RSD-TR-18-83

SOFTWARE SUPPORT FOR THE STEREO CAMERA: PHANTOM AND MEASUREMENT PROGRAMS[1]

Gideon Frieder and Myoung Lee

Department of Computer and Communication Sciences

The University of Michigan

Ann Arbor, Michigan 48109

November 1983

CENTER FOR ROBOTICS AND INTEGRATED MANUFACTURING

Robot Systems Division

COLLEGE OF ENGINEERING

THE UNIVERSITY OF MICHIGAN

ANN ARBOR, MICHIGAN 48109

[1]Supported in part by USAF/SAM via SCEEE Contract No. F-33616-78-D-0317 Task 81 and by CRIM via AFOSR Contract No. F49620-82-C-0089.

TABLE OF CONTENTS

1. Introduction

This set of programs is designed to provide a test bed for the verification and testing of the stereo camera system[1,2]. The stereo camera system can rapidly acquire the location of a large number of sample points on a surface. The system involves a coded-laser-beam-array projector with a programmable shutter, and a camera. The projector illuminates the surface with predetermined patterns. Points which are illuminated by the projector are imaged on the camera plane.

The stereo camera testing program consists of three parts: a phantom program, a surface mapping program, and an analysis program. The phantom program simulates a measurement of a given surface. It calculates the points on the surface, called "surface points", which are illuminated by the projector, and tne points on the camera plane ,called "image points", which are the images of the surface points. The surface mapping program reads the image points which were produced by the phantom program and calculates the surface points. The analysis program compares the surface points from the phantom program and the computed surface points from the surface mapping program, calculates the distance between the two and provides some error statistics.

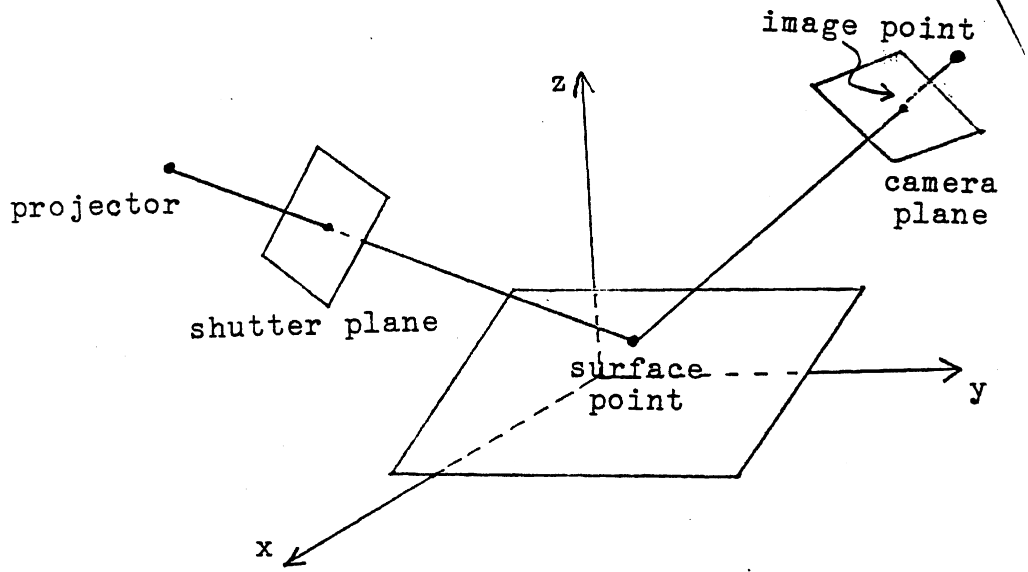


Figure 1. Stereo Camera System

2. The Phantom Program

The phantom program simulates a measurement process and thus provides data to be used by the surface mapping program. It produces one file of surface points and another of image points. The phantom program also produces a file of calibration data for the calibration option of the surface mapping program. For a surface mapping program which does not calibrate, it provides T and L matrices calculated directly from geometrical formulas.

2.1. The Coordinate System

The testbed programs use three coordinate systems. The world coordinate system is a three dimensional Cartesian coordinate system which is used to measure the position of the camera, the shutter and the surface to be mapped. Points on the surface illuminated by the projector and surface points which are computed by the surface mapping program are measured in this coordinate system. x, y, and z designate the three axes of this system.

The shutter coordinate system is a two dimensional Cartesian coordinate system on the shutter plane of the projector. u and v are the two axes of this system.

The camera coordinate system is a two dimensional Cartesian coordinate system on the camera plane. x_s and y_s are the two axes of this system.

When positions and parameters of the shutter (camera) are known, a coordinate of a point on the shutter (camera) plane can be converted into a coordinate in the world coordinate system, and vice versa.

2.2. Program Data

In the phantom program, the position and parameters of the shutter and the camera can be changed. In the current version, the surface to be mapped is always a plane, the equation of which can be changed. The size, the position, and the orientation of the calibration cube can be changed for the surface mapping program option which calibrates the T and L matrices. When the phantom program is invoked, it requests the following parameters from the user:

For the projector

Dist_sh, theta_sh, phi_sh	r θ,ϕ of the focal point of the projector
D_sh	Distance from the focal point of the projector to the center of the shutter plane
Su, Sv	Half width and half length of the shutter plane
N_u, N_v	Number of sample points in the whole shutter plane in u and v axis at present, $N_u, N_v \leq 64$

For the camera

Dist_ca, theta_ca, phi_ca	r θ,ϕ of the focal point of the camera
D_ca	Distance from the focal point of the camera to the center of the camera plane
Sx, Sy	Half width and half length of the camera plane
N_x, N_y	Number of sample points on the camera in x_s and y_s axis At present, $N_x, N_y \leq 128$ and are powers of 2

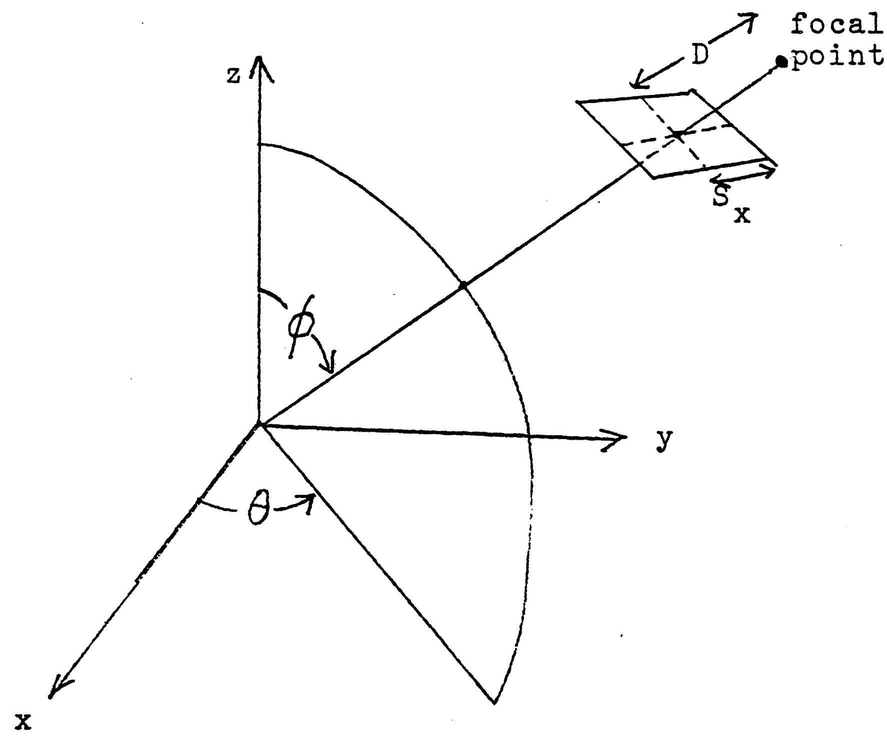
For the calibration cube

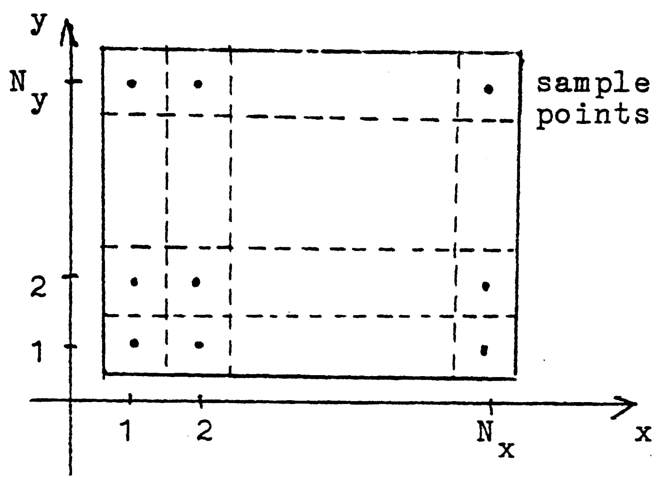
Length	Length of the side of the cube
Step	Step in u and v axis. To control the number of the calibration data points, the calibration cube is illuminated by a projector at each shutter point (u, v) with "step" apart.
Axis and Angle	Rotation of the cube around the world "axis"
xt, yt, zt	Translation of the cube along x, y, z axis.

For the surface to be mapped

A, B, C, D	Equation of the surface, $Ax + By + Cz + D = 0$

Figure 2. Parameters of the Phantom Program





2.3. Set Up of the Projector and the Camera

The projector and the camera are set up to aim at the same point, which is chosen as the origin of the world coordinate system. That means that the line perpendicular to the plane of the shutter (camera) passing through the center of the shutter (camera) surface and through the focal point of the shutter (camera), is also passing through the origin of the world coordinate system. Thus, for example, the shutter plane of the projector is perpendicular to the line which passes through the focal point of the projector and the origin of the world coordinate system. The orientation of the plane is such that two unit vectors a_u, a_v are the same as the unit vectors in the spherical world coordinate system a_θ, - a_ϕ. where θ, and ϕ are the parameters of the projector. The camera is positioned in the same way as the projector.

2.4. The Phantom Program Algorithm

The phantom program requests parameters from the user. When all the parameters are given, computation begins. The phantom program first finds the world coordinates of each sample point on the shutter plane (called "shutter points") of the projector. For each shutter point, it finds the line equation of each beam which passes through the focal point of the projector and the shutter point. The phantom program then computes the intersection of the beam and the plane to be mapped, which provides a point which is imaged on the camera plane.

The image points are calculated by three different methods. The first is the geometrical method. Here the equation of the line which passes through the focal point of the camera and the actual surface point is computed. Then the phantom program calculates the intersection of this line and the camera plane. That intersection point on the camera plane is rounded to the nearest sample point, which is the image point we want.

The second method is the one employed in graphics. We first find the transformation matrix T_e which transforms the world coordinate into the clipping coordinate[3]:

$$(x_c\ y_c\ x_c\ 1) = (x\ y\ z\ 1)\cdot T_e$$

where x_c, y_c, and z_c are the clip coordinates, and x, y, z are the world coordinates of the surface point. After it clips, the clipped coordinate is converted to the camera coordinate, which is the image point we want.

And finally, we find the T matrix which transforms the world coordinate directly into the camera coordinate. The T matrix is computed by multiplying T_e by the perspective transformation matrix T_p.

The three methods are basically equivalent. The image points obtained by these methods are compared to ensure that computational errors due to roundoffs and catastrofic cancellations in floating point arithmetic do not provides erroneous results. For each shutter point (u, v) on the shutter plane, the phantom program provides the shutter points (u, v), image points (x_s,y_s), and the intersection (x, y, z) of a beam passing through the shutter point and the surface plane, on file unit 2.

The phantom outputs image patterns on file unit 4. These patterns are determined by the space coding. When the surface mapping program does not calibrate the T and L matrices, the phantom program outputs the T and L matrices on file unit 3. When the surface mapping program calibrates the T and L matrices, the phantom program outputs the calibration data on file unit 3.

2.5. Calibration

Calibration is the process of determining the T and L matrices[2]. These matrices are calculated by the least square method[4], using data points which are measured on the surface of a known geometrical objects, in our case a cube. To determine the T and L matrices for various positions of the projector and the camera, the calibration cube should be properly positioned. Two or three faces of the calibration cube should be visible to both the projector and the camera, (three is the maximum), and there should be at least seven image points on the calibration cube. At present, the maximum number of calibration data points is 100. In the phantom program, it is possible to vary the size and to move the calibration cube. Positioning of the calibration cube involves rotation around the world coordinate x, y, z axes, which can be followed by a translation along the x, y, z axes. Since the rotation operation is not commutative, it is possible to specify rotation in any order, e.g. rotating θ_1 around x axis, followed by θ_2 around z axis, followed by θ_3 around x axis, followed by θ_4 around y axis, etc. After completing the rotation, translation of the cube is optionally performed. Translation is specified by xt, yt, zt, which are the translation along the x, y, z axes respectively. By rotating and translating the cube, it is possible to position the cube in any location, in any orientation. The position of the calibration cube is important for the accurate calibration of the T and L matrices.

When the size of the cube is entered, the calibration cube is located with its center at the origin of the world coordinate system, and its sides parallel to the x, y, z axes. With rotation and translation specified, a transformation matrix H and its inverse H^{-1} are calculated. The new position of the eight vertices and the new equation of the six faces are computed by multiplying them with the translation matrices, H and H^{-1}, respectively[5]. Among the six faces of the calibration cube, only the faces which are visible to both the projector and the camera are chosen. If the number of the faces visible to both the projector and the camera is one or zero, a different position of the calibration cube should be tried, since in this case the least square method does not provide a meaningful solution. When the number of the faces is two or three, intersections of the faces with the beams of the projector are calculated at each shutter point (u, v) with "step" apart, i.e. (u + m step, v + n step), where m, n = 1, 2, ... The intersections are imaged on the camera. The phantom program outputs the world coordinates of the intersections, (x, y, z), shutter column u, and the image points on the camera plane (x_s, y_s) on file unit 3. The number of the calibration data points is controlled by the parameter "step".

