m_{32} \\ m_{33} \end{bmatrix} = [R_i] \tag{3.9}$$

For n stars 3n equations could be formed. Using equation 3.7 as an example and adding a weight matrix gives

$$\begin{bmatrix} w_{P1} & & & \\ & w_{P2} & & \\ & & \ddots & \\ & & & w_{Pn} \end{bmatrix} \begin{bmatrix} U_1 & V_1 & W_1 \\ U_2 & V_2 & W_2 \\ \vdots & \vdots & \vdots \\ U_n & V_n & W_n \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{21} \\ m_{31} \end{bmatrix} = \begin{bmatrix} w_{P1} & & & \\ & w_{P2} & & \\ & & \ddots & \\ & & & w_{Pn} \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_n \end{bmatrix} \tag{3.10}$$

Premultiplying by the transpose of the coefficient matrix for a least squares solution of the equations gives

$$\begin{bmatrix} \sum w_{Pi}(U_i)^2 & \sum w_{Pi}(U_iV_i) & \sum w_{Pi}(U_iW_i) \\ \sum w_{Pi}(U_iV_i) & \sum w_{Pi}(V_i)^2 & \sum w_{Pi}(V_iW_i) \\ \sum w_{Pi}(U_iW_i) & \sum w_{Pi}(V_iW_i) & \sum w_{Pi}(W_i)^2 \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{21} \\ m_{31} \end{bmatrix} = \begin{bmatrix} \sum w_{Pi}P_iU_i \\ \sum w_{Pi}P_iV_i \\ \sum w_{Pi}P_iW_i \end{bmatrix} \tag{3.11}$$

The weight w_{Pi} for the i^{th} observation equation is given by

$$w_{Pi} = \left[\sum P_i\right]^{-1} \tag{3.12}$$

where

$$\left[\sum_{P_i}\right] = \begin{bmatrix} \frac{\partial P_i}{\partial x_i} & \frac{\partial P_i}{\partial y_i} \end{bmatrix} \begin{bmatrix} \sigma_{xi}^2 & \sigma_{xiyi} \\ \sigma_{xiyi} & \sigma_{yi}^2 \end{bmatrix} \begin{bmatrix} \frac{\partial P_i}{\partial x_i} \\ \frac{\partial P_i}{\partial y_i} \end{bmatrix} \tag{3.13}$$

Assuming that the measurements of x and y are uncorrelated and have equal variances, then the covariance matrix of the measuring errors is diagonal. Further, if only relative weights are needed, then a suitable weight w_{Pi} is given by

$$w_{Pi} = \left[\frac{\partial P_i^2}{\partial x_i} + \frac{\partial P_i^2}{\partial y_i} \right]^{-1} \tag{3.14}$$

P_i is defined in terms of x_i and y_i by equation 3.2. Performing the partial differentiation yields

$$w_{Pi} = \frac{(x_i^2 + y_i^2 + f^2)^3}{(y_i^2 + f^2)^2 + x_i^2 y_i^2} \tag{3.15}$$

Using a similar approach, expressions for the weights w_{Qi} and w_{Ri} can

be derived

$$w_{Qi} = \frac{(x_i^2 + y_i^2 + f^2)^3}{(x_i^2 + f^2)^2 + x_i^2 y_i^2} \tag{3.16}$$

$$w_{Ri} = \frac{(x_i^2 + y_i^2 + f^2)^3}{f^2 (x_i^2 + y_i^2)} \tag{3.17}$$

Matrix equations similar to 3.8 can then be written for Q and R.

It is interesting to note that the weights w_{Pi} and w_{Qi} are greatest for star images that are farthest from the principal point while w_{Ri} is greatest for star images that are nearest the principal point. In fact, at the principal point w_{Ri} is undefined since division by zero is undefined. Actually, if a star image were coincident with the principal point, then the proper solution of equation 3.6 would be given by

$$\begin{aligned} m_{31} &= -U_i \\ m_{32} &= -V_i \\ m_{33} &= -W_i \end{aligned} \tag{3.18}$$

It is unlikely that a star image would be located exactly at the principal point, but this result does serve to emphasize the fact that in equation 3.9 images near the center of the photograph should be weighted more heavily than those at the periphery. This is in contrast to the collinearity equations method in which images near the periphery are high leverage observations. From this fact alone it might be

expected that the Church method and the collinearity method will yield different solutions with real data or simulated data with errors. Also, it should be pointed out that the m_{31}, m_{32}, and m_{33} elements are the elements from which the ω and k angles are determined so that effects caused by the difference in efffective weighting of the two methods might show up as different values for ω and k. Even more interesting is the fact that replacing x_i and y_i by their polar coordinate equivalents $r_i \cos \theta_i$ and $r_i \sin \theta_i$ in equations 3.9 and 3.17 yields

$$R_i = \frac{-f}{(r_i^2 + f^2)^{1/2}}$$

$$w_{Ri} = \frac{(r_i^2 + f^2)^3}{f^2 r_i^2} \qquad (3.19)$$

Neither θ nor its trigonometric functions appears in either equation. This means that only radial components of image measurement errors will affect the values of ω and k that are determined by the solution. Tangential components of the errors will have no effect. Again, this is in contrast to the collinearity equation method in which either component can affect the solution. So there are at least two reasons to expect different values for ω and k from the two methods.

Before ending a discusion of weighting it would be enlightening to see what range of weights can be expected for the Church method. For the Apollo missions camera which had a principal distance of 76mm and a

24mm x 36mm format

$$\frac{\text{maximum } w_{Pi}}{\text{minimum } w_{Pi}} = \frac{6468}{5776} = 1.12$$

$$\frac{\text{maximum } w_{Qi}}{\text{minimum } w_{Qi}} = \frac{6714}{5776} = 1.16$$

$$\frac{w_{Ri} \text{ at } r_i = 3\text{mm}}{w_{Ri} \text{ at } r_i = 20\text{mm}} = \frac{3724263}{101961} = 36.5 \qquad (3.20)$$

The weights w_{Pi} and w_{Qi} would have little effect, at least on a camera with a narrow cone angle, but W_{Ri} shows a significant range. So, if weighting is used, only one weight need be implemented and instead of inverting one 3x3 matrix for a solution, two 3x3 matrices must be inverted. In all, the Church method is still computationally simpler than the collinearity equation method and while it may have some problems as far as rigor, it also possesses some very interesting attributes.

3.3 Modification of the Church Method for a Two-Star Solution

Equation 3.3 may be used to solve for the orientation by using only two stars rather than three. To do this an artifice is used to create a fictitious third star image and a corresponding fictitious third star. The direction cosine vector of the fictitious star image in

the camera system is formed by taking the cross product of the direction cosine vectors of two real star images

$$[P_1,Q_1,R_1]'x\ [P_2,Q_2,R_2]' = a[P_3,Q_3,R_3]' \tag{3.21}$$

where a is the sine of the angle between the two camera system vectors. The corresponding celestial system direction cosine vector for the fictitious star is formed by taking the cross product of the of the direction cosine vectors of the two real stars

$$[U_1,V_1,W_1]'x\ [U_2,V_2,W_2]' = b[U_3,V_3,W_3]' \tag{3.22}$$

where b is the sine of the angle between the two celestial system vector . $[P_3,Q_3,R_3]'$ and $[U_3,V_3,W_3]'$ are entered into equation 3.3.

The advantage of being able to able to perform a noniterative solution with only two stars may not seem significant at first. However, if all possible solutions using small subsets of the stellar field are considered, it is seen that the number of combinations for 20 stars taken two at a time is 190 while the number of combinations for 20 stars taken three at a time is 1140. Also, there is a timesaving factor involved where maximum accuracy is not required since it is probably faster to create the fictitious third star than to identify a real third star. Finally, this technique can be adapted to aerial and terrestrial photography where control points may be very few.

3.4 Potential Uses of the Church Modifications

In summary, for most applications the modifications to the Church method would not yield solutions as accurate as those derived using the collinearity equations. However, the modifications do offer a simple and rapid means of computing very accurate initial approximations to the orientation angles for use with the collinearity equations.

4. A Single Rotation Method

Single rotation solutions to 3-dimensional rotation transformation problems are seldom, if ever, used in photogrammetry, the preferred approach being to define a rotation transformation in terms of three angles and three axes. In the physical world the corresponding transformation can be accomplished by a single rotation about a single well chosen axis. As it turns out, the analytical expression of the physical rotation can be employed to determine the attitude of a stellar camera in a non-iterative fashion using only two stars. While the solution is neither brief nor elegant, it does demonstrate how the relationship between the single rotation and the three corresponding orientation angles can be used to solve a practical problem.

To begin, the camera system direction cosines of the star images are computed and the celestial system direction cosines of the stars are computed. It is assumed that the camera axes and the celestial axes are coincident initially. So, in effect, the direction cosine vectors of the stars and their corresponding images represent points on a sphere of unit radius. The object is to find an axis that passes through the center of the sphere and an angle of rotation about the axis that will rotate the points that represent the stars on the unit sphere into the points that represent their images. This will define a rotation transformation which, when performed on the camera axes, will represent the attitude of the camera axes at the time of exposure.

The first task is to solve for the axis of rotation. The axis is defined by the cross product of two vectors as is illustrated in Figure 4.1. The first vector is one whose initial point is the the celestial direction cosine vector of the first star and whose terminus is the vector of camera direction cosines for that star.

$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} = \begin{bmatrix} P_1 - U_1 \\ Q_1 - V_1 \\ R_1 - W_1 \end{bmatrix} \tag{4.1}$$

A second vector is defined similarly for the second star.

$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = \begin{bmatrix} P_2 - U_2 \\ Q_2 - V_2 \\ R_2 - W_2 \end{bmatrix} \tag{4.2}$$

The cross product is then taken to find the rotation axis

$$\begin{bmatrix} X_T \\ Y_T \\ Z_T \end{bmatrix} = \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} \times \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} \tag{4.3}$$

For computational convenience a unit vector $[X_R, Y_R, Z_R]'$ proportional to $[X_T, Y_T, Z_T]'$ is formed

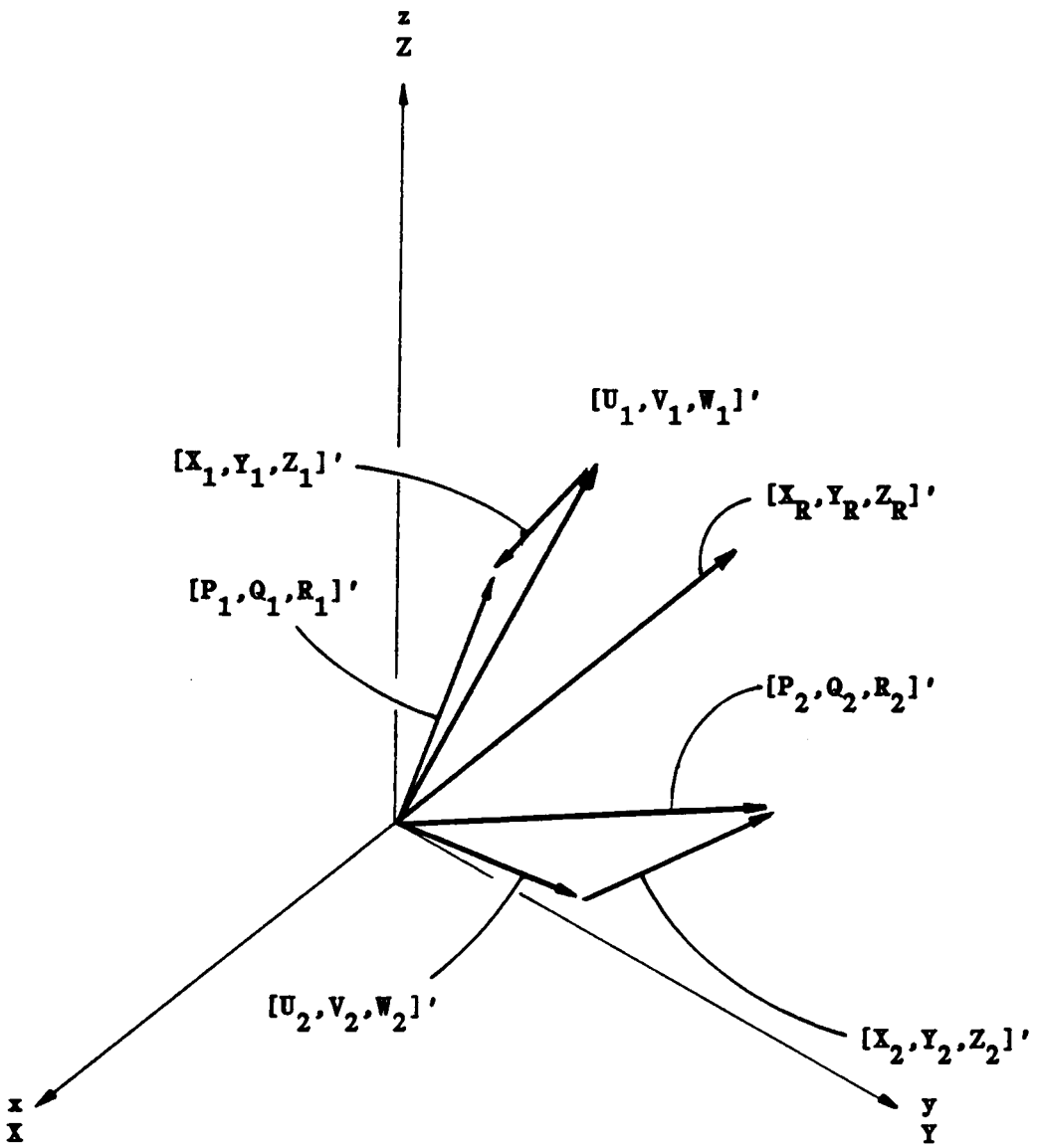

Figure 4.1 The cross product of the difference vectors $[X_1,Y_1,Z_1]'$ and $[X_2,Y_2,Z_2]'$ defines the rotation axis $[X_R,Y_R,Z_R]'$.

$$\begin{bmatrix} X_R \\ Y_R \\ Z_R \end{bmatrix} = a \begin{bmatrix} X_T \\ Y_T \\ Z_R \end{bmatrix} \tag{4.4}$$

where

$$a = (X_T^2 + Y_T^2 + Z_T^2)^{-1/2} \tag{4.5}$$

The next step is to solve for the rotation angle ρ. The projection of the celestial direction cosines vector $[U_1,V_1,W_1]'$ onto the rotation axis (illustated in Figure 4.2) is the cosine of the angle between the vector and the axis mutliplied times a unit vector in the direction of the rotation axis or, in equation form

$$\begin{bmatrix} C_X \\ C_Y \\ C_Z \end{bmatrix} = d \begin{bmatrix} X_R \\ Y_R \\ Z_R \end{bmatrix} \tag{4.6}$$

where

$$d = U_1X_R + V_1Y_R + W_1Z_R \tag{4.7}$$

The projection of the camera direction cosines vector $[P_1,Q_1,R_1]'$ onto the rotation axis $[X_R,Y_R,Z_R]'$ is also $[C_X,C_Y,C_Z]'$ because $[U_1,V_1,W_1]'$ is being rotated into $[P_1,Q_1,R_1]'$ about the rotation axis.