2.6. The Phantom Program

The phantom program consists of one main program and the following procedures. Read_Parameters, Read_Cal_Para, Initialize, Find_Tmat, Find_Lmat, Find_Ray_Eqs, Find_Beam_Eqs, Find_Beam_Planes, Map_To_Camera, Map_To_Cam_Tmat, Map_To_Cam_Geo, Make_Image_Pattern, Output_Pattern, Find_Cal_Face, Output_Cube_Image. Some procedures are redundant and are written to ensure the correctness of the program.

2.6.1. Read_Parameters

This procedure reads the parameters of the projector, the camera, and the equation of the plane to be mapped. It also writes some parameters on the calibration file (unit 3) for use by the surface mapping programs.

2.6.2. Read_Cal_Para

This procedure reads the calibration parameters, the size of the calibration cube, step in u, and v, and the rotation and the translation of the calibration cube. Any combination of the axes and angles can be entered. While reading the axis of the rotation and the angle of the rotation, this routine calculates the rotation matrix R. When it reads a character which is not an 'x', 'y', or 'z', it stops reading the rotation parameters and request translation parameters xt, yt, zt, along x, y, and z axis. It then computes the translation T and its inverse. The H matrix which accounts for both rotations and translations is found by multiplying the rotation matrix R by the translation matrix T. H^{-1} can be found by multiplying T^{-1} by R^{-1}. R^{-1} is equal to R^t since the R matrix is orthogonal, and T^{-1} is equal to a translation matrix which translates - xt, - yt, - zt.

2.6.3. Initialize

This procedure initializes the phantom program.

2.6.4. Find_Tmat

This procedure computes the T matrix from the camera parameters given by the user. T_e is the transformation matrix which transforms the world coordinates into the clip coordinates[6].

$$
\begin{bmatrix}
-F_x\sin(\theta) & -F_y\cos(\theta)\cos(\phi) & -\cos(\theta)\cos(\phi) & 0 \\
F_x\cos(\theta) & -F_y\sin(\theta)\cos(\phi) & -\sin(\theta)\sin(\phi) & 0 \\
0 & F_y\sin(\phi) & -\cos(\phi) & 0 \\
F_x(x\sin(\theta)-y\cos(\theta)) & \begin{matrix} F_y(-z\sin(\phi) \\ +x\cos(\theta)\cos(\phi) \\ +y\sin(\theta)\cos(\phi)) \end{matrix} & \begin{matrix} z\cos(\phi) \\ +x\cos(\theta)\sin(\phi) \\ +y\sin(\theta)\sin(\phi) \end{matrix} & 1
\end{bmatrix}
$$

where $F_x = D / S_x$ and $F_y = D / S_y$. T_p is the perspective transformation matrix.

$$
T_p = \begin{bmatrix} V_{sx} & 0 & 0 & 0 \\ 0 & V_{sy} & 0 & 0 \\ V_{cx} & V_{cy} & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}
$$

where $V_{sx} = N_x / 2$, $V_{sy} = N_y / 2$, and $V_{cx} = (N_x + 1) / 2$, $V_{cy} = (N_y + 1) / 2$. The T matrix is computed by

$$
T = T_e \cdot T_p
$$

2.6.5. Find_Lmat

This procedure computes the L matrix from the parameters of the projector given by the user. All the equations are the same as for the T matrix except that the projector parameters are used instead of the camera parameters.

2.6.6. Find_Beam_Eqs

This procedure computes the line equation of every beam that starts at the focal point of the projector and passes through the shutter point. The unit vectors on the shutter plane of the projector $\vec{a}_u$ and $\vec{a}_v$ are

$$
\vec{a}_u = \vec{a}_\theta = -\vec{i}\sin(\theta) + \vec{j}\cos(\theta)
$$

$$
\vec{a}_v = -\vec{a}_\phi = -\vec{i}\cos(\theta)\cos(\phi) - \vec{j}\sin(\theta)\cos(\phi) + \vec{k}\sin(\phi)
$$

where $\vec{i}$, $\vec{j}$, and $\vec{k}$ are the unit vectors of the Cartesian world coordinate system, and $\vec{a}_\phi$, and $\vec{a}_\theta$

are the unit vectors of the spherical world coordinate system. The world coordinates of the shutter points $\vec{r}_{u,v}$ are

$$\vec{r}_{u,v} = \vec{r}_{sh} + [(u-\frac{N_u}{2}) - 0.5](2\frac{S_u}{N_u})\vec{a}_u + [(v-\frac{N_v}{2}) - 0.5](2\frac{S_v}{N_v})\vec{a}_v$$

where $\vec{r}_{sh}$ is the world coordinate of the center of the shutter plane of the projector. The direction numbers a, b, and c of the lines passing through the focal point (x_f,y_f,z_f) and the shutter point $\vec{r}_{u,v}$ are defined as

$$\frac{x-x_f}{a} = \frac{y-y_f}{b} = \frac{z-z_f}{c}$$

where

$$a = \text{x coordinate of } \vec{r}_{u,v}-x_f$$

$$b = \text{y coordinate of } \vec{r}_{u,v}-y_f$$

$$c = \text{z coordinate of } \vec{r}_{u,v}-z_f$$

2.6.7. Find_Ray_Eqs

This procedure computes the line equation of every ray which starts from the focal point of the camera and passes through the sample points of the camera plane. All the equations are same as Find_Beam_Eqs except that camera parameters used instead of the projector parameters. This routine is used to ensure the correctness of the T matrix.

2.6.8. Find_Beam_Plane

This procedure computes for every column u the equation of the plane which is defined by the focal point (x_f,y_f,z_f) of the projector and two points (u, 1), (u, N_v) on the same column u of the shutter plane of the projector. This is used to ensure the correctness of the L matrix. When a_1,b_1,c_1 and a_n,b_n,c_n are the direction numbers of the beams passing through the points (u, 1) and (u, N_v) respectively, the equation of the plane is $Ax + By + Cz + D = 0$, where

$$A = b_1c_n - b_nc_1$$

$$B = c_1a_n - c_na_1$$

$$C = a_1b_n - a_nb_1$$

$$D = -x_fA - y_fB - z_fC$$

2.6.9. Map_to_Camera

This procedure computes the image of the surface point that is illuminated by the projector. The homogeneous coordinate of the surface point in the world coordinate system is multiplied by transformation matrix T_e.

$$(x_c\ y_c\ x_c\ 1) = (x\ y\ z\ 1)\cdot T_e$$

where x_c, y_c, and z_c are the clip coordinates, and x, y, z are the world coordinate of the surface point[3]. Those points with

$$-z_c \leq x_c \leq z_c \quad \text{and} \quad -y_c \leq x_c \leq y_c$$

are imaged on the camera plane. Finally image points are found by

$$x_s = \mathrm{round}\left(\left(\frac{x_c}{z_c}\right)V_{sx} + V_{cx}\right)$$

$$y_s = \mathrm{round}\left(\left(\frac{y_c}{z_c}\right)V_{sy} + V_{cy}\right)$$

where V_{sx}, V_{sy}, V_{cx}, and V_{cy} are defined and computed as in the procedure Find_Tmat.

2.6.10. Map_to_Cam_Tmat

This procedure performs the same calculations as Map_To_Camera. It is written to compare the image points obtained by this procedure with that obtained by Map_To_Camera. This procedure computes image points by multiplying the homogeneous coordinate of the surface point by the T matrix and then rounding it.

2.6.11. Map_to_Cam_Geo

This procedure performs again the same computation as the two former ones, and is written to compare the image points obtained by this routine with those obtained by the former methods. The equation of the line which passes through the focal point of the camera (x_f,y_f,z_f) and the surface point (x_p,y_p,z_p) is

$$\frac{x-x_f}{a} = \frac{y-y_f}{b} = \frac{z-z_f}{c}$$

where $a = x_p-x_f$, $b = y_p-y_f$, $c = z_p-z_f$. The equation $Ax + By + Cz + D = 0$ of the camera plane is easily computed. A, B, C, and D are the direction numbers of the line which passes through the origin of the world coordinate and the center of the camera plane (x_c,y_c,z_c).

$$A = \cos(\theta)\sin(\phi)$$

$$B = \sin(\theta)\sin(\phi)$$

$$C = \cos(\phi)$$

$$D = -(Ax_c+By_c+Cz_c)$$

Finally the intersection of the line and the plane is (x_f+ta, y_f+tb, z_f+tc), where

$$t = -\frac{Ax_f+By_f+Cz_f+D}{Aa+Bb+Cc}$$

The intersection point is rounded to the nearest sample point on the camera plane.

2.6.12. Make_Image_Pattern

This procedure computes the image points (x_s, y_s) that correspond to the shutter points (u, v). For each shutter point, the line equation of the beam that passes through the focal point and the shutter point is already computed by the procedure Find_Beam_Eqs. In Make_Image_Pattern the intersection of the beam and the plane to be mapped are computed, and are imaged on the camera plane by the procedures Map_to_Camera, Map_to_Cam_Tmat, and Map_to_Cam_Geo. The computed image points are then checked for equality.

If the two different surface points are mapped unto the same image point on the camera, the space coding methodology will fail to identify a proper column value for the image points and all the surface acqusition methodology will therefore fail. To avoid this situation, only the first shutter point which generated that image is recorded, and other shutter points are flagged on the terminal. Such a situation essentially signals a failure of the measurement.

2.6.13. Output_Pattern

This procedure creates a file (on unit 4) of all the patterns as dictated by the space coding method. Each entry consists of pairs of x_s, y_s. Each pattern is terminated by the pair -1, -1. The number of patterns is a function of the shutter parameters.

2.6.14. Find_Cal_Face

This procedure computes the new position of the eight vertices of the calibration cube and the new equations of the six faces of the calibration cube. At first, the calibration cube is located with its center at the origin of the world coordinate system. Denoting the length of the edge of the cube by 2a, the homogeneous coordinates of the eight vertices of the cube are

$$V_1 = (a,a,a,1) \quad V_2 = (a,a,-a,1)$$

$$V_3 = (a,-a,-a,1) \quad V_4 = (a,-a,a,1)$$

$$V_5 = (-a,a,a,1) \quad V_6 = (-a,a,-a,1)$$

$$V_7 = (-a,-a,-a,1) \quad V_8 = (-a,-a,a,1)$$

and the homogeneous equations of the six faces of the cube are

$$F_1 = (0,0,1,-a) \quad F_2 = (-1,0,0,-a)$$

$$F_3 = (0,0,-1,-a) \quad F_4 = (1,0,0,-a)$$

$$F_5 = (0,-1,0,-a) \quad F_6 = (0,1,0,-a)$$

The new homogeneous coordinate $V_{n,i}$ of the moved vertices are found by

$$V_{n,i} = V_i \cdot H$$

where H is the transformation matrix computed at the procedure Read_Cal_Para. The new homogeneous equations of the moved faces $F_{n,i}$ are found by[5]

$$F_{n,i} = H^{-1} \cdot F_i$$

The number of the faces visible by both the projector and the camera are then computed. For the face F_i to be visible to both the projector and the camera, the focal points of both the projector and the camera should be at the same side of the face and should be at the opposite side of the face of the vertex V_{i+1}. Since the sign of V·F tells at which side of the plane F lies the point V, the faces visible to both can be determined.

2.6.15. Output_Cube_Image

This procedure finds the intersection of the beams of the projector and the faces of the calibration cube for each shutter point, where u and v are separated by "step". This routine outputs the world coordinate of the surface point, the column of the shutter which illuminated the surface point, and the image points on the camera of that surface point.

2.6.16. Main Program

The main program opens file units 2, 3, and 4, and then calls the necessary procedures.

3. Surface Mapping Program

This program calculates the surface point from the patterns of image points provided by either the phantom program or by an actual measurement. There are two versions of the surface mapping program, one which reads the T and L matrices provided by the phantom program, the other which calibrates the T and L matrices by the least square method. The surface mapping program reads the patterns of image points from a file (unit 4). From this data, the procedure computes the column of the shutter point which generated the image point, utilizing the space coding method[2]. Using the T and L matrices, the program then solves a set of linear equations to find the surface point. The surface mapping program consists of a main program and the following routines: Init, Calin, Imgin, Surout, Tmat, Lmat, Map, Column, and Setcof.

3.1. Init

This routine calls the Calin routine, and Performs initialization.

3.2. Calin

This routine reads the calibration data from file unit 3. The format of the file on unit 3 is

line	contents
1	number of sample points along u, v axes on the shutter plane
2	number of sample points along x_s, y_s axes on the camera plane
3	a point (x, y, z) on the face of the calibration cube which is illuminated by the projector
4	column u on the shutter plane, and image point (x_s, y_s)
	lines 3 - 4 are repeated for each illuminated point

3.3. Imgin

This routine reads the patterns of images on the camera plane from file unit 4. File unit 4 contains lines of x_s and y_s for each pattern. End of a pattern is indicated by a line of -1 -1.

3.4. Tmat

This routine calculates the T matrix by the least square method utilizing the L2 routine. The routine calculates the normalized T matrix by setting $T_{44} = 1$ and dividing every element of the T matrix by T_{44}. The routine sets up the matrices B, C that are passed to the L2 routine to solve the equation $Bx = C$ by the least square method, where x is the normalized T matrix,

$$x^t = (T_{11}, T_{12}, T_{13}, T_{14}, T_{21}, T_{22}, T_{23}, T_{24}, T_{41}, T_{42}, T_{43})$$

and B, C are the least square matrices[7],

$$B = \begin{bmatrix} x^1 & 0 & -x^1x_s^1 & y^1 & 0 & -y^1x_s^1 & z^1 & 0 & -z^1x_s^1 & 1 & 0 \\ 0 & x^1 & -x^1y_s^1 & 0 & y^1 & -y^1y_s^1 & 0 & z^1 & -z^1y_s^1 & 0 & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ x^i & 0 & -x^ix_s^i & y^i & 0 & -y^ix_s^i & z^i & 0 & -z^ix_s^i & 1 & 0 \\ 0 & x^i & -x^iy_s^i & 0 & y^i & -y^iy_s^i & 0 & z^i & -z^iy_s^i & 0 & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}$$

$$C^t = (-x_s^1 -y_s^1 \cdot -x_s^i -y_s^i \cdot)$$

where $i = 1, 2, \ldots N$, N is the number of calibration data points, (x^i, y^i, z^i) are the world coordinates of a point on the face of the calibration cube, and (x_s^i, y_s^i) is the image point on the camera plane.

When the surface mapping program does not calibrate the T matrix, this routine reads the T matrix calculated by the phantom program.

3.5. Lmat

This routine finds the L matrix by the least square method. By setting $L_{44} = 1$, and dividing every element of L matrix by L_{44}, one finds the normalized L matrix. Following this, the routine calculates the matrices B, C to be passed to the L2 routine to solve the equation Bx = C by the least square method, where x is the normalized L matrix,

$$x = (L_{11}, L_{14}, L_{21}, L_{24}, L_{31}, L_{34}, L_{41})$$

and B, C are the least square matrices,

$$B = \begin{bmatrix} x^1 & -x^1u^1 & y^1 & -y^1u^1 & z^1 & -z^1u^1 & 1 \\ . & . & . & . & . & . & . \\ x^i & -x^iu^i & y^i & -y^iu^i & z^i & -z^iu^i & 1 \\ . & . & . & . & . & . & . \end{bmatrix}$$

$$C = \begin{bmatrix} u^1 \\ . \\ u^i \\ . \end{bmatrix}$$

where u^i is the column of the shutter.

When the surface mapping program does not calibrate the L matrix, this routine reads the L matrix calculated by the phantom program.

3.6. Map

This routine computes the surface points using the T and L matrices. It first finds the column of the shutter which generated the image by calling the routine Column. Next the routine sets up a linear equation by calling the routine Setcof, and then solves the linear equation by calling L2.

3.7. Column

This routine uses the space coding method to infer the shutter column value for each image point.

3.8. Setcof

This routine sets up the linear equation Ax = B where x is the world coordinate (x, y, z) of the surface point, and[1]

$$A = \begin{bmatrix} T_{11}-T_{14}x_s & T_{21}-T_{24}x_s & T_{31}-T_{34}x_s \\ T_{12}-T_{14}y_s & T_{22}-T_{24}y_s & T_{32}-T_{34}y_s \\ L_{11}-L_{14}u & L_{21}-L_{24}u & L_{31}-L_{34}u \end{bmatrix}$$

$$B = \begin{bmatrix} -T_{41}+T_{44}x_s \\ -T_{42}+T_{44}y_s \\ -L_{41}+L_{44}u \end{bmatrix}$$

3.9. Space Coding

Space coding is used to find the column of the shutter which generated the image. In this coding method, the number of patterns necessary to encode and decode the column is $\log_2$(number of columns) + 1. The space coding method is explained in detail elsewhere[1]. As an example, for the case of $N_u = 8$, the 4 patterns of the space coding are:

patterns	columns							
	1	2	3	4	5	6	7	8
pattern 1	O	O	O	O	O	O	O	O
pattern 2	X	X	X	X	O	O	O	O
pattern 3	X	X	O	O	X	X	O	O
pattern 4	X	O	X	O	X	O	X	O

where "O" indicates that the column is open, and "X" indicates that the column is closed.

4. The Analysis Program

This program reads the file of surface points from the phantom program. This file is found on unit 2, and it contains, for all points, the shutter coordinate (u, v), the intersection (x_1,y_1,z_1) of the beam which passes through the shutter point (u, v) and the surface plane, and the image point (x_s, y_s) which is the image of the intersection on the camera plane. The analysis program reads another file of computed surface points provided by the surface mapping program. This file consists of lines of a shutter column u, an image point (x_s, y_s), and an computed surface point (x_2,y_2,z_2). By comparing u, x_s, and y_s, the analysis program matches each surface point from the two files. It then calculates the distance between the two, the average distance, and the standard deviation of the distances. When the two files does not match, the program flags the surface points which do not match. This indicates something amiss in the phantom program or in the surface mapping program.

5. General Description of the Program Files

There are four sets of files: Phan.geo and Eye.geo, Phan.tmat and Eye.tmat, Phan.tlmat and Eye.tlmat, and finally Phan.cal and Eye.cal. Programs beginning with "phan" are phantom programs, and programs beginning with "eye" are surface mapping programs.

Phan.geo and Eye.geo do not use the T and L matrices. They use purely geometrical computation. Instead of T matrix, Eye.geo uses equations of rays which starts at the focal point of the camera and passes through the sample points of the camera plane. Instead of L matrix, Eye.geo uses equations of planes which pass through the focal point of the projector and two points on the same column of the shutter.

Phan.tmat and Eye.tmat use the T matrix as computed from the parameters of the camera, and generate the same output as the .geo programs. This provides for an independent calculation which assures the correctness of the T matrix.

Phan.tlmat and Eye.tlmat use both the T matrix and the L matrix which is computed from the parameters of the projector. Again the same output as .geo and .tmat programs is generated, which ensuring the correctness of the L matrix. All the previous programs do not calibrate the T and L matrices. All the previous set of programs are basically the same, and Phan.tlmat and Eye.tlmat are chosen to represent the geometrical approach.

Phan.cal and Eye.cal use T and L matrices, and calibrate the T and L matrices. The T and L matrices calibrated here should be close to the T and L matrices calculated above, which ensures the correctness of the calibration.

The Support.2 file includes among other routines the routine L2 which solves the least square problem. The solution of the L2 routine was compared with the International Mathematical and Statistical Library routine LLBQF[IMSL] and was found accurate.

The analysis program is found in the file Analysis.

6. Simulations

With the above set of programs, it is possible to simulate various situations and analyze the output obtained, e.g. the effect of the precision of the phantom output on the accuracy of the surface mapping program can be tested. The precision of the phantom output is determined by the number of the sample points on the camera plane, the number of the sample points on the shutter plane of the projector, and the precision of the intersection points of the beam of the projector with the calibration cube. Since phan.tlmat and eye.tlmat calculate the T and L matrices by geometrical formulas, they represent the geometrical approach and the data from them is the bound of the accuracy which can be attained by the surface mapping program.

Table 1 illustrates the effect of the positions of the shutter and the camera on the accuracy of the stereo camera system. With the shutter fixed, various positions of the camera has been tried. The numbers are the average distance between the actual surface point and the computed surface point, and its standard deviation.

Table 2 illustrates the effect of the number of sample points on the accuracy of the stereo camera system. Here, the column headed by "geometry" is the lower bound which can be obtained by the stereo camera system.

In Table 3, the number of decimal places of the surface point on the face of the calibration cube has been changed.

In most situations, the calibration of the T and L matrices was satisfactory. However, sometimes calibration does not provide accurate T and L matrices even though the condition number from the L2 routine is not high. Table 4 illustrates the effect of the position of the calibration cube on the accuracy of the stereo camera system.

This set of programs can be used to find the satisfactory position of the calibration cube: First, find the average distance by the set of programs which use geometry. This is the lower bound. Next, try various positions of the calibration cube until you obtain the average distance close to the lower bound.

camera position θ	φ	average	standard deviation
45	60	1.26	1.33
60	60	.63	.60
90	60	.32	.22
135	60	.29	.12
210	60	1.40	2.03
135	5	6.91	21.1
135	15	.91	.73
135	30	.73	.17
135	60	.29	.12
135	75	.27	.12
135	85	.25	.12

Table 1. Effect of Camera Position

shutter: $r = 20$, $\theta = 30$, $\phi = 45$, $D = 4$, Su, Sv $= 2$ $N_u, N_v = 16$
camera: $r = 20$, $D = 4$, Sx, Sy $= 2$ $N_x, N_y = 32$
equation of the plane: $z = -1$

N_x, N_y	geometry average	geometry standard deviation	calibration average	calibration standard deviation	
32	1.26	1.33	2.42	2.92	$N_u, N_v = 8$
128	.29	.31	.49	.53	
32	1.23	1.42	2.04	2.62	$N_u, N_v = 16$
128	.31	.32	.35	.37	
32	1.44	1.26	2.58	4.78	$N_u, N_v = 32$
128	.30	.31	.45	.55	

Table 2. Effect of Number of Sampling Points

shutter: $r = 20$, $\theta = 30$, $\phi = 45$, $D = 4$, Su, Sv = 2
camera: $r = 20$, $\theta = 60$, $\phi = 60$, $D = 4$, Sx, Sy = 2
calibration: size = 8, rotation = x 10, y 45, z 30, translation = 0, 0, 0
equation of the plane: $z = -1$

number of decimal places	average	standard deviation
7	1.08	1.37
3	1.08	1.37
2	1.06	1.35
1	1.06	1.32
0	4.80	6.23

Table 3. Effect of Number of Decimal Places

shutter: r = 20, θ = 30, ϕ = 45, D = 4, Su, Sv = 2 N_u, N_v = 16
camera: r = 20, θ = 60, ϕ = 60, D = 4, Sx, Sy = 2 N_x, N_y = 32
calibration: size = 8, rotation = x 10, y 45, z 30, translation = 0, 0, 0
equation of the plane: z = - 1

rotations x y z			average	standard deviation	
30	30	60	10.7	16.0	$N_u,N_v = 8$
10	45	30	1.07	.98	
geometry			.70	.52	
30	30	60	.99	.71	$N_u,N_v = 16$
10	45	30	.95	.70	
geometry			.77	.49	

Table 4. Effect of Rotation of Calibration Cube

shutter: $r = 20$, $\theta = 30$, $\phi = 45$, $D = 4$, Su, Sv = 2
camera: $r = 20$, $\theta = 60$, $\phi = 60$, $D = 4$, Sx, Sy = 2
calibration: size = 8, translation = 0, 0, 0
equation of the plane: $z = -1$

7. References

[1] M. D. Altschuler, J. L. Posdamer, G. Frieder, "The Numerical Stereo Camera," SPIE, vol. 283, 1982

[2] J. L. Posdamer, M. D. Altschuler, "Surface Measurement by Space-encoded Projected Beam System," Computer Graphics and Image Processing, vol. 18 pp 1-17, 1982

[3] W. M. Newman, Principles of Interactive Computer Graphics, 2nd ed. pp 339-344, McGraw-Hill, 1979

[4] G. E. Forsythe, M. Malcolm, C. B. Moler, Computer Methods for Mathematical Computations, pp 192-235, Prentice-Hall, 1977

[5] W. M. Newman, et. al., op. cit. pp 492-501

[6] W. M. Newman, et. al., op. cit., pp348-351

[7] D. V. Roger, J. A. Adams, Mathematical Elements for Computer Graphics, pp 78-82, McGraw-Hill, 1976

8. Program Listings

8.1. Phantom Program

LINE NUMBER B P C I STMT # S O U R C E P R O G R A M

```
V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
PROGRAM Phan_Cal ( INPUT, OUTPUT ) ;

/*
 *
 *
 *
 *
 *                     Phantom Program
 *              for Testing Numerical Stereo Camera
 *
 *                  Programmed by Myoung Lee
 *
 *                        July 1983
 *
 *
 *    The Phantom Program provides data for the Surface Mapping
 *    Program.  This program can be used for the testing and
 *    experiment of Surface Mapping Program.
 *    This also generates actual surface points to be mapped
 *    into camera points.
 *    This calculates T and L matrix by geometrical consideration.
 *
 *    For a detailed description of the phantom program and the
 *    related discussion, refer to the separate documentation.
 *
 */
```

LINE NUMBER B P C I   STMT #   S O U R C E   P R O G R A M

```
V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V

CONST

  u_max  = 64 ;    { maximum number of sample points along u axis }
  v_max  = 64 ;    { maximum number of sample points along v axis }
  xs_max = 128 ;   { maximum number of smaple points along Xs axis}
  ys_max = 128 ;   { maximum number of sample points along Ys axis}

  unobserved = -999 ;
  pre  = 7 ;         { number of decimal places of real number }
  wid  = 14 ;        { length of the real field }
  widf = wid - 1 ; { for FORTRAN interface }
  len  = 6 ;         { length of the integer field }
  lenf = len - 1 ; { for FORTRAN interface }


TYPE

 Vector = ARRAY (. 1..3 .) of REAL ;
 Mat_4  = ARRAY (. 1..4, 1..4 .) of REAL ;

VAR

  Cal_f,        { Calibration file }
  Img_f,        { Image file }
  Ana_f : TEXT ;{ File for Anlysis program }

  Tmat,          { T matrix }
  Lmat,          { L matrix }
  View_xform : ARRAY (. 1..4, 1..4 .) of REAL ;
                 { Transforms world coordinates into clip coordinates }

  Ray  : ARRAY (. 1..xs_max, 1..ys_max, 1..3 .) of REAL ;
           { direction numbers of each ray passing through the focal point }
           { of the camera and sample points on the camera plane }

  Beam : ARRAY (. 1..u_max, 1..v_max, 1..3 .) of REAL ;
           { direction numbers of each beam passing through the focal point }
           { of the shutter and sample points on the shutter plane }

  Beam_plane : ARRAY (. 1..u_max, 1..4 .) of REAL ;
           { equation of the planes passing through the focal point of the shutter }
           { and two sample points on the same column of the shutter plane }
  Screen : ARRAY (. 1..xs_max, 1..ys_max .) of BOOLEAN ;
           { If two different shutter points imaged on the same camera point }
           { Space Coding gives a spurious column }
           { To avoid this, the camera point is recorded true when it is imaged }

  Pattern : ARRAY (. 1..u_max, 1..v_max, 1..2 .) of INTEGER ;
           { image point for each sutter point }

  H_mat,          { calibration cube transformation matrix }
```

```
V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V


{ Every name ending with "sh" is associated with shutter }
{ "u" and "v" are coordinate system in the shutter plane }
 .
{ Every name ending with "ca" is associated with camera  }
{ "x" and "y" are coordinate system in the camera plane  }

{ shutter parameters }


  Theta_sh,
  Phi_sh,
  Dist_sh : REAL ;  { position of the center of the shutter }

  Center_sh,        { Coordinate of the center of the shutter }
  FP_sh,            { coordinate of the Focal Point of the shutter}
  Dir_no_sh : Vector ; { direction number of the center of the shutter }

  D_sh,        { distance from center of the shutter to focal point of the shutter }
  Su,          { width of the shutter plane }
  Sv : REAL ;  { height of the shutter plane }

  Nu,              { Number of sample points in u direction }
  Nv : INTEGER ; { Number of sample points in v direction }


{ camera parameters }


  Theta_ca,
  Phi_ca,
  Dist_ca : REAL ;  { position of the center of the camera }

  Center_ca,
  FP_ca,
  Dir_no_ca : Vector ;
{ names are    }
  D_ca,                       { same as     }
  Sx,                         { above       }
  Sy : REAL ;                 { with camera }
                              { instead of  }
  Nx,                         { shutter     }
  Ny : INTEGER ;

  Map_plane : ARRAY (. 1..4 .) of REAL ; { Plane to be mapped }


 { calibration cube }

   Face : ARRAY (. 1..3, 1..4 .) of REAL ;
          { equation of the visible faces of the calibration cube}