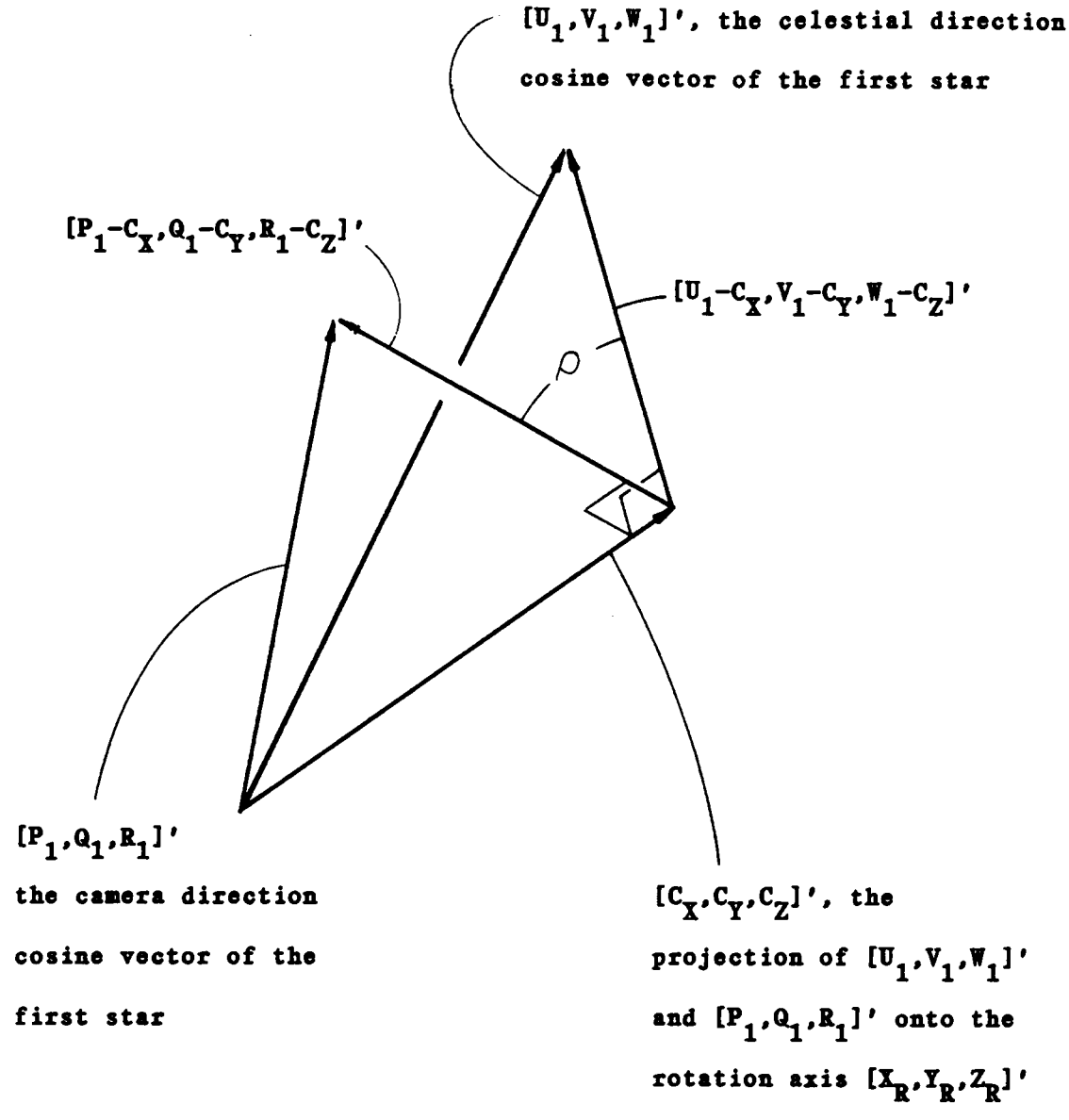

Figure 4.2 The angle of rotation ρ is the angle between the vectors $[U_1-C_X,V_1-C_Y,W_1-C_Z]'$ and $[P_1-C_X,Q_1-C_Y,R_1-C_Z]'$.

The magnitude of the rotation angle ρ is found from the equation

$$\cos \rho = \frac{[U_1-C_X,V_1-C_Y,W_1-C_Z]' \cdot [P_1-C_X,Q_1-C_Y,R_1-C_Z]'}{|[U_1-C_X,V_1-C_Y,W_1-C_Z]'| \; |[P_1-C_X,Q_1-C_Y,R_1-C_Z]'|} \qquad (4.8)$$

This gives the cosine of an angle between 0^o and 180^o. To ascertain whether this is a positive or negative angle, the cross product $[U_1-C_X,V_1-C_Y,W_1-C_Z]' \times [P_1-C_X,Q_1-C_Y,R_1-C_Z]'$ is formed. If it has the opposite direction of $[C_X,C_Y,C_Z]'$, then the rotation angle is negative.

Now that the rotation axis and the rotation angle have been found, the next step is to determine the point $[A_2,B_2,C_2]'$ which represents an arbitrary point $[A_1,B_1,C_1]'$after it has been transformed by the rotation. First, $[A_1,B_1,C_1]'$ is projected onto the rotation axis to form the vector $[A_X,B_Y,C_Z]'$. To calculate the components A_X,B_Y, and C_Z it is first necessary to find the cosine of θ, the angle between the position vector $[A_1,B_1,C_1]'$ and the rotation axis $[X_R,Y_R,Z_R]'$ from the equation

$$\cos \theta = \frac{[A_1,B_1,C_1]' \cdot [X_R,Y_R,Z_R]'}{|[A_1,B_1,C_1]'| \; |[X_R,Y_R,Z_R]'|} \qquad (4.9)$$

and the components of the projection are given by

$$A_X = (X_R)\ (\cos\theta)\ (A_1^2 + B_1^2 + C_1^2)^{-1/2}$$

$$B_Y = (Y_R)\ (\cos\theta)\ (A_1^2 + B_1^2 + C_1^2)^{-1/2}$$

$$C_Z = (Z_R)\ (\cos\theta)\ (A_1^2 + B_1^2 + C_1^2)^{-1/2} \qquad (4.10)$$

Next the two vectors from $[A_X,B_Y,C_Z]'$ to the given point $[A_1,B_1,C_1]'$ and to the unknown point $[A_2,B_2,C_2]'$ are formed. They are defined as follows:

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} A_1 \\ B_1 \\ C_1 \end{bmatrix} - \begin{bmatrix} A_X \\ B_Y \\ C_Z \end{bmatrix} \qquad (4.11)$$

and

$$\begin{bmatrix} D \\ E \\ F \end{bmatrix} = \begin{bmatrix} A_2 \\ B_2 \\ C_2 \end{bmatrix} - \begin{bmatrix} A_X \\ B_Y \\ C_Z \end{bmatrix} \qquad (4.12)$$

The magnitudes of these two vectors are equal so that $A^2 + B^2 + C^2$ equals magnitude of $[A,B,C]'$ times the magnitude of $[D,E,F]'$. So

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} \cdot \begin{bmatrix} D \\ E \\ F \end{bmatrix} = (A^2 + B^2 + C^2)\ \cos\rho \qquad (4.13)$$

or

$$AD + BE + CF = (A^2 + B^2 + C^2) \cos \rho \tag{4.14}$$

Also

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} \times \begin{bmatrix} D \\ E \\ F \end{bmatrix} = (A^2 + B^2 + C^2) \sin \rho \begin{bmatrix} X_R \\ Y_R \\ Z_R \end{bmatrix} \tag{4.15}$$

from which the following equations are formed:

$$BF - CE = X_R (A^2 + B^2 + C^2) \sin \rho \tag{4.16}$$

$$CD - AF = Y_R (A^2 + B^2 + C^2) \sin \rho \tag{4.17}$$

Solving 4.16 and 4.17 for D and E and substituting in 4.14 gives

$$D = \frac{AF + (A^2 + B^2 + C^2)Y_R \sin \rho}{C} \tag{4.18}$$

$$E = \frac{BF - (A^2 + B^2 + C^2)X_R \sin \rho}{C} \tag{4.19}$$

$$F = B(X_R \sin \rho) - A(Y_R \sin \rho) + C \cos \rho \tag{4.20}$$

Substituting D,E, and F into equation 4.12 gives

$$\begin{bmatrix} A_2 \\ B_2 \\ C_2 \end{bmatrix} = \begin{bmatrix} D \\ E \\ F \end{bmatrix} + \begin{bmatrix} A_X \\ B_Y \\ C_Z \end{bmatrix} \tag{4.21}$$

which is the vector $[A_1,B_1,C_1]'$ after being rotated. The rows of the orientation matrix are then computed by setting the position vector $[A_1,B_1,C_1]'$ successively equal to [1,0,0]', [0,1,0]', and [0,0,1]' and computing the transformed position vectors whose components are the elements of the orientation matrix.

Although there are several steps in this solution, it is fairly straightforward and an interesting application of the single rotation concept. It is included to show that a non-iterative two-star solution is possible without having to resort to fictitious stars as was done in the modification of the Church method for two stars.

5. Modifications for Aerial and Terrestrial Photogrammetry

The methods described thus far may be applied to aerial and terrestrial photogrammetry for cases in which the camera position is known and in which the rectangular coordinates of two or more imaged control points are known. The major difference that must be accounted for is that the direction cosines U_i, V_i, and W_i of the i^{th} control point in the object space are not defined in terms of right ascension and declination on the celestial sphere, but in terms of rectangular coordinates of some local coordinate system. Typically if

X_i, Y_i, Z_i = the rectangular coordinates of the i^{th} control point

and

X_C, Y_C, Z_C = the rectangular coordinates of the camera

then

$$U_i = \frac{X_i - X_C}{[(X_i - X_C)^2 + (Y_i - Y_C)^2 + (Z_i - Z_C)^2]^{1/2}}$$

$$V_i = \frac{Y_i - Y_C}{[(X_i - X_C)^2 + (Y_i - Y_C)^2 + (Z_i - Z_C)^2]^{1/2}}$$

$$W_i = \frac{Z_i - Z_C}{[(X_i - X_C)^2 + (Y_i - Y_C)^2 + (Z_i - Z_C)^2]^{1/2}} \qquad (5.1)$$

These values, rather than the corresponding celestial system values defined by equations 3.5, may be entered into the appropriate equations

of the previous sections. As aircraft positioning systems are improved the number of occasions where the coordinates of the camera are accurately known, but where its attitude is unknown may increase. In these cases there may be more opportunities to use the methods described in this paper.

6. Results

The results of executing the computer programs which implemented the methods described are summarized in Tables 6.4 and 6.5 and in Figures 6.1, 6.2, and 6.3. The sample data sets are given in Tables 6.1, 6.2, and 6.3. The purpose of executing the programs was only to verify the correctness of the equations and the computer code and not to compare the merits of the various methods, so only a small sample set was used. The data set was used first without errors and then again with errors added to the photo coordinates of the star images . All programs gave perfect results when executed with error free data, so those results are not tabulated. The results that are tabulated are from using the perturbed data given in Table 6.3 which has errors added to the image coordinates.

While the main purpose of executing the programs with sample data was to verify the correctness of the computer code, the results do give some indication of the relative performance of each method as far as accuracy is concerned. The results in Table 6.4 show that the collinearity equations method gave more accurate results than the Church method modified for four or more stars when the sample data set of four stars was used. Figures 6.1, 6.2, and 6.3 compare the errors from the collinearity equations method, the Church method modified for two stars, and the single rotation method using the six possible pairs of stars from the four in the sample data set. Here there seems to be no

clear superiority for either the collinearity equations method or the Church method modified for two stars even though on a case by case basis the errors from each method are different . The results from the single rotation method using star pairs 2,3 and 2,4 do not compare favorably with the collinearity and Church results, however. The reason for this is not clear. One possibility is that all the information available is not used, that is, the rotation angle could be estimated using either or both of the stars. The computer code only uses the star furthest from the rotation axis. A weighted mean of the two estimates might yield better results. Another possibility is that the sample happened to give ill-conditioned equations in the solution. In any case, the sample data set has two examples where the single rotation method gave less accurate results than the collinearity equation method or the Church method modified for two stars.

From a practical standpoint, further comparison of the methods is probably not justified unless there is some application where one method is to be used at the exclusion of the others. In that case, testing would be designed to determine whether the method under consideration would yield acceptable results.

Table 6.1 Simulated data set with error-free image coordinates and

ω = 156.1413452

ϕ = -18.7472372

k = 46.0052148

Star Index	Right Ascension	Declination	Image x-coordinate mm	Image y-coordinate mm	Principal Distance mm
1	-4^{o} 25' 49.68''	73^{o} 17' 20.53''	24.000	24.000	76.000
2	82^{o} 19' 23.81''	65^{o} 40' 39.65''	-15.000	15.000	76.000
3	67^{o} 24' 41.59''	38^{o} 35' 22.22''	-24.000	-24.000	76.000
4	23^{o} 46' 52.69''	52^{o} 14' 21.68''	20.000	-10.000	76.000

Table 6.2 Direction cosines of stars and star images for simulated data set

Star Index	U	V	W	P	Q	R
1	0.28668456	-0.02221252	0.95776749	0.28964233	0.28603683	-0.91339491
2	0.05501895	0.40817799	0.91124291	-0.19240306	0.19366887	-0.96201530
3	0.30023306	0.72167371	0.62373646	-0.28503185	-0.28863985	-0.95994801
4	0.56036911	0.24693381	0.79057584	0.25009172	-0.12630895	-0.95994801

Table 6.3 Simulated data set with perturbed image coordinates

Star Index	Right Ascension	Declination	Image x-coordinate mm	Image y-coordinate mm	Principal Distance mm
1	-4^{o} 25' 49.68''	73^{o} 17' 20.53''	24.100	23.800	76.000
2	82^{o} 19' 23.81''	65^{o} 40' 39.65''	-15.200	15.300	76.000
3	67^{o} 24' 41.59''	38^{o} 35' 22.22''	-23.700	-24.000	76.000
4	23^{o} 46' 52.69''	52^{o} 14' 21.68''	19.800	-10.000	76.000

Table 6.4 Errors in the orientation angles as determined from the sample data set using the collinearity equations and the modified Church method

Method	Omega error decimal degrees	Phi error decimal degrees	Kappa error decimal degrees
Collinearity Equations	0.0151	0.0216	-0.0605
Church method formula weighted	0.0521	0.2025	-0.2469
Church method uniformly weighted	0.0501	0.2033	-0.2318

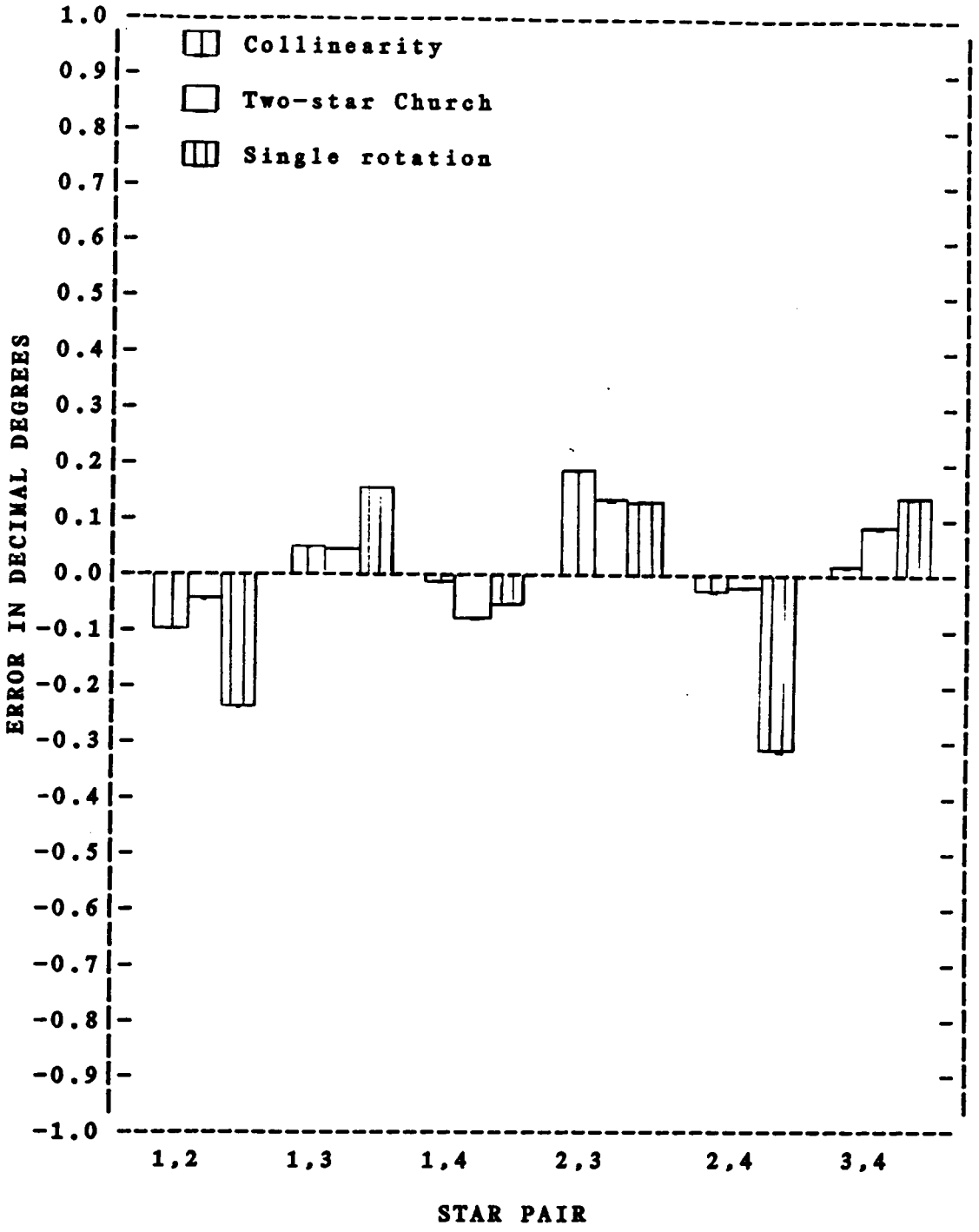

Figure 6.1 Errors in omega angle from solutions by the collinearity equation method, the two-star Church method, and the single rotation method using five different star pairs

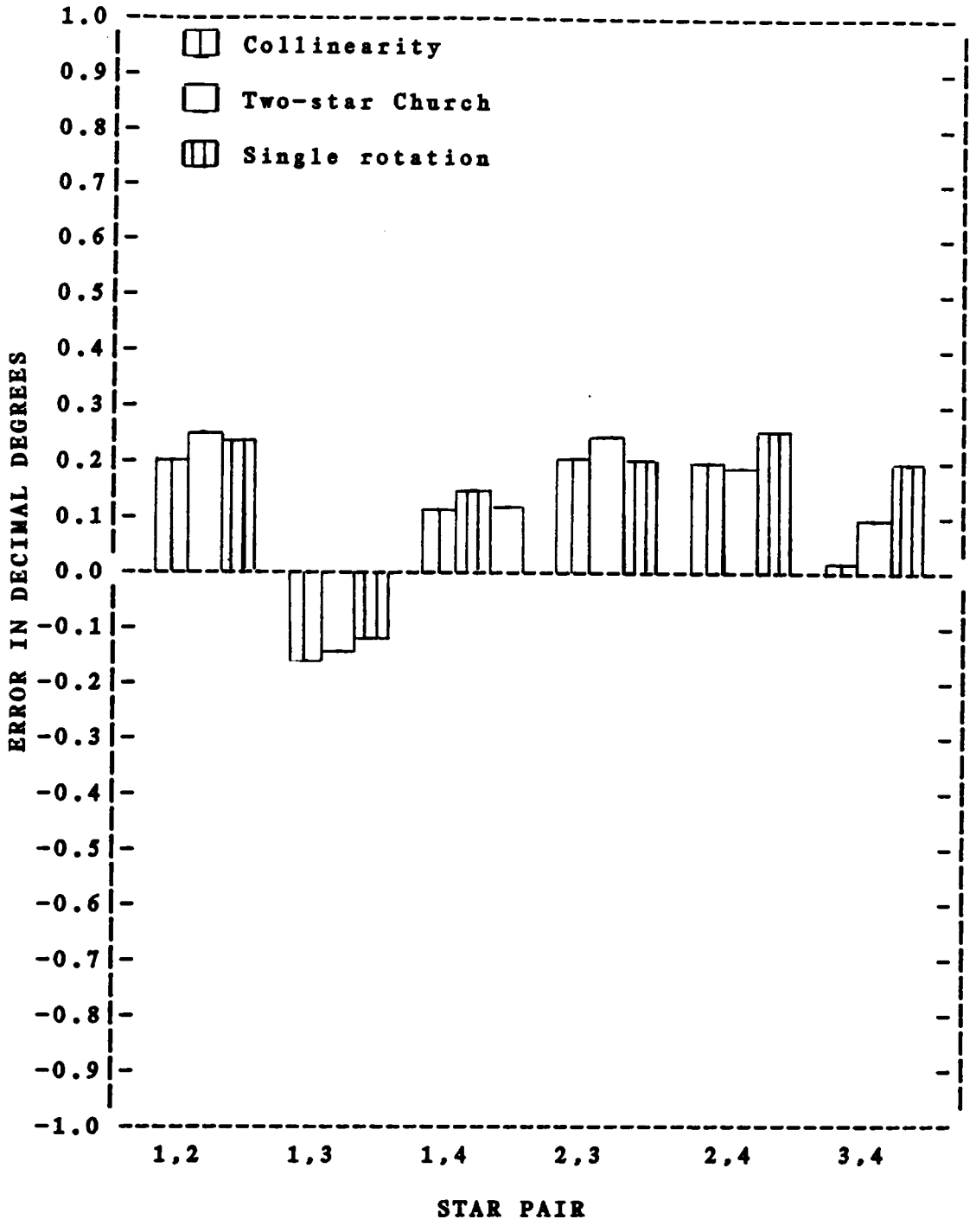

Figure 6.2 Errors in phi angle from solutions by the collinearity equation method, the two-star Church method, and the single rotation method using five different star pairs.

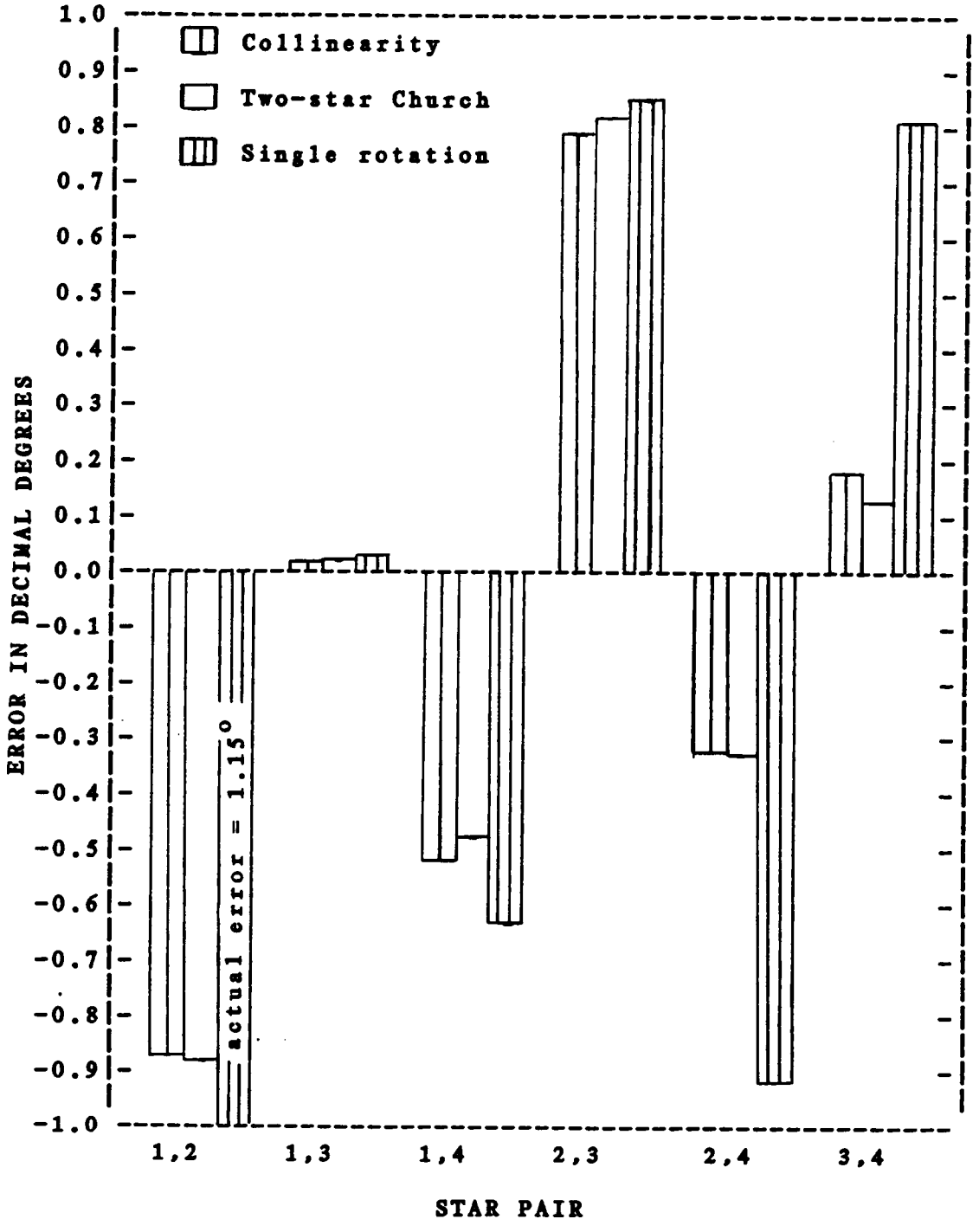

Figure 6.3 Errors in kappa angle from solutions by the collinearity equation method, the two-star Church method, and the single rotation method using five different star pairs.

7.Conclusions

The conclusions that can be drawn form this study are fairly evident. First, the collinearity equation method is the method of choice when large numbers of images are involved and high accuracy is required. Second, any of the modifications of the Church method may be used to calculate initial approximations to the orientation angles for the collinearity equation method. Third, the single rotation method could also be used to calculate initial approximations to the angles, but since the single rotation method is much more complex than the modifications of the Church method, its use for that purpose would require more computations and more programming effort.

8 . Recommendations

Five recommendations can be made based on this study and on observations of stellar photography:

1) studies could be conducted to determine whether or not there is a predictable relationship between the magnitude of a star and the standard error associated with measuring its image coordinates
2) studies could also be conducted to determine whether or not there is a predictable relationship between image elongation (caused by motion of the camera with respect to the celestial sphere) and the standard error associated with measuring its image coordinates
3) if either of the above studies should produce findings that imply that not all image coordinates should be given equal weights, then the effects of weighted observations, data editing, or iteratively reweighted least squares might be investigated
4) if data editing is a problem, then the possibility of using the modification of the Church method for two stars to perform pairwise solutions of the data should be examined
5) the possibility of using the modification for two stars in the full resection problem might be investigated

9. REFERENCES

Junkins, J. L. (1978). Introduction to Optimal Estimation of Dynamical Systems. Sijthoff and Noordhoff International Publishers B. V., Alphen aan den Rijn, Netherlands.

Mikhail, E. M. (1976). Observations and Least Squares. Harper and Row Publishers, Inc., New York, N.Y.

Moffitt, F. H., and Mikhail, E. M. (1980). Photogrammetry. Harper and Row Publishers, Inc., New York, N.Y.

Slama, C. C., Ed. (1980). Manual of Photogrammetry. 3rd ed., American Society of Photogrammetry, Falls Church, Va.

Spiegel, M. R. (1959). Theory and Problems of Vector Analysis. Schaum Publishing Company, New York, N.Y.

Thompsom, M. M., Ed. (1966). Manual of Photogrammetry. American Society of Photogrammetry, Falls Church, Va.

Wolf, P. R., (1974). Elements of Photogrammetry. McGraw-Hill, Inc., New York, N.Y.

Appendix A

Fictitious Data Generator

Appendix A contains the FORTRAN computer program RADEC2 which was used to generate fictitious data for program testing. The output is the right ascension and declination for simulated stars. The input is the principal distance (focal length) of the camera, the ω, ϕ, and k orientation angles and the x,y coordinates of the star images.

```
      PROGRAM RADEC2
C  THIS IS THE FICTITIOUS DATA GENERATOR...COMPUTING THE RIGHT ASCEN-
C  SION AND DECLINATION OF STARS GIVEN THEIR PHOTO COORDINATES AND THE
C  ORIENTATION AND FOCAL LENGTH OF THE CAMERA
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION  KAPPA
      DIMENSION ORMTRX( 3, 3) , X( 50 ), Y( 50 )
      DIMENSION COSALP( 50 ) , COSBET( 50 ) , COSGAM( 50 )
      DIMENSION RAYLEN( 50 )
      DIMENSION RTARAD( 50 ) , DECRAD( 50 )
      DIMENSION RIGHTA( 50 ) , DECLIN( 50 )
      DIMENSION U( 50 ) , V( 50 ) , W( 50 )
C
      RADDEG = 57.2957795131D00
      DEGRAD = 1.74532925199D-02
C
      NPRINT = 6
      N      = 0
C
      WRITE( 6, 1 )
1     FORMAT( /10X, ' ENTER PRINCIPAL DISTANCE : F15.6 FORMAT ')
      READ ( 5, 2 ) F
2     FORMAT( F15.6 )
      FSQ    = F * F
      WRITE( 6, 3 )
3     FORMAT( /10X , ' ENTER OMEGA IN DECIMAL DEGREES ' )
      READ ( 5,2 ) OMEGA
      WRITE ( 6, 4)
4     FORMAT( /10X ,' ENTER PHI IN DECIMAL DEGREES ' )
      READ ( 5, 2 ) PHI
      WRITE( 6, 5 )
5     FORMAT (  /10X , ' ENTER KAPPA IN DECIMAL DEGREES ' )
      READ ( 5 ,2 ) KAPPA
C
C     CONVERT ORIENTATION ANGLES TO RADIANS
C
      OMEGA     = OMEGA * DEGRAD
      PHI       = PHI   * DEGRAD
      KAPPA     = KAPPA * DEGRAD
C
C
      CALL JNKMAT( OMEGA, PHI, KAPPA, ORMTRX )
      WRITE( 6, 10 )
10    FORMAT( /10X, ' ORIENTATION MATRIX ' )
      WRITE( 6, 20 ) (( ORMTRX( I , J ), J = 1, 3 ), I = 1, 3)
111   WRITE( 6, 22 )
22    FORMAT( /10X, ' ENTER X  F15.6 OR 9999.9 TO TERMINATE')
```

```
      READ( 5, 2 ) TEMP
      IF((TEMP .GT. 9999.0) .OR. (N .GT. 49))GO TO 9999
      N         = N + 1
      X( N )    = TEMP
      WRITE( 6, 23 )
23    FORMAT( /10X, ' ENTER Y  F15.6 ' )
      READ( 5, 2 ) Y( N )
C
C     CALCULATE DIRECTION COSINES ( IN CAMERA SYSTEM ) OF
C     STAR IMAGE
C
      DISTSQ = X(N) * X(N)  +  Y(N) * Y(N)  + FSQ
      RAYLEN( N ) = DSQRT( DISTSQ )
C
      COSALP( N ) = X( N ) / RAYLEN( N )
      COSBET( N ) = Y( N ) / RAYLEN( N )
      COSGAM( N ) = -  F   / RAYLEN( N )
C
C     CALCULATE U, V, W FOR NTH STAR WHERE
C     ( COSALP COSBET COSGAM )'  = ORMTRX * ( U V W )
C
      U( N )    = ORMTRX( 1, 1 ) * COSALP( N )
     1          + ORMTRX( 2, 1 ) * COSBET( N )
     2          + ORMTRX( 3, 1 ) * COSGAM( N )
C
      V( N )    = ORMTRX( 1, 2 ) * COSALP( N )
     1          + ORMTRX( 2, 2 ) * COSBET( N )
     2          + ORMTRX( 3, 2 ) * COSGAM( N )
C
      W( N)     = ORMTRX( 1, 3 ) * COSALP( N )
     1          + ORMTRX( 2, 3 ) * COSBET( N )
     2          + ORMTRX( 3, 3 ) * COSGAM( N )
C
C
C     CALCULATE THE RIGHT ASCENSION AND DECLINATION FOR THE NTH STA
C
      RTARAD( N )    = DATAN2( V(N), U(N) )
      DECRAD( N )    = DASIN ( W(N) )
C
      RIGHTA( N )    = RTARAD( N ) * RADDEG
      DECLIN( N )    = DECRAD( N ) * RADDEG
C
      GO TO 111
9999  CONTINUE
20    FORMAT (   3( 7X, F12.9, 5X ), /   )
27    FORMAT( 1H , I3, 2F8.3  )
28    FORMAT( 1H ,' N ',I3,4X,'X ',F10.3,4X,'Y ',F10.3 )
```

```
29        FORMAT( 1H , 'COSALP',F11.9,' COSBET',F11.9,' COSGAM',
     1            F11.9 )
30        FORMAT( 1H , ' U',F11.9,' V',F11.9,' W',F11.9 )
31        FORMAT( 1H , ' RIGHT ASCENSION', F15.8,
     1         '   DECLINATION',F15.8 )
          DO  100 I = 1, N
               WRITE( NPRINT, 28 ) I, X(I), Y(I)
               WRITE( NPRINT, 29 )COSALP(I),COSBET(I),COSGAM(I)
               WRITE( NPRINT, 30 ) U(I), V(I), W(I)
               WRITE( NPRINT, 31 ) RIGHTA(I), DECLIN(I)
100       CONTINUE
          STOP
          END
```