```

LINE NUMBER B P C I STMT # SOURCE PROGRAM

```
V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
    Num_face : INTEGER ;  { number of faces visible by shutter and camera }

    a : REAL ;  { length of the side of a cube = 2 * a }

    xt, yt, zt : REAL ; { translation of the cube }

    Step : INTEGER ; { in calibration }

```

```
LINE NUMBER B P C I  STMT #                  S O U R C E   P R O G R A M

                               V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
   142.000
   143.000
   144.000                     /*
   145.000                      *   Make_Identity
   146.000                      *
   147.000                      *       This makes a 4 by 4 matrix an identity matrix
   148.000                      */
   149.000
   150.000    1                PROCEDURE Make_Identity ( VAR Matrix : mat_4 ) ;
   151.000    1
   152.000    1                 VAR
   153.000    1                    i, j : INTEGER ; { indices }
   154.000    1
   155.000    1                 BEGIN
   156.000    1
   157.000    1                   { Initialize 4 by 4 Matrix as identity matrix }
   158.000    1
   159.000    1   1        1        FOR i := 1 TO 4  DO
   160.000    1   2        2           FOR j := 1 TO 4 DO
   161.000    1            3               Matrix (.i, j.) := 0 ;
   162.000    1
   163.000    1   1        4        FOR i := 1 TO 4  DO
   164.000    1            5            Matrix (.i, i.) := 1 ;
   165.000    1
   166.000                       END ;
   167.000
   168.000
```

```
LINE NUMBER B P C I  STMT #                      S O U R C E   P R O G R A M

                              V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
   169.000                   |
   170.000                   |
   171.000                   |/*
   172.000                   | *   Mult_Mat4
   173.000                   | *
   174.000                   | *
   175.000                   | *      This multiplies two 4 by 4 matrix
   176.000                   | */
   177.000                   |
   178.000   1               |PROCEDURE Mult_Mat4 ( VAR Mat : mat_4 ;
   179.000   1               |                      Mat_a, Mat_b : mat_4 ) ;
   180.000   1               |
   181.000   1               | VAR
   182.000   1               |    i, j, k : INTEGER ; { indices }
   183.000   1               |
   184.000   1               | BEGIN
   185.000   1               |
   186.000   1               |   { multiply 4 by 4 Matrix Mat_a and Mat_b }
   187.000   1               |
   188.000   1   1       1   |     FOR i := 1 TO 4  DO
   189.000   1   2       2   |        FOR j := 1 TO 4 DO
   190.000   1   2           |          BEGIN
   191.000 1 1   2       3   |            Mat (.i, j.) := 0 ;
   192.000 1 1   3       4   |            FOR k := 1 TO 4 DO
   193.000 1 1   2       5   |                Mat (.i, j.) := Mat (.i, j.) + Mat_a (.i, k.) * Mat_b (.k, j.) ;
   194.000   1   2           |          END
   195.000                   | END ;
   196.000                   |
```

```
LINE NUMBER B P C I  STMT #               S O U R C E   P R O G R A M

                        V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
                        |
                        |
                        |/*
                        | *   Read_Parameters
                        | *
                        | *       This reads parameters for phantom program
                        | */
                        |
            1           |PROCEDURE Read_Parameters ;
            1           |
            1           |
            1           | BEGIN
            1           |
            1        1  |     WRITELN ;
            1        2  |     WRITELN ( ' Enter Parameters ' ) ;
            1        3  |     WRITELN ;
            1           |
            1        4  |     WRITELN ( ' ---- shutter parameters ---- ' ) ;
            1        5  |     WRITELN ;
            1           |
            1        6  |     WRITELN ( '  focal point -- r ' ) ;
            1        7  |     READLN ( Dist_sh ) ;
            1           |
            1        8  |     WRITELN ( '  focal point -- theta ' ) ;
            1        9  |     READLN ( Theta_sh ) ;
            1           |
            1       10  |     WRITELN ( '  focal point -- phi ' ) ;
            1       11  |     READLN ( Phi_sh ) ;
            1           |
            1       12  |     WRITELN ( '  shutter plane -- D ' ) ;
            1       13  |     READLN ( D_sh ) ;
            1           |
            1       14  |     WRITELN ( '  shutter plane -- Su' ) ;
            1       15  |     READLN ( Su ) ;
            1           |
            1       16  |     WRITELN ( '  shutter plane -- Sv ' ) ;
            1       17  |     READLN ( Sv ) ;
            1           |
            1       18  |     WRITELN ( '  shutter plane -- Nu ' ) ;
            1       19  |     READLN ( Nu ) ;
            1       20  |     WRITE ( cal_f, Nu :lenf ) ;
            1           |
            1       21  |     WRITELN ( '  shutter plane -- Nv' ) ;
            1       22  |     READLN ( Nv ) ;
            1       23  |     WRITELN ( cal_f, Nv :len ) ;
            1           |
            1       24  |     WRITELN ( ' ---- camera parameters ---- ' ) ;
            1       25  |     WRITELN ;
            1           |
            1       26  |     WRITELN ( '  focal point -- r ' ) ;
            1       27  |     READLN ( Dist_ca ) ;
            1           |
            1       28  |     WRITELN ( '  focal point -- theta ' ) ;
```

```
LINE NUMBER B P C I  STMT #                    S O U R C E   P R O G R A M

                              V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
    250.000   1          29 |     READLN ( Theta_ca ) ;
    251.000   1             |
    252.000   1          30 |     WRITELN ( '  focal point -- phi ' ) ;
    253.000   1          31 |     READLN ( Phi_ca ) ;
    254.000   1             |
    255.000   1          32 |     WRITELN ( '  camera plane -- D ' ) ;
    256.000   1          33 |     READLN ( D_ca ) ;
    257.000   1             |
    258.000   1          34 |     WRITELN ( '  camera plane -- Sx' ) ;
    259.000   1          35 |     READLN ( Sx ) ;
    260.000   1             |
    261.000   1          36 |     WRITELN ( '  camera plane -- Sy ' ) ;
    262.000   1          37 |     READLN ( Sy ) ;
    263.000   1             |
    264.000   1          38 |     WRITELN ( '  camera plane -- Nx ' ) ;
    265.000   1          39 |     READLN ( Nx ) ;
    266.000   1          40 |     WRITE ( cal_f, Nx :lenf ) ;
    267.000   1             |
    268.000   1          41 |     WRITELN ( '  camera plane -- Ny' ) ;
    269.000   1          42 |     READLN ( Ny ) ;
    270.000   1          43 |     WRITELN ( cal_f, Ny :len ) ;
    271.000   1             |
    272.000   1          44 |     WRITELN ( '  Equation of the plane to be mapped -- A, B, C, D' ) ;
    273.000   1          45 |     READLN ( Map_plane(.1.), Map_plane(.2.), Map_plane(.3.), Map_plane(.4.) ) ;
    274.000   1          46 |     WRITELN ;
    275.000   1             |
    276.000                 | END ;
```

```
LINE NUMBER B P C I  STMT #                  S O U R C E   P R O G R A M

                             V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
  277.000                   |
  278.000                   |
  279.000                   |/*
  280.000                   | *   Intialize
  281.000                   | *
  282.000                   | *
  283.000                   | */
  284.000                   |
  285.000     1             |PROCEDURE Initialize ;
  286.000     1             |
  287.000     1             | VAR
  288.000     1             |   ang_rad : REAL ; { angle to radian conversion }
  289.000     1             |   i, j : INTEGER ; { indices }
  290.000     1             |
  291.000     1             | BEGIN
  292.000     1             |
  293.000     1             |  { convert angles to radian }
  294.000     1             |
  295.000     1          1  |   ang_rad := 3.141592 / 180 ;
  296.000     1             |
  297.000     1          2  |   Theta_sh := Theta_sh * ang_rad ;
  298.000     1          3  |   Phi_sh := Phi_sh * ang_rad ;
  299.000     1             |
  300.000     1          4  |   Theta_ca := Theta_ca * ang_rad ;
  301.000     1          5  |   Phi_ca := Phi_ca * ang_rad ;
  302.000     1             |
  303.000     1             |  { calculate direction numbers }
  304.000     1             |
  305.000     1          6  |   Dir_no_sh (.1.) := COS ( Theta_sh )* SIN ( phi_sh ) ;
  306.000     1          7  |   Dir_no_sh (.2.) := SIN ( Theta_sh )* SIN ( phi_sh ) ;
  307.000     1          8  |   Dir_no_sh (.3.) := COS ( phi_sh ) ;
  308.000     1             |
  309.000     1          9  |   Dir_no_ca (.1.) := COS ( Theta_ca )* SIN ( phi_ca ) ;
  310.000     1         10  |   Dir_no_ca (.2.) := SIN ( Theta_ca )* SIN ( phi_ca ) ;
  311.000     1         11  |   Dir_no_ca (.3.) := COS ( phi_ca ) ;
  312.000     1             |
  313.000     1             |  { calcuate Cartesian coordiate of focal points and center of shutter and camera }
  314.000     1             |
  315.000     1   1     12  |   FOR i := 1 TO 3 DO
  316.000     1   1         |     BEGIN
  317.000   1 1   1     13  |       FP_sh (.i.) := Dist_sh * Dir_no_sh (.i.) ;
  318.000   1 1   1     14  |       FP_ca (.i.) := Dist_ca * Dir_no_ca (.i.) ;
  319.000   1 1   1         |
  320.000   1 1   1     15  |       Center_sh (.i.) := ( Dist_sh - D_sh ) * Dir_no_sh (.i.) ;
  321.000   1 1   1     16  |       Center_ca (.i.) := ( Dist_ca - D_ca ) * Dir_no_ca (.i.)
  322.000     1             |     END ;
  323.000     1             |
  324.000     1             |  { initialize screen }
  325.000     1   1     17  |   FOR i := 1 TO Ny DO
  326.000     1   2     18  |       FOR j := 1 TO Nx DO
  327.000     1   2     19  |           Screen (. i, j .) := FALSE
  328.000     1   2         |
  329.000     1   2         |
```

LINE NUMBER B P C I STMT # SOURCE PROGRAM

```
V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
 END ;
```