Appendix B

Computer Program for the Collinearity Equation Method

Appendix B contains the FORTRAN computer program STARS which performs a stellar resection using the collinearity equation method.

```
         PROGRAM STARS
C      PROGRAM TO PERFORM STELLAR RESECTION
C      USING COLLINEARITY EQUATIONS
C234567891111111111222222222233333333334444444444555555555566666666667
C
       IMPLICIT DOUBLE PRECISION ( A-H, O-Z)
       INTEGER ONEORO
       DOUBLE PRECISION KAPPA
       LOGICAL ABFLAG, DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6
       LOGICAL DEBUG7, DEBUG8, DEBUG9, DEBU10,         DEBU11, DEBU12
       DIMENSION RIGHTA( 30 ), DECLIN( 30 ), X( 30 ), Y( 30 )
       DIMENSION    XWEIGH( 30 ), YWEIGH( 30 ), XRESID( 30 )
       DIMENSION    YRESID( 30 ), ABXRES( 30 ), ABYRES( 30 )
       DIMENSION    ORMTRX( 3, 3 ), ATWRES( 3 ), ATWA( 3, 3 )
       DIMENSION    ATWAIN( 3, 3 ), TOTRES( 30 )
C**********************************************************************
       COMMON /GIVEN/ F, X, Y , RIGHTA, DECLIN, N
       COMMON /ORIENT/ OMEGA, PHI, KAPPA, ORMTRX
       COMMON /RESNWT/XRESID, YRESID,   XWEIGH, YWEIGH ,
      1               SUMSQX, SUMSQY, SUMABX, SUMABY, ABXRES,ABYRES
       COMMON/UVW/ U, V, W
       COMMON/PARDRV/ PF1OME, PF1PHI, PF1KAP, PF2OME, PF2PHI, PF2KAP
       COMMON/MATVEC/ATWA, ATWAIN, ATWRES
       COMMON/DEBUGS/DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6,
      1              DEBUG7, DEBUG8, DEBUG9,DEBU10,DEBU11, DEBU12
       COMMON/FRACS/ XNUMER, YNUMER, DENOM
C
C23456789111111111122222222223333333333444444444455555555556666666667
       CALL RAWDAT
1      CONTINUE
       DO 990 ITERAT = 1, 4
       CALL ZEROOO
2      CONTINUE
       CALL JNKMAT( OMEGA, PHI, KAPPA, ORMTRX )
3      CONTINUE
       CALL PRTOPK( OMEGA, PHI, KAPPA )
4      CONTINUE
       DO 400 I = 1, N
          CALL NUMDEN( I )
5         CONTINUE
 666      CALL RESWTS( I )
6         CONTINUE
 777      CALL PARSHL
7         CONTINUE
 888      CALL ATWAM( I )
8         CONTINUE
          CALL ATWRSM( I )
```

```
400   CONTINUE
      CALL PRTRES( N, XRESID,YRESID, SUMSQX, SUMSQY, SUMABX, SUMABY)
9     CONTINUE
      CALL ATWAV
10    CONTINUE
      CALL NEWOPK
990   CONTINUE
999   CONTINUE
      STOP
      END
```

```
+     SUBROUTINE RAWDAT
C234567891111111111222222222233333333334444444444555555555566666666677
C   CALCULATES INPUT FOR STARS PROGRAM ... PHOTO COORDINATES, WITH
C   ERRORS IF NECESSARY, AND STAR COORDINATES
C
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z)
      INTEGER ONEORO
      DOUBLE PRECISION KAPPA
      LOGICAL ABFLAG, DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6
      LOGICAL DEBUG7, DEBUG8, DEBUG9, DEBU10, DEBU11        , DEBU12
      DIMENSION RIGHTA( 30 ), DECLIN( 30 ), X( 30 ), Y( 30 )
      DIMENSION    XWEIGH( 30 ), YWEIGH( 30 ), XRESID( 30 )
      DIMENSION    YRESID( 30 ), ABXRES( 30 ), ABYRES( 30 )
      DIMENSION    ORMTRX( 3, 3 ), ATWRES( 3 ), ATWA( 3, 3 )
      DIMENSION    ATWAIN( 3, 3 ), TOTRES( 30 )
      DIMENSION    DX( 30 ), DY( 30 )
      DIMENSION    JUMBL1( 24 ), JUMBL2( 24 ), JUMBL3( 24 )
      DIMENSION    CNX( 24 ), CNY( 24 )
      DIMENSION    DXRAD( 24 ), DYRAD ( 24 )
C*************************************************************************
      COMMON /GIVEN/ F, X, Y , RIGHTA, DECLIN, N
      COMMON /ORIENT/ OMEGA, PHI, KAPPA, ORMTRX
      COMMON /RESNWT/XRESID, YRESID,   XWEIGH, YWEIGH ,
     1               SUMSQX, SUMSQY, SUMABX, SUMABY, ABXRES, ABYRES
      COMMON/UVW/ U, V, W
      COMMON/PARDRV/ PF1OME, PF1PHI, PF1KAP, PF2OME, PF2PHI, PF2KAP
      COMMON/MATVEC/ATWA, ATWAIN, ATWRES
      COMMON/DEBUGS/ DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6,
     1       DEBUG7, DEBUG8, DEBUG9, DEBU10, DEBU11, DEBU12
C
C234567891111111111222222222233333333334444444444555555555566666666677
      DATA  JUMBL1/3,14,15,13,20,7,22,12,5,16,1,19,2,11,4,21,
     1             8,23,6,9,17,18,14,10/
      DATA  JUMBL2/23,12,24,1,2,19,5,20,14,22,11,13,16,9,8,7,
     1             21,18,4,3,15,10,6,17/
      DATA  JUMBL3/24,7,18,3,6,22,16,19,11,4,17,12,5,21,14,10,
     1             13,20,9,8,23,2,15,1/
      MAXITR       =     6
      F            = 76.000D0
      N            =      3
            IF( N .LT. 24 ) GO TO 718
C
  718 CONTINUE
      X( 1 )       = -17.32001D0
      Y( 1 )       = -10.00001D0
      RIGHTA( 1 )  = 157.0549065OD0
      DECLIN( 1 )  =  76.9704675OD0
```

```
C
      X( 2 )        =  17.320001D0
      Y( 2 )        = -10.000001D0
      RIGHTA( 2 )   = -141.69945479D0
      DECLIN( 2 )   =   32.17232937D0
C
      X( 3 )        =   0.000001D0
      Y( 3 )        = -40.000001D0
      RIGHTA( 3 )   =  177.94765773D0
      DECLIN( 3 )   =   54.96845735D0
C
      X( 1 )        =   -0.00300D0
      Y( 1 )        =   10.00300D0
      RIGHTA( 1 )   = -111.08453563D0
      DECLIN( 1 )   =   63.57809326D0
C
      X( 2 )        =    0.00300D0
      Y( 2 )        =   -0.00300D0
      RIGHTA( 2 )   = -128.07625064D0
      DECLIN( 2 )   =   65.18578304D0
C
      X( 3 )        =   -0.00900D0
      Y( 3 )        =  -20.00000D0
      RIGHTA( 3 )   = -161.29403653D0
      DECLIN( 3 )   =   62.41472141D0
C
C
C     CONVERT RIGHT ASCENSION AND DECLINATION TO RADIANS
C
      DO 1234 I = 1, N
         RIGHTA( I )    = RIGHTA( I ) / 57.2957795131D0
         DECLIN( I )    = DECLIN( I ) / 57.2957795131D0
1234  CONTINUE
C
      OMEGA         =    -0.609119908D0
      PHI           =     0.521853446D0
      KAPPA         =     0.172787595D0
C
      OMEGA         =    20.0D0 / 57.2967700001D0
      PHI           =   -15.0D0 / 57.2957795131D0
      KAPPA         =    50.0D0 / 57.2957795131D0
C
C     DO 51 I = 1, N
C         RADERR    =  0.05 * DSQRT( X(I)*X(I) + Y(I)*Y(I) )
C    1                  * DX( JUMBL2( I ))
C         THATA     = DATAN2( X( I ), Y( I ) )
C         COSTTA    = DCOS( THATA )
```

```
C          SINTTA    = DSIN( THATA )
C          DXRAD( I ) = RADERR * COSTTA
C          DYRAD( I ) = RADERR * SINTTA
C          WRITE( 6, 457 ) I, DXRAD( I ), DYRAD( I ), RADERR
C457       FORMAT( 1H , I5,2X, F10.6, 3X, F10.6, 3X, F10.6,
C     1            '   RADIAL ERRORS' )
C 51       CONTINUE
C      ADD ERRORS TO THE X, Y IMAGE COORDINATES
C      DO 555 I      =     1, N
C         X( I )     =     X( I ) + DXRAD( I ) + CNX( JUMBL1( 1 ))
C         Y( I )     =     Y( I ) + DYRAD( I ) + CNY( JUMBL2( 2 ))
 555   CONTINUE
C
C
       DEBUG1        = .FALSE.
       DEBUG2        = .FALSE.
       DEBUG3        = .FALSE.
       DEBUG4        = .FALSE.
       DEBUG5        = .FALSE.
       DEBUG6        = .FALSE.
       DEBUG7        = .FALSE.
       DEBUG8        = .FALSE.
       DEBUG9        = .FALSE.
       DEBU10        = .FALSE.
       DEBU11        = .FALSE.
       DEBU12        = .FALSE.
       WRITE( 6, 5 )
   5   FORMAT(1H ,'PT    X ERROR     Y ERROR  X COORDINATE Y ',
      1 'COORDINATE' )
C
       DO 10 I = 1, N
            WRITE( 6, 100 )I, DX( I ), DY( I ), X( I ), Y( I )
  10   CONTINUE
  100  FORMAT( 1H , I2, 2X, F10.6, 2X, F10.6, 2X, F10.6, 2X, F10.6)
       RETURN
       END
```

```
      SUBROUTINE ZEROOO
C234567891111111111222222222233333333334444444444555555555566666666677
C INITIALIZES ARRAYS FOR STARS PROGRAM
C
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z)
      INTEGER ONEORO
      DOUBLE PRECISION KAPPA
      LOGICAL ABFLAG, DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6
      LOGICAL DEBUG7, DEBUG8, DEBUG9, DEBU10, DEBU11, DEBU12
      DIMENSION RIGHTA( 30 ), DECLIN( 30 ), X( 30 ), Y( 30 )
      DIMENSION    XWEIGH( 30 ), YWEIGH( 30 ), XRESID( 30 )
      DIMENSION    YRESID( 30 ), ABXRES( 30 ), ABYRES( 30 )
      DIMENSION    ORMTRX( 3, 3 ), ATWRES( 3 ), ATWA( 3, 3 )
      DIMENSION    ATWAIN( 3, 3 ), TOTRES( 30 )
C**********************************************************************
      COMMON /GIVEN/ F, X, Y , RIGHTA, DECLIN, N
      COMMON /ORIENT/ OMEGA, PHI, KAPPA, ORMTRX
      COMMON /RESNWT/XRESID, YRESID,   XWEIGH, YWEIGH,
     1               SUMSQX, SUMSQY, SUMABX, SUMABY,ABXRES,ABYRES
      COMMON/UVW/ U, V, W
      COMMON/PARDRV/ PF1OME, PF1PHI, PF1KAP, PF2OME, PF2PHI, PF2KAP
      COMMON/MATVEC/ATWA, ATWAIN, ATWRES
      COMMON/DEBUGS/DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6,
     1              DEBUG7, DEBUG8, DEBUG9, DEBU10, DEBU11, DEBU12
C
C234567891111111111222222222233333333334444444444555555555566666666677
C     ZERO OUT MATRICES
C
         DO 600 JJ = 1, 3
              ATWRES( JJ ) = 0.0D0
              DO 500 II = 1, 3
                   ATWA( II, JJ) =0.0D0
                   ATWAIN( II, JJ ) = 0.0D0
500             CONTINUE
600      CONTINUE
C
         SUMSQX    = 0.0D0
         SUMSQY    = 0.0D0
         SUMABX    = 0.0D0
         SUMABY    = 0.0D0
C
      IF( DEBUG1 )  WRITE( 6, 100 ) SUMABY
100   FORMAT( 1H , ' IN ZEROOO SUMABY = ', F7.3 )
      RETURN
      END
```

```
      SUBROUTINE JNKMAT( OMEGA, PHI, KAPPA, ORMTRX )
C COMPUTES JUNKINS'S VERSION OF THE ORIENTATION MATRIX FROM OMEGA,PHI,
C AND KAPPA
      DOUBLE PRECISION OMEGA, PHI, KAPPA, ORMTRX( 3, 3 )
      DOUBLE PRECISION CO , SO , CP , SP , CK , SK
C
      CO   = DCOS( OMEGA )
      SO   = DSIN( OMEGA )
      CP   = DCOS( PHI )
      SP   = DSIN( PHI )
      CK   = DCOS( KAPPA )
      SK   = DSIN( KAPPA )
C
      ORMTRX( 1, 1 ) = CK * CP
      ORMTRX( 1, 2 ) = SO * SP * CK + CO * SK
      ORMTRX( 1, 3 ) = -CO * SP * CK + SO * SK
      ORMTRX( 2, 1 ) = -SK * CP
      ORMTRX( 2, 2) = -SO * SP * SK + CO * CK
      ORMTRX( 2, 3) = CO * SP * SK + SO * CK
      ORMTRX( 3, 1 ) = SP
      ORMTRX( 3 ,2) = -SO * CP
      ORMTRX( 3, 3) = CO * CP
      RETURN
      END
```

```
      SUBROUTINE PRTOPK( OMEGA, PHI, KAPPA )
C
C     PRINTS THREE DOUBLE PRECISION RADIAN ANGLES IN DEGREES, MINUTES
C     SECONDS FORMAT.
C
      DOUBLE PRECISION   OMEGA, PHI, KAPPA
      CALL DPRDMS ( OMEGA, IDEGO, MINOME, SECOME )
      CALL DPRDMS ( PHI  , IDEGP, MINPHI, SECPHI )
      CALL DPRDMS ( KAPPA, IDEGK, MINKAP, SECKAP )
C
      WRITE ( 6, 10 ) IDEGO, MINOME, SECOME
      WRITE ( 6, 20 ) IDEGP, MINPHI, SECPHI
      WRITE ( 6, 30 ) IDEGK, MINKAP, SECKAP
10    FORMAT ( ' OMEGA  =', I6, I4, F8.3 )
20    FORMAT ( ' PHI    =', I6, I4, F8.3 )
30    FORMAT ( ' KAPPA  =', I6, I4, F8.3 )
C
      RETURN
      END
```

```
      SUBROUTINE NUMDEN( I )
C234567891111111111222222222233333333334444444444555555555566666666677
C  COMPUTES THE NUMERATOR AND DENOMINATOR FOR THE PARTIAL DERIVATIVE
C  IN THE PARSHL SUBROUTINE
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z)
      INTEGER ONEORO
      DOUBLE PRECISION KAPPA
      LOGICAL ABFLAG, DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6
      LOGICAL DEBUG7, DEBUG8, DEBUG9, DEBU10,         DEBU11, DEBU12
      DIMENSION RIGHTA( 30 ), DECLIN( 30 ), X( 30 ), Y( 30 )
      DIMENSION    XWEIGH( 30 ), YWEIGH( 30 ), XRESID( 30 )
      DIMENSION    YRESID( 30 ), ABXRES( 30 ), ABYRES( 30 )
      DIMENSION    ORMTRX( 3, 3 ), ATWRES( 3 ), ATWA( 3, 3 )
      DIMENSION    ATWAIN( 3, 3 ), TOTRES( 30 )
C*****************************************************************************
      COMMON /GIVEN/ F, X, Y , RIGHTA, DECLIN, N
      COMMON /ORIENT/ OMEGA, PHI, KAPPA, ORMTRX
      COMMON /RESNWT/XRESID, YRESID,   XWEIGH, YWEIGH ,
     1               SUMSQX, SUMSQY, SUMABX, SUMABY, ABXRES, ABYRES
      COMMON/UVW/ U, V, W
      COMMON/PARDRV/ PF1OME, PF1PHI, PF1KAP, PF2OME, PF2PHI, PF2KAP
      COMMON/MATVEC/ATWA, ATWAIN, ATWRES
      COMMON/DEBUGS/ DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6,
     1               DEBUG7, DEBUG8, DEBUG9, DEBU10, DEBU11, DEBU12
      COMMON/FRACS/ XNUMER, YNUMER, DENOM
C
C234567891111111111222222222233333333334444444444555555555566666666677
C
C     COMPUTE U, V, W  - THE DIRECTION COSINES IN THE STELLAR COORDINAT
C     SYSTEM OF THE ITH STAR.
C
         U = DCOS( DECLIN( I )) * DCOS( RIGHTA( I ))
         V = DCOS( DECLIN( I )) * DSIN( RIGHTA( I ))
         W = DSIN( DECLIN( I ))
C
C
C     COMPUTE XNUMER, YNUMER, AND DENOM FOR USE IN THE COLLINEARITY
C     EQUATIONS WHERE:
C        F1 = X + F( XNUMER / DENOM) = 0 ( APPROXIMATELY )
C        F2 = Y + F( YNUMER / DENOM) = 0 ( APPROXIMATELY )
C
      XNUMER  = ORMTRX( 1, 1 ) *  U
     1        + ORMTRX( 1, 2 ) *  V
     2        + ORMTRX( 1, 3 ) *  W
C
      YNUMER  = ORMTRX( 2, 1 ) *  U
     1        + ORMTRX( 2 ,2 ) *  V
```

```
     2         + ORMTRX( 2, 3 ) *  W
C
C
      DENOM    = ORMTRX( 3, 1 ) *  U
     1         + ORMTRX( 3, 2 ) *  V
     2         + ORMTRX( 3, 3 ) *  W
C
C
C
      IF ( DEBUG9 )WRITE( 6, 10 ) I, U, V, W
     1   , XNUMER, YNUMER, DENOM
10    FORMAT( 1H , ' IN DENOM I U V W = ', I3, 6D10.4  )
      RETURN
      END
```

```
      SUBROUTINE RESWTS( I )
C234567891111111111222222222233333333334444444444555555555566666666677
C  COMPUTES THE RESIDUALS FROM AN ITERATION OF THE LINEARIZED COLLIN-
C  EARITY EQUATIONS AND REWEIGHTS FOR AN ITERATIVELY REWEIGHTED
C  SOLUTION IF DESIRED
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z)
      INTEGER ONEORO
      DOUBLE PRECISION KAPPA
      LOGICAL ABFLAG, DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6
      LOGICAL DEBUG7, DEBUG8, DEBUG9, DEBU10,         DEBU11, DEBU12
      DIMENSION RIGHTA( 30 ), DECLIN( 30 ), X( 30 ), Y( 30 )
      DIMENSION    XWEIGH( 30 ), YWEIGH( 30 ), XRESID( 30 )
      DIMENSION    YRESID( 30 ), ABXRES( 30 ), ABYRES( 30 )
      DIMENSION    ORMTRX( 3, 3 ), ATWRES( 3 ), ATWA( 3, 3 )
      DIMENSION    ATWAIN( 3, 3 ), TOTRES( 30 )
C*************************************************************************
      COMMON /GIVEN/ F, X, Y , RIGHTA, DECLIN, N
      COMMON /ORIENT/ OMEGA, PHI, KAPPA, ORMTRX
      COMMON /RESNWT/XRESID, YRESID,   XWEIGH, YWEIGH ,
     1               SUMSQX, SUMSQY, SUMABX, SUMABY,ABXRES, ABYRES
      COMMON/UVW/ U, V, W
      COMMON/PARDRV/ PF1OME, PF1PHI, PF1KAP, PF2OME, PF2PHI, PF2KAP
      COMMON/MATVEC/ATWA, ATWAIN, ATWRES
      COMMON/DEBUGS/DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6,
     1              DEBUG7, DEBUG8, DEBUG9, DEBU10, DEBU11, DEBU12
      COMMON/FRACS/XNUMER, YNUMER, DENOM
C
C234567891111111111222222222233333333334444444444555555555566666666677
C
C     COMPUTE X RESIDUALS AND Y RESIDUALS
C
      XRESID( I )  = X( I ) + F * ( XNUMER / DENOM )
      YRESID( I )  = Y( I ) + F * ( YNUMER / DENOM )
      ABXRES( I )  = DABS( XRESID( I ))
      ABYRES( I )  = DABS( YRESID( I ))
C
      SUMSQX  = SUMSQX  + XRESID( I ) * XRESID( I )
      SUMSQY  = SUMSQY  + YRESID( I ) * YRESID( I )
      SUMABX  = SUMABX  + ABXRES( I )
      SUMABY  = SUMABY  + ABYRES( I )
C
      XWEIGH( I )  = 1.0D0
      YWEIGH( I )  = 1.0D0
C
C     GO TO 99
C
 6057 TOOLTL  = 0.0000001D0
```

```
      ABFLAG  = .TRUE.
      IF ( .NOT. ABFLAG ) GO TO 99
      IF ( ABXRES( I ) .LT. TOOLTL ) XWEIGH( I ) = 0.0D0
      IF ( ABYRES( I ) .LT. TOOLTL ) YWEIGH( I ) = 0.0D0
      IF ( ABXRES( I ) .GE. TOOLTL ) XWEIGH( I ) =
     1 0.000009D0 / ABXRES( I ) **2
C     1 DEXP( -0.3D0 * ( ABXRES( I ) / 0.003D0 ) )
      IF ( ABYRES( I ) .GE. TOOLTL ) YWEIGH( I ) =
     1 0.000009D0 / ABYRES( I ) **2
C     1 DEXP( -0.3D0 * ( ABYRES( I ) / 0.003D0 ) )
C
      IF ( XWEIGH ( I ) .GT. 9.0D0 ) XWEIGH( I ) = 3.0D0
      IF ( YWEIGH ( I ) .GT. 9.0D0 ) YWEIGH( I ) = 3.0D0
99    CONTINUE
      IF( DEBUG6 )  WRITE( 6, 10 ) I, XWEIGH( I ), YWEIGH( I )
10    FORMAT( 1H , I4, ' XWEIGH =',D14.9, '  YWEIGH = ', D14.9 )
      RETURN
      END
```

```
      SUBROUTINE PARSHL
C COMPUTES PARTIAL DERIVATIVES FOR COLLINEARITY EQUATIONS
C234567891111111111222222222233333333334444444444555555555566666666677
C
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z)
      INTEGER ONEORO
      DOUBLE PRECISION KAPPA
      LOGICAL ABFLAG, DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6
      LOGICAL DEBUG7, DEBUG8, DEBUG9, DEBU10,         DEBU11, DEBU12
      DIMENSION RIGHTA( 30 ), DECLIN( 30 ), X( 30 ), Y( 30 )
      DIMENSION    XWEIGH( 30 ), YWEIGH( 30 ), XRESID( 30 )
      DIMENSION    YRESID( 30 ), ABXRES( 30 ), ABYRES( 30 )
      DIMENSION    ORMTRX( 3, 3 ), ATWRES( 3 ), ATWA( 3, 3 )
      DIMENSION    ATWAIN( 3, 3 ), TOTRES( 30 )
C*********************************************************************
      COMMON /GIVEN/ F, X, Y , RIGHTA, DECLIN, N
      COMMON /ORIENT/ OMEGA, PHI, KAPPA, ORMTRX
      COMMON /RESNWT/XRESID, YRESID,   XWEIGH, YWEIGH ,
     1               SUMSQX, SUMSQY, SUMABX, SUMABY, ABXRES, ABYRES
      COMMON/UVW/ U, V, W
      COMMON/PARDRV/ PF1OME, PF1PHI, PF1KAP, PF2OME, PF2PHI, PF2KAP
      COMMON/MATVEC/ATWA, ATWAIN, ATWRES
      COMMON/DEBUGS/DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6,
     1           DEBUG7, DEBUG8, DEBUG9,DEBU10,DEBU11,DEBU12
      COMMON/FRACS/XNUMER, YNUMER, DENOM
C
C234567891111111111222222222233333333334444444444555555555566666666677
C
C
C     COMPUTE THE PARTIAL DERIVATIVES OF XNUMER, YNUMER, AND DENOM
C     WITH RESPECT TO THE ORIENTATION ANGLES OMEGA, PHI, AND KAPPA.
C     THESE WILL BE USED IN COMPUTING THE PARTIAL DERIVATIVES OF F1 AND
C     WITH RESPECT TO OMEGA, PHI, AND KAPPA.
C
      COSOME  = DCOS( OMEGA )
      SINOME  = DSIN( OMEGA )
      COSPHI  = DCOS( PHI   )
      SINPHI  = DSIN( PHI   )
      COSKAP  = DCOS( KAPPA )
      SINKAP  = DSIN( KAPPA )
C
      IF( DEBUG7 ) WRITE( 6, 77 ) U, V, W
  77  FORMAT( 1H , ' IN PARSHL U,V,W = ', 3F14.7 )
      IF( DEBUG7 ) WRITE( 6, 78 )(( ORMTRX(II,JJ),JJ=1,3),II=1,3)
  78  FORMAT( 1H , ' IN PARSHL ORMTRX = ', 3( F12.9,4X ))
C
C     PARTIAL DERIVATIVE OF XNUMER WITH RESPECT TO OMEGA
```

```
C
      PXNOME  = -ORMTRX( 1, 3 ) * V
     1          +ORMTRX( 1, 2 ) * W
C
      IF( DEBUG7 ) WRITE( 6, 80 ) PXNOME
   80 FORMAT(1H ,'PXNOME = ', F20.9 )
C
C     PARTIAL DERIVATIVE OF XNUMER WITH RESPECT TO PHI
C
      PXNPHI  =-SINPHI * COSKAP * U
     1        + SINOME * COSPHI * COSKAP * V
     2        - COSOME * COSPHI * COSKAP * W
C
      IF(DEBUG7 ) WRITE( 6,81 ) PXNPHI
   81 FORMAT( 1H , ' PXNPHI = ', F20.9 )
C
C
C     PARTIAL DERIVATIVE OF XNUMER WITH RESPECT TO KAPPA
C
      PXNKAP  = ORMTRX( 2, 1 ) * U
     1        + ORMTRX( 2, 2 ) * V
     2        + ORMTRX( 2, 3 ) * W
C
      IF( DEBUG7 ) WRITE( 6, 82 )PXNKAP
   82 FORMAT( 1H , 'PXNKAP = ', F20.9 )
C
C
C     PARTIAL DERIVATIVE OF YNUMER WITH RESPECT TO OMEGA
C
      PYNOME  = - ORMTRX( 2, 3 ) * V
     1          + ORMTRX( 2, 2 ) * W
C
      IF(DEBUG7 ) WRITE ( 6, 83 ) PYNOME
   83 FORMAT( 1H , ' PYNOME = ', F20.9 )
C
C
C     PARTIAL DERIVATIVE OF YNUMER WITH RESPECT TO PHI
C
      PYNPHI  = SINKAP * SINPHI * U
     1        - SINOME * COSPHI * SINKAP * V
     2        + COSOME * COSPHI * SINKAP * W
C
      IF( DEBUG7 )WRITE( 6, 84 ) PYNPHI
   84 FORMAT( 1H , ' PYNPHI = ', F20.9 )
C
C
C     PARTIAL DERIVATIVE OF YNUMER WITH RESPECT TO KAPPA
```

```
C
      PYNKAP  =-ORMTRX( 1, 1 ) * U
     1        - ORMTRX( 1, 2 ) * V
     2        - ORMTRX( 1, 3 ) * W
C
      IF ( DEBUG7 ) WRITE ( 6, 85 ) PYNKAP
 85   FORMAT ( 1H , ' PYNKAP = ', F20.9 )
C
C
C     PARTIAL OF DENOM WITH RESPECT TO OMEGA
C
      PDEOME  = - ORMTRX( 3, 3 ) * V
     1          + ORMTRX( 3, 2 ) * W
C
      IF( DEBUG7 ) WRITE( 6, 86 ) PDEOME
 86   FORMAT( 1H , ' PDEOME = ', F20.9 )
C
C
C     PARTIAL DERIVATIVE OF DENOM WITH RESPECT TO PHI
C
      PDEPHI  =   COSPHI * U
     1          + SINOME * SINPHI * V
     2          - COSOME * SINPHI * W
C
      IF( DEBUG7 ) WRITE ( 6, 87 ) PDEPHI
 87   FORMAT( 1H , ' PDEPHI = ', F20.9 )
C
C
C     DEFINE FOVDEN = F / DENOM
C
      FOVDEN  = F / DENOM
C
      IF( DEBUG7 )WRITE( 6, 79 ) F, DENOM
 79   FORMAT( 1H , ' IN PARSHL F = ',F10.3,'  DENOM = ',F20.5 )
C
      IF( DEBUG7 ) WRITE ( 6, 88 ) XNUMER , YNUMER
 88   FORMAT( 1H ,' XNUMER = ', F20.9,'YNUMER = ', F20.9 )
C
C     PARTIAL DERIVATIVES OF F1 WITH RESPECT TO OMEGA, PHI, AND KAPPA
C
      PF1OME  = FOVDEN * ( PXNOME - ( XNUMER / DENOM ) * PDEOME )
C
      PF1PHI  = FOVDEN * ( PXNPHI - ( XNUMER / DENOM ) * PDEPHI )
C
      PF1KAP  = FOVDEN * PXNKAP
C
C     PARTIAL DERIVATIVES OF F2 WITH RESPECT TO OMEGA, PHI, AND KAPPA
```

```
C
      PF2OME  = FOVDEN * ( PYNOME - ( YNUMER / DENOM ) * PDEOME )
C
      PF2PHI  = FOVDEN * ( PYNPHI - ( YNUMER / DENOM ) * PDEPHI )
C
      PF2KAP  = FOVDEN * PYNKAP
C
      IF ( DEBUG7 ) WRITE( 6, 11 )PF1OME, PF1PHI, PF1KAP,
     1                            PF2OME, PF2PHI, PF2KAP
 11   FORMAT( 1H , ' IN PARSHL PARTIALS OF F1 AND F2 '/ 3D20.9 )
      RETURN
      END
```

```
      SUBROUTINE ATWAM( I )
C234567891111111111222222222233333333334444444444555555555566666666677
C
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z)
      INTEGER ONEORO
      DOUBLE PRECISION KAPPA
      LOGICAL ABFLAG, DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6
      LOGICAL DEBUG7, DEBUG8, DEBUG9, DEBU10,         DEBU11, DEBU12
      DIMENSION RIGHTA( 30 ), DECLIN( 30 ), X( 30 ), Y( 30 )
      DIMENSION    XWEIGH( 30 ), YWEIGH( 30 ), XRESID( 30 )
      DIMENSION    YRESID( 30 ), ABXRES( 30 ), ABYRES( 30 )
      DIMENSION    ORMTRX( 3, 3 ), ATWRES( 3 ), ATWA( 3, 3 )
      DIMENSION    ATWAIN( 3, 3 ), TOTRES( 30 )
C*****************************************************************
      COMMON /GIVEN/ F, X, Y , RIGHTA, DECLIN, N
      COMMON /ORIENT/ OMEGA, PHI, KAPPA, ORMTRX
      COMMON /RESNWT/XRESID, YRESID,   XWEIGH, YWEIGH ,
     1               SUMSQX, SUMSQY, SUMABX, SUMABY, ABXRES, ABYRES
      COMMON/UVW/ U, V, W
      COMMON/PARDRV/ PF1OME, PF1PHI, PF1KAP, PF2OME, PF2PHI, PF2KAP
      COMMON/MATVEC/ATWA, ATWAIN, ATWRES
      COMMON/DEBUGS/DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6,
     1           DEBUG7, DEBUG8, DEBUG9,DEBU10,DEBU11,DEBU12
C
C234567891111111111222222222233333333334444444444555555555566666666677
C
C     FORM THE ATWA MATRIX ( A TRANSPOSE TIMES THE WEIGHT MATRIX TIMES
C     A ).
C
      ATWA( 1, 1 ) = ATWA( 1, 1 ) + PF1OME * PF1OME * XWEIGH( I )
     1             + PF2OME * PF2OME * YWEIGH( I )
C
      ATWA( 1, 2 ) = ATWA( 1, 2 ) + PF1OME * PF1PHI * XWEIGH( I )
     1             + PF2OME * PF2PHI * YWEIGH( I )
C
      ATWA( 1, 3 ) = ATWA( 1, 3 ) + PF1OME * PF1KAP * XWEIGH( I )
     1             + PF2OME * PF2KAP * YWEIGH( I )
C
      ATWA( 2, 2 ) = ATWA( 2, 2 ) + PF1PHI * PF1PHI * XWEIGH( I )
     1             + PF2PHI * PF2PHI * YWEIGH( I )
C
      ATWA( 2, 3 ) = ATWA( 2, 3 ) + PF1PHI * PF1KAP * XWEIGH( I )
     1             + PF2PHI * PF2KAP * YWEIGH( I )
C
      ATWA( 3, 3 ) = ATWA( 3, 3 ) + PF1KAP * PF1KAP * XWEIGH( I )
     1             + PF2KAP *  PF2KAP * YWEIGH( I )
C
```

```
      IF ( DEBUG8 )WRITE( 6,10 )((ATWA( J, K ),K=1,3), J = 1, 3 )
10    FORMAT( 1H ,' ATWA   ', 3F20.9    )
      RETURN
      END
```

```
      SUBROUTINE ATWRSM( I )
C234567891111111111222222222233333333334444444444555555555566666666677
C
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z)
      INTEGER ONEORO
      DOUBLE PRECISION KAPPA
      LOGICAL ABFLAG, DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6
      LOGICAL DEBUG7, DEBUG8, DEBUG9, DEBU10,         DEBU11, DEBU12
      DIMENSION RIGHTA( 30 ), DECLIN( 30 ), X( 30 ), Y( 30 )
      DIMENSION    XWEIGH( 30 ), YWEIGH( 30 ), XRESID( 30 )
      DIMENSION    YRESID( 30 ), ABXRES( 30 ), ABYRES( 30 )
      DIMENSION    ORMTRX( 3, 3 ), ATWRES( 3 ), ATWA( 3, 3 )
      DIMENSION    ATWAIN( 3, 3 ), TOTRES( 30 )
C**********************************************************************
      COMMON /GIVEN/ F, X, Y , RIGHTA, DECLIN, N
      COMMON /ORIENT/ OMEGA, PHI, KAPPA, ORMTRX
      COMMON /RESNWT/XRESID, YRESID,   XWEIGH, YWEIGH ,
     1               SUMSQX, SUMSQY, SUMABX, SUMABY, ABXRES, ABYRES
      COMMON/UVW/ U, V, W
      COMMON/PARDRV/ PF1OME, PF1PHI, PF1KAP, PF2OME, PF2PHI, PF2KAP
      COMMON/MATVEC/ATWA, ATWAIN, ATWRES
      COMMON/DEBUGS/DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6,
     1           DEBUG7, DEBUG8, DEBUG9,DEBU10,DEBU11,DEBU12
C
C234567891111111111222222222233333333334444444444555555555566666666677
C
C     FORM THE ATWRES MATRIX ( A TRANSPOSE TIMES THE WEIGHT MATRIX
C     TIMES THE VECTOR OF RESIDUALS )
C
      ATWRES( 1 )  = ATWRES( 1 ) + PF1OME * XWEIGH( I ) * XRESID( I )
     1                           + PF2OME * YWEIGH( I ) * YRESID( I )
C
      ATWRES( 2 )  = ATWRES( 2 ) + PF1PHI * XWEIGH( I ) * XRESID( I )
     1                           + PF2PHI * YWEIGH( I ) * YRESID( I )
C
      ATWRES( 3 )  = ATWRES( 3 ) + PF1KAP * XWEIGH( I ) * XRESID( I )
     1                           + PF2KAP * YWEIGH( I ) * YRESID( I )
C
400   CONTINUE
C
      IF( DEBUG9 )WRITE( 6, 10 ) ( ATWRES( J ), J = 1, 3 )
 10   FORMAT( 1H , ' IN ATWRSM ATWRES = ', 3F20.9 )
      RETURN
      END
```

```
      SUBROUTINE PRTRES( N, XRESID, YRESID, SUMSQX, SUMSQY
     1                 , SUMABX, SUMABY )
C  PRINTS OUT THE PLATE COORDINATE RESIDUALS FROM AN ITERATION OF STARS
      IMPLICIT DOUBLE PRECISION  ( A-H, O-Z )
      DIMENSION    XRESID( N ), YRESID( N )
      WRITE( 6,20 )
      WRITE( 6, 10 ) ( I, XRESID( I ), YRESID( I ), I = 1, N)
10    FORMAT ( 1H , I10, 10X, F10.4, 10X, F10.4 )
20    FORMAT ( '   OBSERVATION NUMBER', 10X, 'X RESIDUAL', 10X,
     1             'Y RESIDUAL' )
      WRITE( 6, 30 ) SUMSQX
      WRITE( 6, 31 ) SUMSQY
      WRITE( 6, 32 ) SUMABX
      WRITE( 6, 33 ) SUMABY
C
30    FORMAT ( ' SUM OF THE SQUARED X RESIDUALS =', F14.7 )
31    FORMAT ( ' SUM OF THE SQUARED Y RESIDUALS =', F14.7 )
32    FORMAT ( ' SUM OF THE ABSOLUTE X RESIDUALS =', F13.7 )
33    FORMAT ( ' SUM OF THE ABSOLUTE Y RESIDUALS =', F13.7 )
      RETURN
      END
```

```
      SUBROUTINE ATWAV
C234567891111111111222222222233333333334444444444555555555566666666677
C
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z)
      INTEGER ONEORO
      DOUBLE PRECISION KAPPA
      LOGICAL ABFLAG, DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6
      LOGICAL DEBUG7, DEBUG8, DEBUG9, DEBU10,         DEBU11, DEBU12
      DIMENSION RIGHTA( 30 ), DECLIN( 30 ), X( 30 ), Y( 30 )
      DIMENSION    XWEIGH( 30 ), YWEIGH( 30 ), XRESID( 30 )
      DIMENSION    YRESID( 30 ), ABXRES( 30 ), ABYRES( 30 )
      DIMENSION    ORMTRX( 3, 3 ), ATWRES( 3 ), ATWA( 3, 3 )
      DIMENSION    ATWAIN( 3, 3 ), TOTRES( 30 )
C*****************************************************************************
      COMMON /GIVEN/ F, X, Y , RIGHTA, DECLIN, N
      COMMON /ORIENT/ OMEGA, PHI, KAPPA, ORMTRX
      COMMON /RESNWT/XRESID, YRESID,   XWEIGH, YWEIGH ,
     1               SUMSQX, SUMSQY, SUMABX, SUMABY, ABXRES, ABYRES
      COMMON/UVW/ U, V, W
      COMMON/PARDRV/ PF1OME, PF1PHI, PF1KAP, PF2OME, PF2PHI, PF2KAP
      COMMON/MATVEC/ATWA, ATWAIN, ATWRES
      COMMON/DEBUGS/DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6,
     1          DEBUG7, DEBUG8, DEBUG9,DEBU10,DEBU11,DEBU12
C
C234567891111111111222222222233333333334444444444555555555566666666677
C
C     FORM ATWAIN - THE INVERSE OF ATWA - USING THE ADJOINT METHOD
C
      DET     = ATWA( 1, 1 ) * ATWA( 2, 2 ) * ATWA( 3, 3 )
     1        - ATWA( 1, 1 ) * ATWA( 2, 3 ) * ATWA( 2, 3 )
     2        - ATWA( 1, 2 ) * ATWA( 1, 2 ) * ATWA( 3, 3 )
     3        + ATWA( 1, 2 ) * ATWA( 1, 3 ) * ATWA( 2, 3 )
     4        + ATWA( 1, 3 ) * ATWA( 1, 2 ) * ATWA( 2, 3 )
     5        - ATWA( 1, 3 ) * ATWA( 1, 3 ) * ATWA( 2, 2 )
C
C
      ATWAIN( 1, 1 )    =
     1(ATWA( 2, 2 ) * ATWA( 3, 3 ) - ATWA( 2, 3 ) * ATWA( 2, 3 ))/DET
C
      ATWAIN( 1, 2 )    =
     1(ATWA( 1, 3 ) * ATWA( 2, 3 ) - ATWA( 1, 2) * ATWA( 3, 3 ))/DET
C
      ATWAIN( 1, 3 )    =
     1(ATWA( 1, 2 ) * ATWA( 2, 3 ) - ATWA( 1, 3 ) * ATWA( 2, 2 ))/ DET
C
      ATWAIN( 2, 2 ) =
     1(ATWA( 1, 1 ) * ATWA( 3, 3 ) - ATWA( 1, 3 ) * ATWA( 1, 3 ))/DET
```

```
C
      ATWAIN( 2, 3 )    =
     1(ATWA( 1, 3 ) * ATWA( 1, 2 ) - ATWA( 1, 1 ) * ATWA( 2, 3 ))/DET
C
      ATWAIN( 3, 3)     =
     1(ATWA( 1, 1 ) * ATWA( 2, 2 ) - ATWA( 1, 2 ) * ATWA( 1, 2 ))/DET
C
      ATWAIN( 2, 1 )    = ATWAIN( 1, 2 )
      ATWAIN( 3, 1 )    = ATWAIN( 1, 3 )
      ATWAIN( 3, 2 )    = ATWAIN( 2, 3 )
C
      IF( DEBU10 )WRITE( 6, 10 )((ATWAIN( J, K),K=1,3),J=1,3)
 10   FORMAT( 1H ,' IN ATWAV ATWAIN = ', 3F20.9 )
      RETURN
      END
```

```
      SUBROUTINE NEWOPK
C234567891111111111222222222233333333334444444444555555555566666666677
C
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z)
      INTEGER ONEORO
      DOUBLE PRECISION KAPPA
      LOGICAL ABFLAG, DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6
      LOGICAL DEBUG7, DEBUG8, DEBUG9, DEBU10,         DEBU11, DEBU12
      DIMENSION RIGHTA( 30 ), DECLIN( 30 ), X( 30 ), Y( 30 )
      DIMENSION    XWEIGH( 30 ), YWEIGH( 30 ), XRESID( 30 )
      DIMENSION    YRESID( 30 ), ABXRES( 30 ), ABYRES( 30 )
      DIMENSION    ORMTRX( 3, 3 ), ATWRES( 3 ), ATWA( 3, 3 )
      DIMENSION    ATWAIN( 3, 3 ), TOTRES( 30 )
C**********************************************************************
      COMMON /GIVEN/ F, X, Y , RIGHTA, DECLIN, N
      COMMON /ORIENT/ OMEGA, PHI, KAPPA, ORMTRX
      COMMON /RESNWT/XRESID, YRESID,   XWEIGH, YWEIGH ,
     1               SUMSQX, SUMSQY, SUMABX, SUMABY, ABXRES, ABYRES
      COMMON/UVW/ U, V, W
      COMMON/PARDRV/ PF1OME, PF1PHI, PF1KAP, PF2OME, PF2PHI, PF2KAP
      COMMON/MATVEC/ATWA, ATWAIN, ATWRES
      COMMON/DEBUGS/DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6,
     1           DEBUG7, DEBUG8, DEBUG9,DEBU10,DEBU11,DEBU12
C
C234567891111111111222222222233333333334444444444555555555566666666677
C
C23456789111111111122222222223333333333444444444455555555556666666667
C
C     SOLVE FOR NEW OMEGA, PHI, AND KAPPA
C
      DO 40 II = 1, 3
         OMEGA     = OMEGA   - ATWAIN( 1, II ) * ATWRES( II )
         PHI       = PHI     - ATWAIN( 2, II ) * ATWRES( II )
         KAPPA     = KAPPA   - ATWAIN( 3, II ) * ATWRES( II )
40    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DPRDMS( ANGLE, IDEG, MIN, SEC )
C     CONVERTS A DOUBLE PRECISION RADIAN ANGLE...ANGLE
C     TO DEGREES,MINUTES,SECONDS
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z )
      INTEGER IDEG, MIN
C
      DEGREE  = ANGLE * 57.2957795D0
      IDEG    = DEGREE
      IF ( ANGLE .LT. 0.0D0 ) DEGREE = - DEGREE
      DEG     = IDEG
      IF ( ANGLE .LT. 0.0D0 ) DEG = - DEG
      RMIN    = ( DEGREE - DEG ) * 2.90888209D-04
      MIN     = RMIN
      SEC     = ( RMIN - RRMIN ) * 4.8481368D-06
      RETURN
      END
```

```
      SUBROUTINE PRTMAT
C  PRINTS OUT ORIENTATION MATRIX
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z )
      DOUBLE PRECISION  KAPPA
      DIMENSION ORMTRX( 3, 3 )
      OMEGA    = -0.610865238D0
      PHI      = 0.5235987750D0
      KAPPA    =  0.174532925D0
C
      CALL JNKMAT( OMEGA, PHI, KAPPA, ORMTRX )
C
      WRITE( 6, 100 )((ORMTRX( II, JJ ), JJ = 1, 3), II= 1, 3)
100   FORMAT( 3( 10X, F12.9 ))
      STOP
      END
```