LINE NUMBER B P C I   STMT #   S O U R C E   P R O G R A M

```
              V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
                  |
                  |
                  |/*
                  | *   Find_Tmat
                  | *
                  | *        This calculates T matrix by geometrical consideration
                  | */
                  |
 1                |PROCEDURE Find_Tmat ;
 1                |
 1                | VAR
 1                |   Pers_xform : ARRAY (. 1..4, 1..4 .) of REAL ;
 1                |                 { Perspective transformation matrix }
 1                |   SFx, SFy : REAL ;     { Scale Factors }
 1                |   x, y, z  : REAL ;     { focal point of the camera }
 1                |   i, j, k  : INTEGER ;  { indices   }
 1                |   temp : REAL ;         { temporary }
 1                |
 1                | BEGIN
 1                |
 1           1    |  SFx := D_ca / Sx ;
 1           2    |  SFy := D_ca / Sy ;
 1                |
 1           3    |  x := FP_ca (.1.) ;
 1           4    |  y := FP_ca (.2.) ;
 1           5    |  z := FP_ca (.3.) ;
 1                |
 1           6    |  View_xform (. 1, 1 .) := - SFx * SIN(Theta_ca) ;
 1           7    |  View_xform (. 2, 1 .) := SFx * COS(Theta_ca) ;
 1           8    |  View_xform (. 3, 1 .) := 0 ;
 1           9    |  View_xform (. 4, 1 .) := SFx * ( x * SIN ( Theta_ca ) - y * COS ( Theta_ca ) ) ;
 1                |
 1          10    |  View_xform (. 1, 2 .) := - SFy * COS ( Theta_ca ) * COS ( Phi_ca ) ;
 1          11    |  View_xform (. 2, 2 .) := - SFy * SIN ( Theta_ca ) * COS ( Phi_ca ) ;
 1          12    |  View_xform (. 3, 2 .) := SFy * SIN ( Phi_ca ) ;
 1          13    |  View_xform (. 4, 2 .) := SFy * ( - z * SIN(Phi_ca) + x * COS(Theta_ca) * COS(Phi_ca)
 1                |                            + y * SIN(Theta_ca) * COS(Phi_ca) ) ;
 1                |
 1          14    |  View_xform (. 1, 3 .) := - COS ( Theta_ca ) * SIN ( Phi_ca ) ;
 1          15    |  View_xform (. 2, 3 .) := - SIN ( Theta_ca ) * SIN ( Phi_ca ) ;
 1          16    |  View_xform (. 3, 3 .) := - COS ( Phi_ca ) ;
 1          17    |  View_xform (. 4, 3 .) := z * COS(Phi_ca) + x * COS(Theta_ca) * SIN(Phi_ca)
 1                |                            + y * SIN(Theta_ca) * SIN(Phi_ca) ;
 1                |
 1          18    |  View_xform (. 1, 4 .) := 0 ;
 1          19    |  View_xform (. 2, 4 .) := 0 ;
 1          20    |  View_xform (. 3, 4 .) := 0 ;
 1          21    |  View_xform (. 4, 4 .) := 1 ;
 1                |
 1                |
 1                | { Perspective transform }
 1                |
 1   1      22    |  FOR i := 1 TO 4 DO
```

```
LINE NUMBER B P C I  STMT #                     S O U R C E   P R O G R A M

                               V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
   386.000     1  2     23  |    FOR j := 1 TO 4 DO
   387.000     1        24  |        Pers_xform (. i, j .) := O ;
   388.000     1            |
   389.000     1        25  |  Pers_xform (. 1, 1 .) :=  Nx / 2 ;
   390.000     1        26  |  Pers_xform (. 2, 2 .) :=  Ny / 2 ;
   391.000     1        27  |  Pers_xform (. 3, 1 .) := ( 1 + Nx ) / 2 ;
   392.000     1        28  |  Pers_xform (. 3, 2 .) := ( 1 + Ny ) / 2 ;
   393.000     1        29  |  Pers_xform (. 3, 4 .) := 1 ;
   394.000     1            |
   395.000     1            |
   396.000     1            | { multiply View_xform by Pers_xform }
   397.000     1            |
   398.000     1        30  |  WRITELN ;  WRITELN ;
   399.000     1        32  |  WRITELN ( ' ---- T matrix ---- ' ) ;
   400.000     1  1     33  |  FOR i := 1 TO 4 DO
   401.000     1  1         |    BEGIN
   402.000     1  1         |
   403.000   1 1  2     34  |      FOR j := 1 TO 4 DO
   404.000   1 1  2         |        BEGIN
   405.000   2 1  2     35  |          temp := O ;
   406.000   2 1  3     36  |          FOR K := 1 TO 4 DO
   407.000   2 1  2     37  |              temp := temp + View_xform (. i, k :) * Pers_xform (. k, j .) ;
   408.000   2 1  2     38  |          Tmat (. i, j .):= temp ;
   409.000   2 1  2     39  |          WRITE ( Tmat (.i, j.) :wid :pre )
   410.000   1 1  1         |        END ; {for j}
   411.000   1 1  1     40  |      WRITELN ;
   412.000   1 1  1         |
   413.000     1            |    END ; {for i}
   414.000     1            |
   415.000     1        41  |    WRITELN ;
   416.000     1  1     42  |    FOR i := 1 TO 4 DO
   417.000     1  1         |        BEGIN
   418.000   1 1  2     43  |          FOR j := 1 TO 4 DO
   419.000   1 1  1     44  |             WRITE ( Tmat (.i, j.) / Tmat (.4, 4.) :wid :pre ) ;
   420.000   1 1  1         |          WRITELN
   421.000     1  1     45  |        END
   422.000     1  1         |
   423.000                  |  END ;
   424.000                  |
   425.000                  |
```

```
LINE NUMBER B P C I  STMT #                     S O U R C E   P R O G R A M

                        V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V

                         /*
                          *   Find_Lmat
                          *
                          *        This calculates L matrix from the geometrical consideration
                          */

            1            PROCEDURE Find_Lmat ;
            1
            1             VAR
            1               View_mat : ARRAY (. 1..4, 1..4 .) of REAL ;
            1                          { L matrix equivalent of View_xform of T matrix }
            1               Pers_mat : ARRAY (. 1..4, 1..4 .) of REAL ;
            1                          { Perspective transformation matrix of shutter }
            1               SFx, SFy : REAL ;     { Scale Factors }
            1               x, y, z  : REAL ;     { focal point }
            1               i, j, k  : INTEGER ; { indices }
            1               temp : REAL ;         { temporary }
            1
            1             BEGIN
            1
            1         1     SFx := D_sh / Su ;
            1         2     SFy := D_sh / Sv ;
            1
            1         3     x := FP_sh (.1.) ;
            1         4     y := FP_sh (.2.) ;
            1         5     z := FP_sh (.3.) ;
            1
            1         6     View_mat (. 1, 1 .) := - SFx * SIN(Theta_sh) ;
            1         7     View_mat (. 2, 1 .) := SFx * COS(Theta_sh) ;
            1         8     View_mat (. 3, 1 .) := 0 ;
            1         9     View_mat (. 4, 1 .) := SFx * ( x * SIN ( Theta_sh ) - y * COS ( Theta_sh ) ) ;
            1
            1        10     View_mat (. 1, 2 .) := - SFy * COS ( Theta_sh ) * COS ( Phi_sh ) ;
            1        11     View_mat (. 2, 2 .) := - SFy * SIN ( Theta_sh ) * COS ( Phi_sh ) ;
            1        12     View_mat (. 3, 2 .) := SFy * SIN ( Phi_sh ) ;
            1        13     View_mat (. 4, 2 .) := SFy * ( - z * SIN(Phi_sh) + x * COS(Theta_sh) * COS(Phi_sh)
            1                                         + y * SIN(Theta_sh) * COS(Phi_sh) ) ;
            1
            1        14     View_mat (. 1, 3 .) := - COS ( Theta_sh ) * SIN ( Phi_sh ) ;
            1        15     View_mat (. 2, 3 .) := - SIN ( Theta_sh ) * SIN ( Phi_sh ) ;
            1        16     View_mat (. 3, 3 .) := - COS ( Phi_sh ) ;
            1        17     View_mat (. 4, 3 .) := z * COS(Phi_sh) + x * COS(Theta_sh) * SIN(Phi_sh)
            1                                 + y * SIN(Theta_sh) * SIN(Phi_sh) ;
            1
            1        18     View_mat (. 1, 4 .) := 0 ;
            1        19     View_mat (. 2, 4 .) := 0 ;
            1        20     View_mat (. 3, 4 .) := 0 ;
            1        21     View_mat (. 4, 4 .) := 1 ;
            1
            1
            1
```

```
LINE NUMBER B P C I  STMT #                        S O U R C E   P R O G R A M

                                 V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
  479.000   1                   | { Perspective transform }
  480.000   1                   |
  481.000   1   1      22       |  FOR i := 1 TO 4 DO
  482.000   1   2      23       |     FOR j := 1 TO 4 DO
  483.000   1          24       |         Pers_mat (. i, j .) := 0 ;
  484.000   1                   |
  485.000   1          25       |  Pers_mat (. 1, 1 .) := Nu / 2 ;
  486.000   1          26       |  Pers_mat (. 2, 2 .) := Nv / 2 ;
  487.000   1          27       |  Pers_mat (. 3, 1 .) := ( 1 + Nu ) / 2 ;
  488.000   1          28       |  Pers_mat (. 3, 2 .) := ( 1 + Nv ) / 2 ;
  489.000   1          29       |  Pers_mat (. 3, 4 .) := 1 ;
  490.000   1                   |
  491.000   1                   |
  492.000   1                   | { multiply View_mat by Pers_mat }
  493.000   1                   |
  494.000   1          30       |  WRITELN ;  WRITELN ;
  495.000   1          32       |  WRITELN ( ' ---- L matrix ---- ' ) ;
  496.000   1   1      33       |  FOR i := 1 TO 4 DO
  497.000   1   1               |    BEGIN
  498.000   1   1               |
  499.000 1 1   2      34       |      FOR j := 1 TO 4 DO
  500.000 1 1   2               |        BEGIN
  501.000 2 1   2      35       |          temp := 0 ;
  502.000 2 1   3      36       |          FOR K := 1 TO 4 DO
  503.000 2 1   2      37       |              temp := temp + View_mat (. i, k .) * Pers_mat (. k, j .) ;
  504.000 2 1   2      38       |          Lmat (. i, j .):= temp ;
  505.000 2 1   2      39       |          WRITE ( Lmat (.i, j.) :wid :pre )
  506.000 1 1   1               |        END ; {for j}
  507.000 1 1   1               |
  508.000 1 1   1      40       |      WRITELN ;
  509.000 1 1   1               |
  510.000   1                   |    END ;{for i}
  511.000   1                   |
  512.000   1                   |
  513.000   1          41       |    WRITELN ;
  514.000   1   1      42       |    FOR i := 1 TO 4 DO
  515.000   1   1               |        BEGIN
  516.000 1 1   2      43       |          FOR j := 1 TO 4 DO
  517.000 1 1   1      44       |             WRITE ( Lmat (.i, j.) / Lmat (.4, 4.) :wid :pre ) ;
  518.000 1 1   1               |          WRITELN
  519.000   1   1      45       |        END
  520.000   1   1               |
  521.000   1   1               |
  522.000                       | END ;
  523.000                       |
```

LINE NUMBER B P C I STMT # S O U R C E P R O G R A M

```
V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
.

/*
 *   Find_Ray_Eqs
 *
 *       This calculates equations of the lines from the focal point
 *       of the camera to the sample points on the camera plane.
 *       This is used for verifing the T matrix calculated by
 *       geometrical consideration.
 */

PROCEDURE Find_Ray_Eqs ;

 VAR
    axe,
    aye : Vector ; { unit vectors in the camera plane }

    Camera : ARRAY (. 1..xs_max, 1..ys_max, 1..3 .) of REAL ; { coordinates of camera points }

    xs, ys, k : INTEGER ; { indices }

 BEGIN

  { find unit vectors in the camera plane }

   axe (.1.) := - SIN ( Theta_ca ) ;
   axe (.2.) := COS ( Theta_ca ) ;
   axe (.3.) := 0 ;

   aye (.1.) := - COS ( Phi_ca ) * COS ( Theta_ca ) ;
   aye (.2.) := - COS ( Phi_ca ) * SIN ( Theta_ca ) ;
   aye (.3.) := SIN ( Phi_ca ) ;


  { find the line eq of each ray }


  FOR xs := 1 TO Nx  DO
   FOR ys := 1 TO Ny  DO
     BEGIN

        FOR k := 1 TO 3  DO
           Camera (.xs,ys,k.) := Center_ca (.k.) + ( 2 * xs - Nx - 1 ) * (Sx / Nx) * axe(.k.)
                                  + ( 2 * ys - Ny - 1 ) * (Sy / Ny) * aye(.k.) ;

        FOR k := 1 TO 3 DO
          BEGIN
            Ray (.xs, ys, k.) :=  Camera (.xs, ys, k.) - FP_ca (.k.) ;
          END ;

     END

END ; { find_ray_eq }
```

```
LINE NUMBER B P C I  STMT #            S O U R C E   P R O G R A M
                                 V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
    577.000                      |
    578.000                      |
    579.000                      |
```

LINE NUMBER B P C I STMT # SOURCE PROGRAM

```
V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V


/*
 *   Find_Beam_Eqs
 *
 *
 *       This calculates equations of the lines from the focal point
 *       of the shutter to the sample points on the shutter plane.
 *       This is used for finding the intersection of each beam
 *       with the plane to be mapped.
 */

PROCEDURE Find_Beam_Eqs ;

 VAR
    axe,
    aye : Vector ; { unit vectors in the shutter plane }

    Shutter : ARRAY (. 1..u_max, 1..v_max, 1..3 .) of REAL ; { coordinates of shutter points }

    u, v, k : INTEGER ; { indices }

 BEGIN

  { find unit vectors in the shutter plane }

   axe (.1.) := - SIN ( Theta_sh ) ;
   axe (.2.) := COS ( Theta_sh ) ;
   axe (.3.) := 0 ;

   aye (.1.) := - COS ( Phi_sh ) * COS ( Theta_sh ) ;
   aye (.2.) := - COS ( Phi_sh ) * SIN ( Theta_sh ) ;
   aye (.3.) := SIN ( Phi_sh ) ;


  { find the line eq of each beam }


  FOR v := 1 TO Nv  DO
   FOR u := 1 TO Nu  DO
     BEGIN

        FOR k := 1 TO 3  DO
           Shutter (.u,v,k.) := Center_sh (.k.) + ( 2 * u - Nu - 1 ) * (Su / Nu) * axe(.k.)
                                + ( 2 * v - Nv - 1 ) * (Sv / Nv) * aye(.k.) ;

        FOR k := 1 TO 3 DO
           Beam (.u, v, k.) :=  Shutter (.u, v, k.) - FP_sh (.k.)

      END

END ; { find_beam_eq }
```

```
LINE NUMBER B P C I  STMT #                S O U R C E   P R O G R A M
                           V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
    633.000                |
```

LINE NUMBER B P C I STMT # SOURCE PROGRAM

```
        V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V


       /*
        *   Find_Beam_Planes
        *
        *       This calculates equations of the lines from the focal point
        *       of the shutter to the sample points on the shutter plane.
        *       This is used for verifing the L matrix calculated by
        *       geometrical consideration.
        */

       PROCEDURE Find_Beam_Planes ;

        VAR
          a1, b1, c1,          { direction numbers on column 1 }
          an, bn, cn : REAL ;  { direction numbers on column n }
          u : INTEGER ;        { index }

        BEGIN

   1      FOR u := 1 TO Nu DO
            BEGIN
   2          a1 := Beam (.u, 1, 1.) ;
   3          b1 := Beam (.u, 1, 2.) ;
   4          c1 := Beam (.u, 1, 3.) ;

   5          an := Beam (.u, Nv, 1.) ;
   6          bn := Beam (.u, Nv, 2.) ;
   7          cn := Beam (.u, Nv, 3.) ;

   8          Beam_Plane (.u, 1.) := b1 * cn - bn * c1 ; { A }
   9          Beam_Plane (.u, 2.) := c1 * an - cn * a1 ; { B }
  10          Beam_Plane (.u, 3.) := a1 * bn - an * b1 ; { C }
  11          Beam_Plane (.u, 4.) := - FP_sh(.1.) * Beam_plane (.u, 1.) - FP_sh(.2.) * Beam_plane (.u, 2.)
                                     - FP_sh(.3.) * Beam_plane (.u, 3.) ; { D }
            END

        END ;