Appendix C

Computer Program for the Modified Church Method for Three or More Images

Appendix C contains the FORTRAN computer program CHURCH which uses the modified Church method to perform a stellar resection.

```
      PROGRAM CHURCH
C PERFORMS STELLAR RESECTION USING CHURCH METHOD
C SAMPLE DATA SET OF FOUR STARS IS BUILT IN
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DOUBLE PRECISION M(3,3) , C(3,3), CV(3,3), F(3,3)
      DOUBLE PRECISION X(4) , Y(4) , U(4) , V(4) , W(4)
      DOUBLE PRECISION                  P(4) , Q(4) , R(4)
      FL      = 76.0D0
      FLSQ    =5776.0D0
      DEGRAD  = 57.2957795131D0
      X(1)    =  24.100D0
      X(2)    = -15.200D0
      X(3)    = -23.700D0
      X(4)    =  19.800D0
      Y(1)    =  23.800D0
      Y(2)    =  15.300D0
      Y(3)    = -24.000D0
      Y(4)    = -10.000D0
C
      DO 1 I = 1,4
         DIST = DSQRT( X(I) * X(I) + Y(I) * Y(I) + FLSQ )
         P(I)      = X(I) / DIST
         Q(I)      = Y(I) / DIST
         R(I)      = -FL / DIST
 1    CONTINUE
C
      U(1)    =  0.286684559D0
      V(1)    = -0.022212524D0
      W(1)    =  0.957767491D0
      U(2)    =  0.055018946D0
      V(2)    =  0.408177992D0
      W(2)    =  0.911242911D0
      U(3)    =  0.300233062D0
      V(3)    =  0.721673709D0
      W(3)    =  0.623736456D0
      U(4)    =  0.560369112D0
      V(4)    =  0.246933809D0
      W(4)    =  0.790575836D0
C
      DO 2 I = 1, 3
         DO 2 J = 1,3
            C(I,J) = 0.0D0
            F(I,J) = 0.0D0
            M(I,J) = 0.0D0
            CV(I,J) = 0.0D0
 2    CONTINUE
C
```

```
      DO 3 I = 1,4
         C(1,1)    = C(1,1) + U(I) * U(I)
         C(1,2)    = C(1,2) + U(I) * V(I)
         C(1,3)    = C(1,3) + U(I) * W(I)
C
         C(2,1)    = C(1,2)
         C(2,2)    = C(2,2) + V(I) * V(I)
         C(2,3)    = C(2,3) + V(I) * W(I)
C
         C(3,1)    = C(1,3)
         C(3,2)    = C(2,3)
         C(3,3)    = C(3,3) + W(I) * W(I)
C
         F(1,1)    = F(1,1) + P(I) * U(I)
         F(1,2)    = F(1,2) + Q(I) * U(I)
         F(1,3)    = F(1,3) + R(I) * U(I)
C
         F(2,1)    = F(2,1) + P(I) * V(I)
         F(2,2)    = F(2,2) + Q(I) * V(I)
         F(2,3)    = F(2,3) + R(I) * V(I)
C
         F(3,1)    = F(3,1) + P(I) * W(I)
         F(3,2)    = F(3,2) + Q(I) * W(I)
         F(3,3)    = F(3,3) + R(I) * W(I)
3     CONTINUE
C
      CALL INV3X3(C, CV)
C
      DO 4 I = 1, 3
         DO 4 J = 1,3
              DO 4 K = 1, 3
                   M(I,J)    = M(I,J) + CV(I,K) * F(K,J)
4     CONTINUE
C NOTE... M IS ACTUALLY M TRANSPOSE.....................
C
      DO 5 I = 1, 3
         DO 5 J = 1, 3
              WRITE ( 6 ,100 ) J, I , M(I,J)
5     CONTINUE
 100  FORMAT( 1H0, ' M(', I3,',',I3,' )= ', F20.8 )
      ALPHA = DATAN2( -M(2,3), -M(1,3) ) * DEGRAD
      DELTA = DASIN( -M(3,3) ) * DEGRAD
      SWING = DATAN2( M(3,1), M(3,2) ) * DEGRAD
      WRITE( 6, 222 ) ALPHA , DELTA , SWING
 222  FORMAT(1H0, 'ALPHA = ', F20.10,' DELTA=',F20.10,' SWING=',F20.10)
      END
```

```
      SUBROUTINE INV3X3(R,A)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C COMPUTES THE INVERSE OF THE 3X3 MATRIX R AND RETURNS IT
C IN THE MATRIX  A
      DIMENSION R( 3, 3 ) , O( 3, 3 ), A(3,3)
      O(1,1) = R(2,2) * R(3,3) - R(3,2) * R(2,3)
      O(1,2) =-R(2,1) * R(3,3) + R(3,1) * R(2,3)
      O(1,3) = R(2,1) * R(3,2) - R(3,1) * R(2,2)
      O(2,1) =-R(1,2) * R(3,3) + R(3,2) * R(1,3)
      O(2,2) = R(1,1) * R(3,3) - R(3,1) * R(1,3)
      O(2,3) =-R(1,1) * R(3,2) + R(3,1) * R(1,2)
      O(3,1) = R(1,2) * R(2,3) - R(2,2) * R(1,3)
      O(3,2) =-R(1,1) * R(2,3) + R(2,1) * R(1,3)
      O(3,3) = R(1,1) * R(2,2) - R(2,1) * R(1,2)
C
      DET    = R(1,1) * O(1,1) + R(1,2) * O(1,2)
     1         + R(1,3) * O(1,3)
C
      DO 1 I = 1,3
         DO 2 J = 1,3
            O(I,J) = O(I,J) / DET
    2       CONTINUE
    1    CONTINUE
C
C  TRANSPOSE
      DO  4  I  = 1, 3
         DO  3  J  = 1, 3
            A(I,J)  = O(J,I)
    3       CONTINUE
    4    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE TRUOPK( OMEGA, PHI, KAPPA, M12, M13, M22, M23,
     1  TRUOME, TRUPHI, TRUKAP )
C     THIS ROUTINE FINDS THE REAL OR TRUE VALUES OF OMEGA, PHI, AND
C     KAPPA USING THE VALUES OMEGA = ARCTAN( -M32/M33 )
C     PHI = ARCSIN(M31),      KAPPA = ARCTAN(-M21/M11 ), AND
C     THE M12, M13, M22, AND M23 ELEMENTS OF THE ORIENTATION
C     MATRIX. REFERENCE: MANUAL OF PHOTOGRAMMETRY 1980
C     INPUT:OMEGA,PHI,KAPPA,M12,M13,M22,M23...OUTPUT TRUE OPK
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DOUBLE PRECISION KAPPA, KA(2)
      DIMENSION OM(2), PH(2)
      SMALL   = 0.00001D0
C     FIND THE SINE AND COSINE FOR EACH OF THE THREE ANGLES
C
      SINO    = DSIN( OMEGA )
      COSO    = DCOS( OMEGA )
      SINP    = DSIN( PHI )
      COSP    = DCOS( PHI )
      SINK    = DSIN( KAPPA )
      COSK    = DCOS( KAPPA )
C=====================================================================
C     FIND TWO POTENTIAL VALUES FOR EACH OF THE THREE ANGLES
C
      OM(1)   = DATAN2( SINO, COSO )
      OM(2)   = DATAN2(-SINO,-COSO )
      PH(1)   = DATAN2( SINP, COSP )
      PH(2)   = DATAN2( SINP,-COSP )
      KA(1)   = DATAN2( SINK, COSK )
      KA(2)   = DATAN2(-SINK,-COSK )
C=====================================================================
C     LOOP THROUGH EACH OF THE EIGHT COMBINATIONS OF POTENTIAL
C     VALUES, EXITING WHEN A SUITABLE SET OF ANGLES HAS BEEN
C     FOUND
C
      DO 40 I = 1, 2
         DO 30 J = 1, 2
              DO 20 K = 1, 2
                   SINO = DSIN( OM(I) )
                   COSO = DCOS( OM(I) )
                   SINP = DSIN( PH(J) )
                   COSP = DCOS( PH(J) )
                   SINK = DSIN( KA(K) )
                   COSK = DCOS( KA(K) )
C                  FORM THE TEST VALUES OF THE MATRIX ELEMENTS
C                  AND COMPARE WITH THE ACTUAL ELEMENTS
                   T12 = SINO*SINP*COSK + COSO*SINK
                   T13 =-COSO*SINP*COSK + SINO*SINK
```

```
                    T22 =-SINO*SINP*SINK + COSO*COSK
                    T23 = COSO*SINP*SINK + SINO*COSK
C
                    IF(((T12-M12)*(TI2-M12)/M12/M12) .GT. SMALL)
     1                   GO TO 10
                    IF(((T13-M13)*(T13-M13)/M13/M13) .GT. SMALL)
     1                   GO TO 10
                    IF(((T22-M22)*(T12-M22)/M22/M22) .GT. SMALL )
     1                   GO TO 10
                    IF(((T23-M23)*(T23-M23)/M23/M23) .GT. SMALL )
     1                   GO TO 10
C
C                   IF THE TEST ANGLES PASS ALL COMPARISONS THEN THE
C                   THE TRUE VALUES HAVE BEEN FOUND
                    TRUOME = OM(I)
                    TRUPHI = PH(J)
                    TRUKAP = KA(K)
                    GO TO 50
 10                 CONTINUE
 20               CONTINUE
 30          CONTINUE
 40       CONTINUE
 50       CONTINUE
          RETURN
          END
```