```

LINE NUMBER B P C I STMT # S O U R C E P R O G R A M

```
                      V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+--1V


                      /*
                       *   Map_to_Camera
                       *
                       *        This calcuates the sample point on the camera plane
                       *        using the clip coordinates.  This is the standard
                       *        method used in graphics.
                       *        The output is checked against the output obtained
                       *        using T matrix and the output obtained by geometrical
                       *        consideration.
                       */

   1                  FUNCTION Map_to_Camera ( Pt : Vector ;
   1                                           VAR  Xs, Ys : INTEGER ) : BOOLEAN ;
   1                   VAR
   1                     Clip : Vector ; { clip coordinates }
   1
   1                     sum : REAL ;      { sum }
   1                     i, k : INTEGER ; { indices }
   1
   1                   BEGIN
   1
   1                    { find clip coordinates by multiplying View_xform matrix }
   1
   1   1         1         FOR i := 1 TO 3  DO
   1   1                       BEGIN
 1 1   1         2               sum := 0 ;
 1 1   2         3               FOR k := 1 TO 3  DO
 1 1   1         4                   sum := sum + Pt(.k.) * View_xform (.k, i.) ;
 1 1   1         5               Clip (.i.) := sum + View_xform (.4, i.)
   1                           END ;
   1
   1                    { clip and find camera coordinates }
   1 1           6      IF  ( - Clip(.3.) <= Clip(.1.) ) & ( Clip(.1.) <= Clip(.3.) ) &
   1 1                      ( - Clip(.3.) <= Clip(.2.) ) & ( Clip(.2.) <= Clip(.3.) )
   1 1                  THEN
   1 1                     BEGIN
 1 1 1           7              Xs := ROUND ( ( Clip(.1.) / Clip(.3.) ) * ( Nx / 2 ) + ( Nx / 2 + 0.5 ) ) ;
 1 1 1           8              Ys := ROUND ( ( Clip(.2.) / Clip(.3.) ) * ( Ny / 2 ) + ( Ny / 2 + 0.5 ) ) ;
 1 1 1           9              Map_to_Camera := TRUE
   1 1                     END
   1 1
   1 1                  ELSE  BEGIN
 1 1 1          10              Map_to_camera := FALSE ;
 1 1 1          11              Xs := unobserved ;
 1 1 1          12              Ys := unobserved
   1 1                        END
   1 1
                       END ;

```

```
LINE NUMBER B P C I  STMT #                          S O U R C E   P R O G R A M

                             V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
  725.000                   |
  726.000                   |
  727.000                   |/*
  728.000                   | *   Map_to_Cam_Tmat
  729.000                   | *
  730.000                   | *       This finds the sample points on the camera plane
  731.000                   | *       using the T matrix.
  732.000                   | */
  733.000                   |
  734.000   1               |FUNCTION Map_to_Cam_Tmat ( Pt : Vector ;
  735.000   1               |                           VAR  Xs, Ys : INTEGER ) : BOOLEAN ;
  736.000   1               | VAR
  737.000   1               |   Homo : ARRAY(.1..4.) of REAL ; { homogeneous coordinates }
  738.000   1               |
  739.000   1               |   sum : REAL ;       { sum }
  740.000   1               |   i, k : INTEGER ;   { indices }
  741.000   1               |
  742.000   1               | BEGIN
  743.000   1               |
  744.000   1               |  { find homogeneous coordinates by multiplying Tmat matrix }
  745.000   1               |
  746.000   1   1        1  |   FOR i := 1 TO 4  DO
  747.000   1   1           |       BEGIN
  748.000 1 1   1        2  |         sum := 0 ;
  749.000 1 1   2        3  |         FOR k := 1 TO 3  DO
  750.000 1 1   1        4  |             sum := sum + Pt(.k.) * Tmat (.k, i.) ;
  751.000 1 1   1        5  |         Homo (.i.) := sum + Tmat (.4, i.)
  752.000   1               |       END ;
  753.000   1               |
  754.000   1               |   { find screen coordinates }
  755.000   1               |
  756.000   1              6|     Xs := ROUND ( Homo(.1.) / Homo(.4.) ) ;
  757.000   1              7|     Ys := ROUND ( Homo(.2.) / Homo(.4.) ) ;
  758.000   1               |
  759.000   1               |   { clip }
  760.000   1               |
  761.000   1 1            8|     IF  ( 1 <= Xs ) & ( Xs <= Nx ) &
  762.000   1 1             |         ( 1 <= Ys ) & ( Ys <= Ny )
  763.000   1 1            9|     THEN  Map_to_Cam_Tmat := TRUE
  764.000   1 1             |     ELSE BEGIN
  765.000 1 1 1           10|            Map_to_cam_Tmat := FALSE ;
  766.000 1 1 1           11|            Xs := unobserved ;
  767.000 1 1 1           12|            Ys := unobserved
  768.000   1 1             |          END
  769.000   1 1             |
  770.000                   | END ;
  771.000                   |
  772.000                   |
  773.000                   |
```

LINE NUMBER B P C I STMT # S O U R C E P R O G R A M

```
       V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V

       /*
        *   Map_to_Cam_Geo
        *
        *           This calculates sample points on the camera plane
        *           by the geometrical consideration.
        */

       FUNCTION Map_to_Cam_Geo ( Pt : Vector ;
                                 VAR  Xs, Ys : INTEGER ) : BOOLEAN ;
        VAR
            Ap, Bp, Cp, Dp,          { equation of the plane }
            al, bl, cl :    REAL ; { equaiton of line }

            axe, aye : Vector ;     { unit vectors on the camera plane }

            Cam_pt : Vector ;       { intersection of the camera plane and the line }

            S1, S2, t : REAL ;      { sums, temporary }
            i : INTEGER ;           { index }

        BEGIN

         { equation of camera plane : Ap x + Bp y + Cp z + Dp = 0 }

 1          Ap := Dir_no_ca (.1.) ;
 2          Bp := Dir_no_ca (.2.) ;
 3          Cp := Dir_no_ca (.3.) ;
 4          Dp := - ( Ap * Center_ca (.1.) + Bp * Center_ca (.2.) + Cp * Center_ca (.3.) ) ;

         { direction numbers of a line from the point to the focal point of a camera }

 5          al := Pt (.1.) - FP_ca (.1.) ;
 6          bl := Pt (.2.) - FP_ca (.2.) ;
 7          cl := Pt (.3.) - FP_ca (.3.) ;

         { intersection of a camera plane and the line }

 8          t := - ( Ap * FP_ca(.1.) + Bp * FP_ca(.2.) + Cp * FP_ca(.3.) + Dp ) /
                   ( Ap * al + Bp * bl + Cp * cl ) ;

 9          Cam_pt (.1.) := FP_ca (.1.) + t * al ;
10          Cam_pt (.2.) := FP_ca (.2.) + t * bl ;
11          Cam_pt (.3.) := FP_ca (.3.) + t * cl ;

         { find unit vectors in the camera plane }

12          axe (.1.) := - SIN ( Theta_ca ) ;
13          axe (.2.) := COS ( Theta_ca ) ;
14          axe (.3.) := 0 ;

15          aye (.1.) := - COS ( Phi_ca ) * COS ( Theta_ca ) ;
```

```
B P C I   STMT #                    S O U R C E   P R O G R A M

                  V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
  1         16  |    aye (.2.) := - COS ( Phi_ca ) * SIN ( Theta_ca ) ;
  1         17  |    aye (.3.) := SIN ( Phi_ca ) ;
  1             |
  1             |  { find camera coordintes for the world coordinate point Cam_pt }
  1             |
  1         18  |   S1 := O ;  S2 := O ;
  1   1     20  |   FOR i := 1 TO 3  DO
  1   1         |      BEGIN
1 1   1     21  |         S1 := S1 + ( Cam_pt(.i.) - Center_ca(.i.) ) * axe (.i.) ;
1 1   1     22  |         S2 := S2 + ( Cam_pt(.i.) - Center_ca(.i.) ) * aye (.i.)
  1             |      END ;
  1             |
  1         23  |   Xs := ROUND ( ( S1 * ( Nx / Sx ) + 1 + Nx ) / 2 ) ;
  1         24  |   Ys := - ROUND ( ( - S2 * ( Ny / Sy ) + 1 + Ny ) / 2 ) + Ny + 1 ;
  1             |
  1             |  { Clip }
  1             |
  1 1       25  |  IF ( 1 <= Xs ) & ( Xs <= Nx ) &
  1 1           |     ( 1 <= YS ) & ( Ys <= Ny )
  1 1       26  |     THEN  Map_to_Cam_Geo := TRUE
  1 1           |     ELSE BEGIN
1 1 1       27  |            Map_to_Cam_Geo := FALSE ;
1 1 1       28  |            Xs := unobserved ;
1 1 1       29  |            Ys := unobserved
  1 1           |          END
  1 1           |
                |END ;
                |
```

LINE NUMBER B P C I STMT # S O U R C E P R O G R A M

```
V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V


/*
 *   Make_Image_pattern
 *
 *       This makes a pattern of images to be passed to the
 *       Surface Mapping Program.
 *       If two different points mapped into the same point
 *       on the camera, Surface Mapping Program generates a
 *       spurious output.  This routine delete all such points
 *       from the pattern and flags that point.
 */

PROCEDURE Make_Image_Pattern ;

 VAR
    Intersect : Vector ;   { intersection of the plane and the beam of the shutter }

    A, B, C, D : REAL ; { Ax + By + Cz + D = O }

    Xs1, Ys1,
    Xs2, Ys2,
    Xs3, Ys3 : INTEGER ; { Screen coordinates }

    t : REAL ;            { temporary }
    dummy : BOOLEAN ;     { dummy }
    u, v, k : INTEGER ;   { indices }

 BEGIN

  A := Map_plane (.1.) ;
  B := Map_plane (.2.) ;
  C := Map_plane (.3.) ;
  D := Map_plane (.4.) ;

  FOR u :=.1 TO Nu  DO
     FOR v := 1 TO Nv  DO

        BEGIN
          t := - ( A * FP_sh(.1.) + B * FP_sh(.2.) + C * FP_sh(.3.) + D ) /
                 ( A * Beam(.u, v, 1.) + B * Beam(.u, v, 2.) + C * Beam(.u, v, 3.) ) ;

          FOR k := 1 TO 3 DO
              Intersect (.k.) := FP_sh (.k.) + t * Beam (.u, v, k.) ;

          IF Map_to_Camera ( Intersect, Xs1, Ys1 ) = TRUE
             THEN BEGIN
                    IF Screen (.Xs1, Ys1.) = FALSE
                       THEN BEGIN
                              WRITELN ( ana_f, u :lenf, v :len, Xs1:len, Ys1:len, Intersect(.1.) :wid
                                        :pre, Intersect(.2.) :wid :pre, Intersect(.3.) :wid :pre ) ;

                              Screen (.Xs1, Ys1.) := TRUE ;
```

```
LINE NUMBER B P C I  STMT #                         S O U R C E   P R O G R A M

                               V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
  908.000 3 1 2 2      14     |                         Pattern (.u, v, 1.) := Xs1 ;
  909.000 3 1 2 2      15     |                         Pattern (.u, v, 2.) := Ys1
  910.000 2 1 2 2             |                       END
  911.000 2 1 2 2             |                  ELSE BEGIN
  912.000 2 1 2 2             |                   { Different beams map into same camera point }
  913.000 3 1 2 2      16     |                         Pattern (.u, v, 1.) := unobserved ;
  914.000 3 1 2 2      17     |                         WRITELN ( ' ***** Duplicate points--- ', u :len, v :len )
  915.000 2 1 2 2             |                       END
  916.000 1 1 1 2             |             END
  917.000 1 1 1 2             |
  918.000 1 1 1 2             |       ELSE BEGIN
  919.000 2 1 1 2      18     |               Pattern (.u, v, 1.) := unobserved ;
  920.000 2 1 1 2      19     |               WRITELN ( ' VMAT *** not observed *** ', u, v )
  921.000 1 1   2             |            END ;
  922.000 1 1   2             |
  923.000 1 1   2             |      { comparison of 3 methods }
  924.000 1 1   2             |
  925.000 1 1   2      20     |         dummy := Map_to_Cam_Geo ( Intersect, Xs2, YS2 ) ;
  926.000 1 1   2      21     |         dummy := Map_to_Cam_Tmat ( Intersect, Xs3, YS3 ) ;
  927.000 1 1   2             |
  928.000 1 1 1 2      22     |         If (Xs1 <> Xs2) | (Xs2 <> Xs3) | (Ys1 <> Ys2) | (Ys2 <> Ys3)
  929.000 1 1 1 2             |            THEN BEGIN
  930.000 2 1 1 2      23     |               WRITE ( '##### Inconsistency ', u :len, v :len );
  931.000 2 1 1 2      24     |               WRITELN ( ' VMAT ', Xs1 :len, Ys1 :len ) ;
  932.000 2 1 1 2      25     |               WRITELN ( ' GEO  ', Xs2 :len, Ys2 :len ) ;
  933.000 2 1 1 2      26     |               WRITELN ( ' Tmat ', Xs3 :len, Ys3 :len )
  934.000 1 1 1 2             |             END
  935.000 1 1 1 2             |
  936.000   1   2             |       END {for u,v}
  937.000   1   2             |
  938.000                     |END ;
  939.000                     |
```

LINE NUMBER B P C I STMT # S O U R C E P R O G R A M

```
V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V

/*
 *   Output_pattern
 *
 *       ++ HERE ++   Nu is a power of 2
 *
 *            This generates patterns of images to be used
 *            by the Surface Mapping Program.
 */

PROCEDURE Output_Pattern ;

 VAR
   No_pattern,               { Number of patterns }
   Column,                   { column on the shutter plane }
   Run,                      { run run run .........       }
   Pat        : INTEGER ;    { This goes 1, 2, 4, 8, 16, ... }

   Mask_val   : BOOLEAN ;    { shutter is open or close ? }
   i, j, k, v : INTEGER ;    { indices }

 BEGIN

   No_pattern := ROUND ( LN ( Nu ) / LN ( 2 ) ) + 1 ;

   Pat := 1 ;    { This goes  1, 2, 4, 8, 16, ... }

   FOR i := 1 TO No_pattern  DO
     BEGIN

       Column := Nu + 1 ;
       Mask_val := TRUE ;    { FALSE --- Closed shutter }
                             { TRUE  --- Open shutter }
       Run := Nu DIV Pat ;

       FOR j := 1 TO Pat  DO
          BEGIN

            FOR k := 1 TO Run  DO
               BEGIN
                 Column := Column - 1 ;

                 IF Mask_val = True  { open }
                 THEN
                    FOR v := 1 TO Nv DO
                        IF Pattern (.Column, v, 1.) ¬= unobserved
                        THEN  WRITELN ( Img_f, Pattern (.Column, v, 1.) :lenf,
                                        Pattern (.Column, v, 2.) :len ) ;
               END ; {for k}

           Mask_val := NOT Mask_val
```

```
LINE NUMBER B P C I   STMT #                       S O U R C E   P R O G R A M

                                V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
    993.000 1 1   1            |         END ; {for j}
    994.000 1 1   1            |
    995.000 1 1   1       15   |      Pat := 2 * Pat ;
    996.000 1 1   1            |
    997.000 1 1   1     . 16   |   .  WRITELN ( Img_f, -1 :lenf ,  -1 :len )
    998.000 1 1   1            |
    999.000   1   1            |    END {for i}
   1000.000   1   1            |
   1001.000                    | END ;
   1002.000                    |
   1003.000                    |
   1004.000                    |
```

LINE NUMBER B P C I STMT # SOURCE PROGRAM

```
                 V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V


                 /*
                  *   Read_Cal_Para
                  *
                  *          This reads parameters about the calibration cube.
                  */

1                PROCEDURE Read_Cal_Para ;
1
1                 VAR
1
1                   R_mat,              { Rotation matrix          }
1                   Rot,                { temporary rotation matrix  }
1                   T_inv,{ inverse translation matrix }
1                   Temp_mat : mat_4 ;  { temporary matrix           }
1
1                   Axis : CHAR ;       { axis of rotation of the cube }
1                   Angle,              { amount of rotation of the cube in degree }
1                   Length      : REAL ; { length of the side of the cube }
1
1                   i, j : INTEGER ;    { indices }
1
1
1                 BEGIN
1
1            1        WRITELN ;
1            2        WRITELN ( ' ------ Enter Calibration Parameters ' ) ;
1            3        WRITELN ;
1
1            4        WRITELN ( '  length of the side of a cube -- Length ' ) ;
1            5        READLN ( Length ) ;
1
1            6        WRITELN ( '  Step in u and v  -- Step ' ) ;
1            7        READLN ( Step ) ;
1
1            8        a := Length / 2 ;
1
1                  { Rotation of the cube about x , y, and z axis }
1
1            9        WRITELN ( ' ---- rotation of the cube ---- ' ) ;
1           10        WRITELN ;
1
1           11        WRITELN ( ' Enter axis ( x, y, z, or / ) and angle ' ) ;
1           12        READLN ( Axis, Angle ) ;
1           13        Angle := Angle * 3.141592 / 180 ;
1
1                   { Find rotation matrix R_mat }
1
1           14       Make_Identity ( R_mat ) ;
1
1   1       15       WHILE  ( Axis = 'x' ) | ( Axis = 'y' ) | ( Axis = 'z' )  DO
1   1                  BFGIN
```

```
LINE NUMBER B P C I  STMT #                       S O U R C E   P R O G R A M

                                 V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
  1058.000   1   1             |
  1059.000 1 1   1        16   |       Make_Identity ( Rot ) ;
  1060.000 1 1   1             |
  1061.000 1 1 1 1        17   |       CASE Axis OF
  1062.000 1 1 1 1             |
  1063.000 1 1 1 1             |            'x' : BEGIN
  1064.000 2 1 1 1        18   |                    Rot(.2, 2.) := COS (Angle) ;
  1065 000 2 1 1 1        19   |                    Rot(.2, 3.) := - SIN (Angle) ;
  1066.000 2 1 1 1        20   |                    Rot(.3, 2.) := SIN (Angle) ;
  1067.000 2 1 1 1        21   |                    Rot(.3, 3.) := COS (Angle)
  1068.000 1 1 1 1             |                  END ;
  1069.000 1 1 1 1             |
  1070.000 1 1 1 1             |            'y' : BEGIN
  1071.000 2 1 1 1        22   |                    Rot(.1, 1.) := COS (Angle) ;
  1072.000 2 1 1 1        23   |                    Rot(.1, 3.) := SIN (Angle) ;
  1073.000 2 1 1 1        24   |                    Rot(.3, 1.) := - SIN (Angle) ;
  1074.000 2 1 1 1        25   |                    Rot(.3, 3.) := COS (Angle)
  1075.000 1 1 1 1             |                  END ;
  1076.000 1 1 1 1             |
  1077.000 1 1 1 1             |            'z' : BEGIN
  1078.000 2 1 1 1        26   |                    Rot(.1, 1.) := COS (Angle) ;
  1079.000 2 1 1 1        27   |                    Rot(.1, 2.) := - SIN (Angle) ;
  1080.000 2 1 1 1        28   |                    Rot(.2, 1.) := SIN (Angle) ;
  1081.000 2 1 1 1        29   |                    Rot(.2, 2.) := COS (Angle)
  1082.000 1 1 1 1             |                  END
  1083.000 1 1 1 1             |
  1084.000 1 1   1             |       END ;{case}
  1085.000 1 1   1             |
  1086.000 1 1   1        30   |       Temp_mat := R_mat ;
  1087.000 1 1   1        31   |       Mult_Mat4 ( R_mat, Temp_mat, Rot ) ;
  1088.000 1 1   1             |
  1089.000 1 1   1        32   |       WRITELN ( ' Enter axis ( x, y, z, or / ) and angle ' ) ;
  1090.000 1 1   1        33   |       READLN ( Axis, Angle ) ;
  1092.000 1 1   1        34   |       Angle := Angle * 3.141592 /180
  1093.000 1 1   1             |
  1094.000   1                 |     END ; {while}
  1095.000   1                 |
  1096.000   1                 |
  1097.000   1                 | { Translation of the cube }
  1098.000   1                 |
  1099.000   1            35   |    WRITELN ( ' ---- Translation of the cube ---- ' ) ;
  1100.000   1            36   |    WRITELN ;
  1101.000   1                 |
  1102.000   1            37   |    WRITELN ( ' Enter translation along x, y, and z axis ' ) ;
  1103.000   1            38   |    WRITELN ( '     e.g.  3.8  10  8.5 ') ;
  1104.000   1            39   |    READLN ( xt, yt, zt ) ;
  1105.000   1                 |
  1106.000   1                 |
  1107.000   1            40   |   Make_Identity ( T_inv ) ;
  1108.000   1                 |
  1109.000   1            41   |   T_inv (.4, 1.) := - xt ;
  1110.000   1            42   |   T_inv (.4, 2.) := - yt ;
  1111.000   1            43   |   T_inv (.4, 3.) := - zt ;
```

```
 LINE NUMBER B P C I  STMT #                          S O U R C E   P R O G R A M

                              V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
    1112.000    1
    1113.000    1              { find H_mat and inverse H_inv }
    1114.000    1
    1115.000    1         44      H_mat := R_mat ;
    1116.000    1         45      H_mat (.4, 1.) := xt ;
    1117.000    1         46      H_mat (.4, 2.) := yt ;
    1118.000    1         47      H_mat (.4, 3.) := zt ;
    1119.000    1
    1120.000    1   1     48      FOR i := 1 TO 4 DO
    1121.000    1   2     49         FOR J := 1 TO 4 DO
    1122.000    1         50             Temp_mat (.i, j.) := R_mat (.j, i.) ; { transpose }
    1123.000    1
    1124.000    1         51      Mult_Mat4 ( H_inv, T_inv, Temp_mat ) ;
    1125.000    1
    1126.000                   END ;
```

LINE NUMBER B P C I   STMT #   S O U R C E   P R O G R A M

```
V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V


/*
 *   Find_Cal_Face
 *
 *       This finds the equation the faces of the cube.
 *       Then this test if the face is visible by shutter and
 *       camera, and output the number of faces visible by shutter
 *       and camera.
 */

PROCEDURE Find_Cal_Face ;

 VAR

   V,                                     { vertices of the cube }
   Vt : ARRAY (.1..8, 1..4.) of REAL ;    { moved vertices of the cube}
   F,                                     { faces of the cube }
   Ft : ARRAY (.1..6, 1..4.) of REAL ;    { moved faces of the cube }

   sign1, sign2, sign3 : REAL ;   { signs }

   i, j, k, n : INTEGER ;         { indices }

 BEGIN

  { position of the vertices before the cube is moved }

    V(.1, 1.) :=   a ; V(.1, 2.) :=   a ; V(.1, 3.) :=   a ; V(.1, 4.) := 1 ;
    V(.2, 1.) :=   a ; V(.2, 2.) :=   a ; V(.2, 3.) := - a ; V(.2, 4.) := 1 ;
    V(.3, 1.) :=   a ; V(.3, 2.) := - a ; V(.3, 3.) := - a ; V(.3, 4.) := 1 ;
    V(.4, 1.) :=   a ; V(.4, 2.) := - a ; V(.4, 3.) :=   a ; V(.4, 4.) := 1 ;
    V(.5, 1.) := - a ; V(.5, 2.) :=   a ; V(.5, 3.) :=   a ; V(.5, 4.) := 1 ;
    V(.6, 1.) := - a ; V(.6, 2.) :=   a ; V(.6, 3.) := - a ; V(.6, 4.) := 1 ;
    V(.7, 1.) := - a ; V(.7, 2.) := - a ; V(.7, 3.) := - a ; V(.7, 4.) := 1 ;
    V(.8, 1.) := - a ; V(.8, 2.) := - a ; V(.8, 3.) :=   a ; V(.8, 4.) := 1 ;

 { equation of faces of a cube before the cube is moved }

    F(.1, 1.) := 0   ; F(.1, 2.) := 0   ; F(.1, 3.) := 1   ; F(.1, 4.) := - a ;
    F(.2, 1.) := - 1 ; F(.2, 2.) := 0   ; F(.2, 3.) := 0   ; F(.2, 4.) := - a ;
    F(.3, 1.) := 0   ; F(.3, 2.) := 0   ; F(.3, 3.) := - 1 ; F(.3, 4.) := - a ;
    F(.4, 1.) := 1   ; F(.4, 2.) := 0   ; F(.4, 3.) := 0   ; F(.4, 4.) := - a ;
    F(.5, 1.) := 0   ; F(.5, 2.) := - 1 ; F(.5, 3.) := 0   ; F(.5, 4.) := - a ;
    F(.6, 1.) := 0   ; F(.6, 2.) := 1   ; F(.6, 3.) := 0   ; F(.6, 4.) := - a ;

  { Move Vertices to new position -- Vt(.1, 4.) is 1 }

  FOR i := 1 TO 8 DO
    FOR j := 1 TO 4 DO
      BEGIN
        Vt(.i, j.) := 0 ;
        FOR K := 1 TO 4 DO
```

```
B P C I   STMT #                  S O U R C E   P R O G R A M

                   V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
1 1   3      61  |          Vt(.i, j.) := Vt(.i, j.) + V(.i, k.) * H_mat(.k, j.)
  1              |       END ;
  1              |
  1              |   { Move Faces to new position }
  1              |
  1   1      62  |   FOR i := 1 TO 6 DO
  1   2      63  |     FOR j := 1 TO 4 DO
  1   2          |       BEGIN
1 1   2      64  |         Ft(.i, j.) := 0 ;
1 1   3      65  |         FOR K := 1 TO 4 DO
1 1   3      66  |             Ft(.i, j.) := Ft(.i, j.) + H_inv(.j, k.) * F(.i, k.)
  1              |       END ;
  1              |
  1              |
  1              |   { find faces which are visible by both shutter and camera }
  1              |
  1          67  |     n := 0 ;
  1              |
  1   1      68  |     FOR i := 1 TO 6 DO
  1   1          |       BEGIN
  1   1          |
  1   1          |        { By the sign of sign1, we know at which side of the face focal point of shutter is }
  1   1          |
1 1   1      69  |         sign1 := 0 ;
1 1   2      70  |         FOR k := 1 TO 3 DO
1 1   1      71  |             sign1 := sign1 + FP_ca(.k.) * Ft(.i, k.) ;
1 1   1      72  |         sign1 := sign1 + Ft(.i, 4.) ;
1 1   1          |
1 1   1      73  |         sign2 := 0 ;
1 1   2      74  |         FOR k := 1 TO 3 DO
1 1   1      75  |             sign2 := sign2 + FP_sh(.k.) * Ft(.i, k.) ;
1 1   1      76  |         sign2 := sign2 + Ft(.i, 4.) ;
1 1   1          |
1 1   1      77  |         sign3 := 0 ;
1 1   2      78  |         FOR K := 1 TO 4 DO
1 1   1      79  |             sign3 := sign3 + Vt(.i+1, k.) * Ft(.i, k.) ;
1 1   1          |
1 1   1          |        { focal points of shutter and camera should be at the same side of the face
1 1   1          |        { and they should be at the opposite side of the face to a vertex selected }
1 1   1          |
1 1 1 1      80  |         IF ( ( sign1 < 0 ) & ( sign2 < 0 ) & ( sign3 > 0 ) ) |
1 1 1 1          |            ( ( sign1 > 0 ) & ( sign2 > 0 ) & ( sign3 < 0 ) )  THEN
1 1 1 1          |            BEGIN
2 1 1 1      81  |              n := n + 1 ;
2 1 1 2      82  |              FOR k := 1 TO 4 DO
2 1 1 2      83  |                  Face (.n, k.) := Ft(.i, k.)
1 1 1 1          |            END ;
1 1 1 1          |
  1              |      END ;{for}
  1          84  |      Num_face := n ;
  1              |
  1          85  |    WRITELN ;
  1          86  |    WRITELN ( ' Num face ', Num face )
```

```
LINE NUMBER B P C I  STMT #                  S O U R C E   P R O G R A M
                               V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
   1233.000   1              |    |
   1234.000                  | END ;
   1235.000                  |
   1236.000                  |
```