Appendix D

Computer Program for the Modified Church Method with Two Images

Appendix D contains the FORTRAN computer program TESTU which performs a stellar resection using two star images by generating the image coordinates for a fictitious third star and then using the modified Church method.

```
      PROGRAM TESTU
      IMPLICIT DOUBLE PRECISION   (A-H, O-Z)
      DOUBLE PRECISION M(3,3) , C(3,3), CV(3,3), F(3,3)
      DOUBLE PRECISION X(4) , Y(4) , U(4) , V(4) , W(4)
      DOUBLE PRECISION  OMAT(3,3),   P(4) , Q(4) , R(4)
      FL       = 76.0D0
      FLSQ     =5776.0D0
      DEGRAD   = 57.2957795131D0
      X(1)     =  24.000D0
      X(2)     = -15.000D0
      X(3)     = -24.000D0
      X(4)     =  20.000D0
      Y(1)     =  24.000D0
      Y(2)     =  15.000D0
      Y(3)     = -24.000D0
      Y(4)     = -10.000D0
C
      DO 1 I = 1,4
         DIST = DSQRT( X(I) * X(I) + Y(I) * Y(I) + FLSQ )
         P(I)       = X(I) / DIST
         Q(I)       = Y(I) / DIST
         R(I)       = -FL / DIST
 1    CONTINUE
C
      U(1)     =  0.286684559D0
      V(1)     = -0.022212524D0
      W(1)     =  0.957767491D0
      U(2)     =  0.055018946D0
      V(2)     =  0.408177992D0
      W(2)     =  0.911242911D0
      U(3)     =  0.300233062D0
      V(3)     =  0.721673709D0
      W(3)     =  0.623736456D0
      U(4)     =  0.560369112D0
      V(4)     =  0.246933809D0
      W(4)     =  0.790575836D0
         CALL TUSTAR (P(1),Q(1),R(1),P(2),Q(2),R(2),U(1),V(1),W(1),
     1          U(2),V(2),W(2),
     1          OMAT,OMEGA,PHI,KAPPA,ALPHA,DELTA,SWING,INTERS,DISCRE)
      CONTINUE
         END
```

```
      SUBROUTINE TUSTAR(P1,Q1,R1,P2,Q2,R2,U1,V1,W1,U2,V2,W2,OMAT
     1                  ,OMEGA,PHI,KAPPA,ALPHA,DELTA,SWING,
     2                   INTERS,DISCRE )
C  PERFORMS  STELLAR RESECTION USING CHURCH METHOD AND A FICTITIOUS
C  STAR. INPUT DIRECTION COSINES P-W FOR TWO STARS. OUTPUT
C  ORIENTATION MATRIX,INTERSTAR ANGLE FOR THE TWO STARS AND THE
C  DISCREPANCY BETWEEN THE PHOTO AND CELESTIAL INTERSTAR ANGLES
      IMPLICIT DOUBLE PRECISION (A-H, O-Z )
      DOUBLE PRECISION KAPPA, INTERS
      DIMENSION    OMAT(3,3), PQR(3,3), UVW(3,3), UVWINV(3,3)
      DEGRAD  = 57.2957795131D0
C
      P3      = Q1*R2 - Q2*R1
      Q3 = P2*R1 - P1*R2
      R3      = P1*Q2 - P2*Q1
C
      U3      = V1*W2 - V2*W1
      V3      = U2*W1 - U1*W2
      W3      = U1*V2 - U2*V1
C
      PQR(1,1)     = P1
      PQR(1,2)     = P2
      PQR(1,3)     = P3
      PQR(2,1)     = Q1
      PQR(2,2)     = Q2
      PQR(2,3)     = Q3
      PQR(3,1)     = R1
      PQR(3,2)     = R2
      PQR(3,3)     = R3
C
      UVW(1,1)     = U1
      UVW(1,2)     = U2
      UVW(1,3)     = U3
      UVW(2,1)     = V1
      UVW(2,2)     = V2
      UVW(2,3)     = V3
      UVW(3,1)     = W1
      UVW(3,2)     = W2
      UVW(3,3)     = W3
C
      CALL INV3X3( UVW, UVWINV )
C
      DO 10 I = 1, 3
         DO 20 J = 1, 3
              OMAT(I,J) = 0.0D0
              DO 30 K = 1, 3
                   OMAT(I,J) = OMAT(I,J) + PQR(I,K) * UVWINV(K,J)
```

```
30            CONTINUE
20         CONTINUE
10      CONTINUE
C       COMPUTE THE INTERSTAR ANGLE
        ANGPHO  =    DACOS(P1*P2 + Q1*Q2 + R1*R2)
        INTERS  = DACOS(U1*U2 + V1*V2 + W1*W2)
        DISCRE  = 206264.8062471D0 * (INTERS - ANGPHO)
C
        OMEGA   = DATAN2(-OMAT(3,2), OMAT(3,3)) * DEGRAD
        COSOME  = DCOS( OMEGA )
        COSPHI  = OMAT(3,3) / COSOME
        IF( COSPHI .GT. 1.0D0 ) COSPHI = 1.0D0
        PHI     = DATAN2( OMAT(3,1), COSPHI) * DEGRAD
        KAPPA   = DATAN2( -OMAT(2,1), OMAT(1,1)) * DEGRAD
C
        ALPHA   = DATAN2( -OMAT(3,2), -OMAT(3,1)) * DEGRAD
        COSALP  = DCOS( ALPHA )
        COSDEL  = -OMAT(3,1) / COSALP
        IF( COSDEL .GT. 1.0D0 ) COSDEL = 1.0D0
        DELTA   = DATAN2( -OMAT(3,3), COSDEL ) * DEGRAD
        SWING   = DATAN2( OMAT(1,3), OMAT(2,3) ) * DEGRAD
C
        WRITE(6,100) P1,P2,P3
        WRITE(6,110)Q1,Q2,Q3
        WRITE(6,120)R1,R2,R3
        WRITE(6,130)U1,U2,U3
        WRITE(6,140)V1,V2,V3
        WRITE(6,150)W1,W2,W3
        WRITE(6,160)(( OMAT(I,J), J=1,3), I=1,3)
        WRITE(6,170) OMEGA, PHI, KAPPA
        WRITE(6,180) ALPHA, DELTA, SWING
C
100     FORMAT(1H0, 'P1',F12.9,3X,'P2',F12.9,3X,'P3',F12.9)
110     FORMAT(1H0, 'Q1',F12.9,3X,'Q2',F12.9,3X,'Q3',F12.9)
120     FORMAT(1H0, 'R1',F12.9,3X,'R2',F12.9,3X,'R3',F12.9)
130     FORMAT(1H0, 'U1',F12.9,3X,'U2',F12.9,3X,'U3',F12.9)
140     FORMAT(1H0, 'V1',F12.9,3X,'V2',F12.9,3X,'V3',F12.9)
150     FORMAT(1H0, 'W1',F12.9,3X,'W2',F12.9,3X,'W3',F12.9)
160     FORMAT(1H0, 3(3X, F12.9 )  )
170     FORMAT(1H0, ' OMEGA = ',F9.4,'  PHI = ',F9.4,'  KAPPA = ',F9.4)
180     FORMAT(1H0, ' ALPHA = ',F9.4,' DELTA= ',F9.4,'  SWING = ',F9.4)
        RETURN
        END
```

```
      SUBROUTINE INV3X3(R,A)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C COMPUTES THE INVERSE OF THE 3X3 MATRIX R AND RETURNS IT
C IN THE MATRIX    A
      DIMENSION R( 3, 3 ) , O( 3, 3 ), A(3,3)
      O(1,1) = R(2,2) * R(3,3) - R(3,2) * R(2,3)
      O(1,2) =-R(2,1) * R(3,3) + R(3,1) * R(2,3)
      O(1,3) = R(2,1) * R(3,2) - R(3,1) * R(2,2)
      O(2,1) =-R(1,2) * R(3,3) + R(3,2) * R(1,3)
      O(2,2) = R(1,1) * R(3,3) - R(3,1) * R(1,3)
      O(2,3) =-R(1,1) * R(3,2) + R(3,1) * R(1,2)
      O(3,1) = R(1,2) * R(2,3) - R(2,2) * R(1,3)
      O(3,2) =-R(1,1) * R(2,3) + R(2,1) * R(1,3)
      O(3,3) = R(1,1) * R(2,2) - R(2,1) * R(1,2)
C
      DET    = R(1,1) * O(1,1) + R(1,2) * O(1,2)
     1         + R(1,3) * O(1,3)
C
      DO 1 I = 1,3
         DO 2 J = 1,3
            O(I,J) = O(I,J) / DET
    2       CONTINUE
    1    CONTINUE
C
C  TRANSPOSE
      DO  4  I  = 1, 3
         DO  3  J  = 1, 3
            A(I,J)  = O(J,I)
    3       CONTINUE
    4    CONTINUE
      RETURN
      END
```

Appendix E

Computer Program for the Single Rotation Method

Appendix E contains the FORTRAN computer program TWOSTAR which performs a stellar resection using the single rotation method.

```
C     PROGRAM TWOSTAR
C  PERFORMS STELLAR RESECTION USING THE SINGLE ROTATION METHOD
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z )
      LOGICAL       DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6
      DIMENSION  X( 2 ), Y( 2 ), RA( 2 ), DECL( 2 )
      COMMON /PARAMS/ X, Y, F, RA, DECL,
     1              P1, Q1, R1, U1, V1, W1, P2, Q2, R2, U2, V2, W2,
     2              A1, B1, C1, A2, B2, C2, XR, YR, ZR, ROTANG,
     3              COSXX, COSYY, COSZZ,
     4              DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6
C
      CALL XYRDIN
1     CONTINUE
      CALL SUBCRS
2     CONTINUE
      CALL PIKFAR
3     CONTINUE
      CALL ROTATE( 0.D0, 0.D0, 1.D0, XR, YR, ZR, ROTANG,
     1             COSXZP, COSYZP, COSZZP )
4     CONTINUE
      WRITE( 6, 100 ) COSXZP, COSYZP,  COSZZP
100   FORMAT( 5X, ' COSXZP = ', F12.9, 5X,
     1 'COSZYP = ', F12.9, 5X , 'COSZZP = ', F12.9 )
      STOP
      END
```

```
      SUBROUTINE XYRDIN
C READ X Y RIGHT ASCENSION, DECLINATION FOR TWO STAR SOLUTION
C AND COMPUTE CORRESPONDING DIRECTION COSINES IN THE CAMERA
C AND INERTIAL SYSTEMS.
C
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z )
      LOGICAL       DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6
      DIMENSION  X( 2 ), Y( 2 ), RA( 2 ), DECL( 2 )
      COMMON /PARAMS/ X, Y, F, RA, DECL,
     1              P1, Q1, R1, U1, V1, W1, P2, Q2, R2, U2, V2, W2,
     2              A1, B1, C1, A2, B2, C2, XR, YR, ZR, ROTANG,
     3              COSXX, COSYY, COSZZ,
     4              DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6
C
      RADDEG  = 57.295779513D0
C
      DEBUG1  = .TRUE.
      DEBUG2  = .TRUE.
      DEBUG3  = .TRUE.
      DEBUG4  = .TRUE.
      DEBUG5  = .TRUE.
      DEBUG6  = .TRUE.
C
C     READ X, Y, RIGHT ASCENSION, DECLINATION, AND FOCAL LENGTH
C     FOR TWO STARS.
C
      GO TO 888
111   CONTINUE
      DO 10 I = 1, 2
         WRITE( 6, 600 ) I
600      FORMAT( ' ENTER X COORDINATE FOR STAR ', I4, ' F14.6 ' )
         READ( 5, 500 ) X( I )
         WRITE( 6, 610 ) I
610      FORMAT( ' ENTER Y COORDINATE FOR STAR', I4 ,' F14.6 ' )
         READ( 5, 500 ) Y( I )
500      FORMAT( F14.6 )
         WRITE( 6, 620 ) I
620      FORMAT( ' ENTER RIGHT ASCENSION FOR STAR' , I4,
     1             ' IN DECIMAL DEGREES ')
         READ( 5, 510 ) RA( I )
510      FORMAT( F16.11 )
         WRITE( 6, 630 ) I
630      FORMAT( ' ENTER DECLINATION FOR STAR ', I4,
     1             ' IN DECIMAL DEGREES ' )
         READ( 5, 510 ) DECL( I )
10    CONTINUE
      WRITE( 6, 640 )
```

```
640   FORMAT( ' ENTER FOCAL LENGTH ( PRINCIPAL DISTANCE ) F14.6 ' )
      READ( 5, 500 ) F
C
888   X( 1 )        = -9.5992D0
      Y( 1 )        = -0.8069D0
      X( 2 )        =  2.5188D0
      Y( 2 )        =  8.4833D0
      RA( 1 )       = 35.0000D0
      DECL(1 )      = 35.0000D0
      RA( 2 )       = 35.0000D0
      DECL( 2 )     = 55.0000D0
      F             = 42.9800D0
C
C     CONVERT RIGHT ASCENSION AND DECLINATION ANGLES TO RADIANS
C
      RA( 1 )       = RA( 1 ) / RADDEG
      DECL( 1 )     = DECL( 1 ) / RADDEG
      RA( 2 )       = RA( 2 ) / RADDEG
      DECL( 2 )     = DECL( 2 )/ RADDEG
C
C     COMPUTE DIRECTION COSINES
C
      DIST1   = DSQRT( X(1) * X(1) + Y(1) * Y(1) + F * F )
      P1      = +X( 1 ) / DIST1
      Q1      = +Y( 1 ) / DIST1
      R1      =       F / DIST1
C
      U1      = DCOS( DECL( 1 )) * DCOS( RA( 1 ))
      V1      = DCOS( DECL( 1 )) * DSIN( RA( 1 ))
      W1      = DSIN( DECL( 1 ))
C
      DIST2   = DSQRT( X(2) * X(2) + Y(2) * Y(2) + F * F )
      P2      = +X( 2 ) / DIST2
      Q2      = +Y( 2 ) / DIST2
      R2      =       F / DIST2
C
      U2      = DCOS( DECL( 2 )) * DCOS( RA( 2 ))
      V2      = DCOS( DECL( 2 )) * DSIN( RA( 2 ))
      W2      = DSIN( DECL( 2 ))
      GO TO 999
777   U1      =-0.969846D0
      V1      = 0.171010D0
      W1      = 0.173648D0
      P1      = -0.852868D0
      Q1      = -0.492403D0
      R1      = 0.173648D0
      U2      = -0.969846D0
```

```
      V2       = -0.171010D0
      W2       = -0.173648D0
      P2       = -0.852868D0
      Q2       = -0.150383D0
      R2       =  0.5D0
999   IF( DEBUG1 ) WRITE ( 6, 101 )
101   FORMAT(' COMING FROM XYRDIN WITH U1,V1,W1,U2,V2,W2,',
     1 'P1,Q1,R1,P2,Q2,R2 = ')
      IF( DEBUG1 ) WRITE( 6,222 )U1,V1,W1,U2,V2,W2,P1,Q1,R1,
     1                           P2,Q2,R2
222   FORMAT ( 10X, F15.9, 5X, F15.9, 5X, F15.9 )
      RETURN
      END
```

```
      SUBROUTINE CROSS( A1,B1,C1,A2,B2,C2,A3,B3,C3  )
C     COMPUTES THE CROSS PRODUCT OF VECTORS 1 AND 2 OUTPUTS 3
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z )
C
      A3        = B1 * C2 - B2 * C1
      B3        = C1 * A2 - C2 * A1
      C3        = A1 * B2 - A2 * B1
C
      RETURN
      END
```

```
      SUBROUTINE AXIANG( A, B, C, D, E, F, X, Y, Z, ROTANG )
C
C     GIVEN THAT A ROTATION ABOUT THE ' AXIS ' ( X, Y, Z ) TRANSFORMS
C     THE POINT ( A, B, C ) INTO THE POINT  ( D, E, F ) THIS ROUTINE
C     COMNPUTES THE ANGLE OF ROTATION ABOUT THE AXIS.
C     SINCE THE AXIS CAN HAVE TWO DIFFERENT SENSES OR DIRECTIONS
C     ONE BEING THE NEGATIVE OF THE OTHER AND SINCE THE DIRECTION
C     COSINES OF THE AXIS ARE USED LATER IN SOME CONDITION EQUATIONS
C     THE AXIS IS RECOMPUTED AND UNITIZED.
C
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z )
C
C     FIRST FIND THE COSINE OF THETA , THE ANGLE BETWEEN THE POSITION
C     VECTOR ( A, B, C ) AND THE ROTATION AXIS ( X, Y, Z )
C
      ONEMAG  = DSQRT( A * A + B * B + C * C )
      AXIMAG  = DSQRT( X * X + Y * Y + Z * Z )
C
      CTHETA  = ( A * X + B * Y + C * Z ) / ( ONEMAG * AXIMAG )
C
C     FIND THE PROJECTION OF ( A, B, C ) ON THE ROTATION AXIS ( X, Y, Z
C
      CX      = CTHETA * X * ONEMAG
      CY      = CTHETA * Y * ONEMAG
      CZ      = CTHETA * Z * ONEMAG
C
C     FORM THE VECTORS FROM ( CX, CY, CZ ) TO ( A, B, C ) AND TO
C     ( D, E, F )
C
      AR      = A - CX
      BR      = B - CY
      CR      = C - CZ
      DR      = D - CX
      ER      = E - CY
      FR      = F - CZ
C
C     FIND THE ROTATION ANGLE ROTANG BY DOTTING ( AR, BR, CR ) WITH
C     ( DR, ER, FR ) AND THEN DIVIDING THE DOT PRODUCT BY THE
C     PRODUCT OF THE MAGNITUDES OF THE TWO VECTORS.
C     NOTE: THE MAGNITUDES OF THE TWO VECTORS SHOULD BE NEARLY EQUAL
C     AN ERROR CHECK COULD BE PERFORMED AT THIS POINT FOR ZERO OR
C     UNEQUAL MAGNITUDES.
C
      VMAGSQ  = ( AR * AR + BR * BR + CR * CR )
C
      ROTANG  = DARCOS( ( AR * DR + BR * ER + CR * FR ) / VMAGSQ )
C
```

```
C     FIND OUT IF THE ROTATION ANGLE  ROTANG IS POSITIVE OR NEGATIVE
C     BY COMPARING THE  DIRECTION OR SENSE OF THE ROTATION AXIS WITH
C     THAT OF THE CROSS PRODUCT ( AR,BR,CR ) X ( DR,ER,FR ).
C     IF THEY ARE THE SAME, THEN ROTANG IS POSITIVE OTHERWISE
C     ROTANG IS NEGATIVE.
C
      CALL CROSS( AR, BR, CR, DR, ER, FR, XX,YY,ZZ )
C
      IF((DABS( X + XX ) + DABS( Y +YY ) +DABS( Z + ZZ )) .LT.
     1   (DABS( X ) + DABS( Y ) + DABS( Z )) ) ROTANG = -ROTANG
      RETURN
      END
```

```
      SUBROUTINE SUBCRS
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z )
      LOGICAL       DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6
      DIMENSION  X( 2 ), Y( 2 ), RA( 2 ), DECL( 2 )
      COMMON /PARAMS/ X, Y, F, RA, DECL,
     1              P1, Q1, R1, U1, V1, W1, P2, Q2, R2, U2, V2, W2,
     2              A1, B1, C1, A2, B2, C2, XR, YR, ZR, ROTANG,
     3              COSXX, COSYY, COSZZ,
     4              DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6
C
C     FIND THE DIRECTION COSINE DIFFERENCESμ THAT IS, THE
C     CAMERA COSINES MINUS THE INERITAL COSINES
C
      A1       = P1 - U1
      B1       = Q1 - V1
      C1       = R1 - W1
      A2       = P2 - U2
      B2       = Q2 - V2
      C2       = R2 - W2
C
C     TAKE THE CROSS PRODUCT OF THE TWO DIFFERENCE VECTORS TO FIND
C     THE ROTATION AXIS
C
      XR       = B1 * C2 - B2 * C1
      YR       = C1 * A2 - C2 * A1
      ZR       = A1 * B2 - A2 * B1
C
C     UNITIZE THE ROTATION AXIS
C
      DISTR    = DSQRT( XR * XR + YR * YR + ZR * ZR )
      XR       = XR / DISTR
      YR       = YR / DISTR
      ZR       = ZR / DISTR
C
      IF( DEBUG2 ) WRITE( 6, 100 ) A1, B1, C1
100   FORMAT( 10X,'A1 = ',F12.9,5X,'B1 = ', F12.9,5X,'C1 = ',F12.9)
      IF( DEBUG2 ) WRITE( 6,101 )A2, B2, C2
101   FORMAT(10X, 'A2 = ',F12.9,5X,'B2 = ',F12.9,5X,'C2 = ',F12.9 )
      IF( DEBUG2 ) WRITE( 6, 102 )XR, YR, ZR
102   FORMAT(10X, 'XR = ',F12.9,5X,'YR = ', F12.9,5X,'ZR = ',F12.9 )
      RETURN
      END
```

```
      SUBROUTINE PIKFAR
C     THIS ROUTINE PICKS THE STAR WHICH IS FURTHER FROM THE
C     ROTATION AXIS AND CALLS AXIANG TO RECOMPUTE THE AXIS
C     AND COMPUTE THE ANGLE THROUGH WHICH THE STARS ARE ROTATED
C     IN GOING FROM THE INERTIAL TO THE CAMERA SYSTEM.
C
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z )
      LOGICAL       DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6
      DIMENSION  X( 2 ), Y( 2 ), RA( 2 ), DECL( 2 )
      COMMON /PARAMS/ X, Y, F, RA, DECL,
     1              P1, Q1, R1, U1, V1, W1, P2, Q2, R2, U2, V2, W2,
     2              A1, B1, C1, A2, B2, C2, XR, YR, ZR, ROTANG,
     3              COSXX, COSYY, COSZZ,
     4              DEBUG1, DEBUG2, DEBUG3, DEBUG4, DEBUG5, DEBUG6
C
C     PICK THE STAR WHICH IS FARTHEST FROM THE ROTATION AXIS
C
      DIST1   = DISP2A( U1, V1, W1, XR, YR, ZR )
      DIST2   = DISP2A( U2, V2, W2, XR, YR, ZR )
C
C     REDEFINE THE ROTATION AXIS, PICKING THE STAR WHICH IS
C     FARTHEST FROM THE ROTATION AXIS FOR BEST GEOMETRY.
C
      IF( DIST1 .GT. DIST2 )
     1 CALL AXIANG( U1, V1, W1, P1, Q1, R1, XR, YR, ZR, ROTANG )
C
      IF( DIST2 .GE. DIST1 )
     1 CALL AXIANG( U2, V2, W2, P2, Q2, R2, XR, YR, ZR, ROTANG )
C
      ROTDEG  = ROTANG * 57.295779513D0
      AXISDE  = DARSIN( ZR ) * 57.295779513D0
      AXISRA  = DATAN2( YR, XR ) * 57.295779513D0
      IF( DEBUG3 ) WRITE ( 6, 215 )AXISRA, AXISDE
215   FORMAT( '   IN PIKFAR AXISRA = ', F14.6,5X,'AXISDE = ',F14.6)
      IF ( DEBUG3 ) WRITE( 6, 100 ) XR, YR, ZR, ROTDEG
100   FORMAT( ' IN PIKFAR XR =' , F12.9, 5X, 'YR = ', F12.9,
     1  5X, ' ZR = ', F12.9, 5X, ' ROTANG = ', F14.9 )
      RETURN
      END
```

```
      SUBROUTINE ROTATE( A1, B1, C1, X, Y, Z, ROTANG, A2, B2, C2 )
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z )
      LOGICAL DEBUG
C     INPUT   : A1, B1, C1, X, Y, Z, ROTANG
C     OUTPUT  : A2, B2, C2
C
C     THIS ROUTINE ROTATES THE POINT ( A1, B1, C1 ) THROUGH AN ANGLE
C     ROTANG ABOUT AN AXIS ( X, Y, Z ) INTO THE POINT ( A2, B2, C2 ).
C     A PERPENDICULAR FROM THE POINT ( A1, B1, C1 ) TO THE ROTATION AX
C     WILL  INTERSECT THE AXIS AT POINT ( AX, BY, CZ ). THE CROSS
C     PRODUCT
C     ( A1 - AX, B1 -BY, C1 - CZ ) X ( A2 -AX, B2 - BY, C2 -CZ )
C     WILL BE A VECTOR POINTING IN THE SAME DIRECTION AS ( X, Y, Z )
C
C      FIND THE ANGLE BETWEEN THE AXIS ( X, Y, Z ) AND THE POSITION
C      VECTOR ( A1, B1, C1 )
C
       AXIMAG      = DSQRT( X * X + Y * Y + Z * Z )
       POSMAG      = DSQRT( A1 * A1 + B1 * B1 + C1 * C1 )
C
      COSTHE   = ( A1 *X + B1 * Y + C1 * Z ) /  ( AXIMAG * POSMAG )
C     CHECK FOR ONE OR MINUS ONE
C
C     UNITIZE THE ROTATION AXIS
      UX      = X / AXIMAG
      UY      = Y / AXIMAG
      UZ      = Z / AXIMAG
C
      AX      = POSMAG * COSTHE * UX
      BY      = POSMAG * COSTHE * UY
      CZ      = POSMAG * COSTHE * UZ
C
C     LET
      A       = A1 - AX
      B       = B1 - BY
      C       = C1 - CZ
C     THESE ARE ' KNOWN '
C     AND LET
C     D       = A2 - AX
C     E       = B2 - BY
C     F       = C2 - CZ
C     THESE ARE UNKOWN.THE VARIABLES A, B, C, D, E, AND F ARE INTERMED
C      VARIABLES INTRODUCED TO MAKE THE SOLVING FOR A2, B2, AND C2 A
C      LITTLE MORE READABLE.
      AABBCC  = A* A + B * B + C * C
C
      F       = B * DSIN( ROTANG ) * UX - A* DSIN( ROTANG ) * UY
```

```
     1         + C * DCOS( ROTANG )
C
      E        = ( B * F - AABBCC * DSIN( ROTANG ) * X ) / C
C
      D        = ( A * F + AABBCC * DSIN( ROTANG ) * Y ) / C
C
      DDEEFF   = D * D + E * E + F * F
C
      A2       = D + AX
      B2       = E + BY
      C2       = F + CZ
      DEBUG    = .TRUE.
      IF( DEBUG ) WRITE ( 6, 100 ) A1, B1, C1
100   FORMAT(' A1 = ', F14.9,5X , ' B1 = ',F14.9,5X,' C1 = ',F14.9 )

      IF( DEBUG ) WRITE( 6, 101 ) A2, B2, C2
101   FORMAT(' A2 = ', F14.9,5X, ' B2 = ',F14.9,5X,'C1 = ', F14.9 )
      IF( DEBUG ) WRITE( 6, 102 ) X,Y, Z
102   FORMAT(' X = ',F14.9,5X,' Y = ',F14.9,5X,' Z = ', F14.9 )
      IF( DEBUG ) WRITE ( 6, 103 ) AX, BY, CZ
103   FORMAT( ' AX = ',F14.9,5X,' BY = ',F14.9,5X,' CZ = ',F14.9 )
      IF ( DEBUG ) WRITE ( 6, 104 ) AABBCC, DDEEFF
104   FORMAT( 5X, ' AABBCC = ',F14.9 , 5X, 'DDEEFF = ', F14.9 )
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION DISP2A( A, B, C, D, E, F )
C
C     THIS FUNCTION COMPUTES THE DISTANCE BETWEEN A POINT ( A, B, C )
C     AND AN 'AXIS' WHICH IS A LINE THROUGH THE ORIGIN AND WHOSE
C     DIRECTION IS GIVEN BY THE VECTOR ( D, E, F ).
C
      IMPLICIT DOUBLE PRECISION ( A-H, O-Z )
C     FIRST FIND THE ANGLE BETWEEN THE POSITION VECTOR ( A, B, C )
C     AN D THE AXIS.
C
      ONEMAG  = DSQRT( A * A + B * B + C * C )
      TWOMAG  = DSQRT( D * D + E * E + F * F )
C
      COSTHE  = ( A * D + B * E + C * F ) / ( ONEMAG * TWOMAG )
C
C     COMPUTE THE DISTANCE FROMT THE POINT ( A, B, C ) TO THE AXIS.
C
      DISP2A  = ONEMAG * DSIN( DARCOS( COSTHE ) )
      RETURN
      END
```

The vita has been removed from the scanned document