LINE NUMBER B P C I STMT # S O U R C E P R O G R A M

```
V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V


/*
 *   Output_Cube_Image
 *
 *          This outputs sample points of the cube on the
 *          camera plane.
 */

PROCEDURE Output_Cube_Image ;

 VAR

   A, B, C, D,              { equation of the faces }
   bound,                   { square of length from the center of the cube to the vertex }
   dist,                    { distance }
   t           : REAL ;     { temporary }

   Intersect : Vector ;     { intersection of the face and the beam of the shutter }

   u, v, i, k,
   Xs, Ys      : INTEGER ; { indices }

 BEGIN

  IF Num_face <= 0 THEN
     BEGIN
        WRITELN ( ' ***** Calibration cube is not visible ***** ' );
        RETURN
     END ;

  bound := 3 * a * a ;

  u := 1 ;
  WHILE ( u <= Nu ) DO
    BEGIN
      v := 1 ;
      WHILE v <= Nv DO
        BEGIN

          FOR i := 1 TO Num_face DO
            BEGIN

              { find intersection of beam(.u, v, *.) and Face }

                A := Face (.i, 1.) ;
                B := Face (.i, 2.) ;
                C := Face (.i, 3.) ;
                D := Face (.i, 4.) ;

                t := - ( A * FP_sh(.1.) + B * FP_sh(.2.) + C * FP_sh(.3.) + D ) /
                       ( A * Beam(.u, v, 1.) + B * Beam(.u, v, 2.) + C * Beam(.u, v, 3.) ) ;

```

```
LINE NUMBER B P C I  STMT #                         S O U R C E   P R O G R A M

                                  V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
   1290.000 3 1   4     15        |               FOR k := 1 TO 3 DO
   1291.000 3 1   3     16        |                   Intersect (.k.) := FP_sh (.k.) + t * Beam (.u, v, k.) ;
   1292.000 3 1   3               |
   1293.000 3 1   3     17        |               dist := SQR( Intersect(.1.) - xt ) + SQR( Intersect(.2.) - yt ) +
   1294.000 3 1   3               |                   SQR( Intersect(.3.) - zt ) ;
   1295.000 3 1   3               |
   1296.000 3 1 1 3     18        |               IF  ( dist <=  bound ) THEN  { inside cube }
   1297.000 4 1 2 3     19        |                  BEGIN IF Map_to_Camera ( Intersect, Xs, Ys ) = TRUE
   1298.000 4 1 2 3               |                          THEN BEGIN
   1299.000 5 1 2 3     20        |                                 WRITELN ( cal_f, Intersect(.1.) :widf :3, Intersect(.2.):wid:3,
   1300.000 5 1 2 3               |                                           Intersect(.3.) :wid :3 ) ;
   1301.000 5 1 2 3               |
   1302.000 5 1 2 3     21        |                                 WRITELN ( cal_f, u :lenf, Xs :len, Ys :len ) ;
   1303.000 4 1 2 3               |                               END
   1304.000 3 1 1 3               |                  END
   1305.000 3 1 1 3               |
   1306.000 2 1   2               |            END ;{for}
   1307.000 2 1   2               |
   1308.000 2 1   2     22        |          v := v + step
   1309.000 1 1   1               |        END ;{while v}
   1310.000 1 1   1               |
   1311.000 1 1.  1     23        |      u := u + step
   1312.000   1   1               |    END {while u}
   1313.000   1   1               |
   1314.000                       | END ;
   1315.000                       |
   1316.000                       |
   1317.000                       |
```

```
LINE NUMBER B P C I  STMT #                    S O U R C E   P R O G R A M
                     V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V


                     /*********************
                      *
                      *       MAIN
                      *
                      *********************
                      *
                      */


                      BEGIN

                        { Open files }

                1         RESET ( INPUT, 'UNIT=SCARDS, INTERACTIVE' ) ;
                2         REWRITE ( Cal_f, 'UNIT=3' ) ;
                3         REWRITE ( Img_f, 'UNIT=4' ) ;
                4         REWRITE ( Ana_f, 'UNIT=2' ) ;


                5         Read_Parameters ;
                6         Read_Cal_Para ;
                7         Initialize ;

                8         Find_Tmat ;
                9         Find_Lmat ;

               10         Find_Ray_Eqs ;
               11         Find_Beam_Eqs ;
               12         Find_Beam_Planes ;

               13         Make_Image_Pattern ;
               14         Output_Pattern ;

               15         Find_Cal_Face ;
                          Output_Cube_Image

                      END.

```

NO COMPILER DETECTED ERRORS

OPTIONS IN EFFECT: MARGINS(1,100), NOSEQUENCE, LANGLVL(EXTENDED), LINECOUNT(60), PAGEWIDTH(132), CHECK,
DEBUG, GOSTMT, SOURCE, WARNING

SOURCE LINES: 1356; COMPILE TIME: 1.16 SECONDS; COMPILE RATE: 70004 LPM

SOURCE LINES: 1356; TRANSLATE TIME: 0.71 SECONDS; TRANSLATE RATE: 115277 LPM

TOTAL TIME: 2.33 SECONDS; TOTAL RATE: 34963 LPM

8.2. Eye.Cal Program

```
          C
          C      Mapping Program : Eye.Cal
          C
          C
          C
          C      Version : This is based on the program of Dr. Martin Altschuler
          C                Calibration of T matrix and L matrix are done
          C                by least square method
          C
          C
          C
          C
          C
          C  DIRECTORY
          C     SROW       # OF ROWS OF CONTOUR GRID
          C     SCOL       # OF COLUMNS OF CONTOUR GRID
          C     CROW       # OF ROWS IN ENTIRE RASTER SCREEN
          C     CCOL       # OF COLUMNS IN ENTIRE RASTER SCREEN
          C     NIMAGE       # OF INAGES NEEDED TO CODE SCENE
          C
          C     LUCAL        LOGICAL UNIT FOR CALIBRATION INFO
          C     LUDAT        LOGICAL UNIT FOR INPUT DATA
          C     LUOUT        LOGICAL UNIT FOR FILE OUTPUT
          C     LUTERM       LOGICAL UNIT FOR TERMINAL OUTPUT
          C
          C     PAT1       MATRIX OF RASTER IMAGE # 1
          C       :
          C       :
          C     PAT7       MATRIX OF RASTER IMAGE # 7
          C
          C
          C  LOGICAL DEVICES
          C
          C
          C     LOGICAL UNIT #3   CALIBRATION INFO
          C     LOGICAL UNIT #4   INPUT FILE
          C     LOGICAL UNIT #5   OUTPUT FILE
          C     LOGICAL UNIT #6   TERMINAL
          C
          C
          C------------------------------------------------------------
          C
          C             Main program
          C
          C------------------------------------------------------------
          C
          C   Local variables
          C
          C   Transformation matrices
          C
 0001           REAL T(4,4), L(4,4)
          C
          C   Calibration cube data
          C
```

```
0003              INTEGER CUBIMG(100,2), CUBCOL(100)
            C  Begin
            C
            C   Acquire and check the initial parameters.
0004              CALL INIT ( CUBSUR, CUBIMG, CUBCOL )
            C
            C   Calibrate the system.
0005              CALL TMAT( CUBSUR, CUBIMG, T )
0006              CALL LMAT( CUBSUR, CUBCOL, L )
            C
            C   Solve for the unknown surface.
0007              CALL IMGIN
0008              CALL MAP( T, L )
            C
0009              STOP
0010              END
 *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
 *OPTIONS IN EFFECT*  NAME = MAIN    , LINECNT =       57
 *STATISTICS*    SOURCE STATEMENTS =        10,PROGRAM SIZE =     2916
 *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
        C
        C-------------------------------------------------------
        C
0001          SUBROUTINE INIT ( CUBSR, CUBIM, CUBCO )
        C
        C-------------------------------------------------------
        C
        C  include GLOBAL and IO common
        C
        C  "GLOBAL",common (100,120)
        C
0002          INTEGER SROW, SCOL
        C       Dimensions of the shutter grid
        C
0003          INTEGER CROW, CCOL
        C       Dimensions of the camera grid
        C
0004          INTEGER NIMAGE, NUMCAL
        C       Number of patterns needed
        C       Number of calibration data
        C
0005          COMMON/GLOBAL/
            1     SROW, SCOL, CROW, CCOL, NIMAGE, NUMCAL
        C
        C  "IO" common (200, 210 )
0006          INTEGER LUCAL, LUDAT, LUOUT, LUTERM
        C        Logical units for the calibration file, data, output,
        C         and the terminal.
        C
0007          COMMON/IO/
            1     LUCAL, LUDAT, LUOUT, LUTERM
        C
        C
        C
        C parameters -- calibration cube data
        C
0008          REAL CUBSR(100, 3)
0009          INTEGER CUBIM(100, 2), CUBCO(100)
        C
        C  Begin initialization
        C
0010          LUCAL =  3
0011          LUDAT =  4
0012          LUOUT =  5
0013          LUTERM = 6
        C
0014          CALL CALIN( CUBSR, CUBIM, CUBCO )
        C
0015          NIMAGE = INT(1.001 + ALOG(FLOAT(SCOL))/ALOG(2.))
        C
0016          RETURN
0017          END
 *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
 *OPTIONS IN EFFECT*  NAME = INIT      LINECNT =      57
```

```
               C
               C*****************************************************************
               C
               C  File I/O Module
               C
               C  This module consists of procedures to handle the input/output
               C  activities for the surface acquisition program.
               C  Include procedures are
               C
               C    Calibin- reads all the data from the calibration file
               C    Imagein- reads the image data
               C    Surfout- writes the surface points
               C
               C*****************************************************************
               C
               C----------------------------------------------
               C
0001                 SUBROUTINE CALIN (CUBS, CUBI, CUBC)
               C
               C-------------------------------------------------------
               C
               C     INPUT ROUTINE FOR the CALIBRATION file
               C
               C  INCLUDE Global and IO commons
               C  "GLOBAL" common (100,120)
               C
0002                  INTEGER SROW, SCOL
               C        Dimensions of the shutter grid
               C
0003                  INTEGER CROW, CCOL
               C        Dimensions of the camera grid
               C
0004                  INTEGER NIMAGE, NUMCAL
               C        Number of patterns needed
               C        Number of calibration data
               C
0005                  COMMON/GLOBAL/
                  1       SROW, SCOL, CROW, CCOL, NIMAGE, NUMCAL
               C
               C  "IO" common (200, 210 )
0006                  INTEGER LUCAL, LUDAT, LUOUT, LUTERM
               C         Logical units for the calibration file, data, output,
               C          and the terminal.
               C
0007                  COMMON/IO/
                  1       LUCAL, LUDAT, LUOUT, LUTERM
               C
               C
               C
               C
               C  Declare parameters
0008                REAL CUBS(100,3)
0009                INTEGER CUBI(100,2), CUBC(100)
               C
```

STATISTICS NO DIAGNOSTICS GENERATED

```
0010               READ(LUCAL, 2000) SROW, SCOL, CROW, CCOL
0011          2000 FORMAT ( 2I6 / 2I6 )
             C
             C  Now read the data for the calibration cube
             C
0012               CALL CLRTBL( CUBS, 100, 3 )
0013               CALL CLRTBL( CUBI, 100, 2 )
0014               CALL CLRVC( CUBC, 100 )
             C
0015               NUMCAL = 0
0016               DO 100 I = 1,100
0017                  READ(LUCAL, 300, END = 2100) ( CUBS(I,J), J = 1, 3 )
0018                  READ(LUCAL, 400) ( CUBC(I), CUBI(I,1), CUBI(I,2) )
0019                  NUMCAL = NUMCAL + 1
0020           100 CONTINUE
0021          2100 CONTINUE
             C
             C    if there are too many data, stop
             C
0022               IF (( NUMCAL .LE. 100 ) .AND. ( NUMCAL .GE. 7 ) )  GO TO 4000
0023               WRITE(LUTERM, 3015) NUMCAL
0024               STOP
0025          4000 CONTINUE
             C
             C     WRITE(LUTERM, 3000) ((CUBS(I,J),J=1,3),I=1,100)
             C     WRITE(LUTERM, 3005) ((CUBI(I,J),J=1,2), I=1,100)
             C     WRITE(LUTERM, 3010) (CUBC(I),I=1,100)
             C
0026           300 FORMAT( 3F14.3 )
0027           400 FORMAT( 3I6 )
             C
0028          3000 FORMAT(99(3F10.2,/),3F10.2)
0029          3005 FORMAT(99(2I10,/),2I10)
0030          3010 FORMAT(100(I10,/))
0031          3015 FORMAT(1X, 'Number of calibration data is not adequate',
                  1         ' NumCal = ', I10 )
             C
0032               RETURN
0033               END
 *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
 *OPTIONS IN EFFECT*  NAME = CALIN    , LINECNT =       57
 *STATISTICS*    SOURCE STATEMENTS =        33,PROGRAM SIZE =      966
 *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
             C
             C ------------------------------------------------------------
             C
0001                SUBROUTINE IMGIN
             C
             C-------------------------------------------------------------
             C
             C      INPUT ROUTINE FOR RASTER DATA
             C
             C  COMMONS  included here
             C  "GLOBAL" common (100,120)
             C
0002                 INTEGER SROW, SCOL
             C         Dimensions of the shutter grid
             C
0003                 INTEGER CROW, CCOL
             C         Dimensions of the camera grid
             C
0004                 INTEGER NIMAGE, NUMCAL
             C         Number of patterns needed
             C         Number of calibration data
             C
0005                 COMMON/GLOBAL/
                  1       SROW, SCOL, CROW, CCOL, NIMAGE, NUMCAL
             C
             C  "IO" common (200, 210 )
0006                 INTEGER LUCAL, LUDAT, LUOUT, LUTERM
             C          Logical units for the calibration file, data, output,
             C            and the terminal.
             C
0007                 COMMON/IO/
                  1       LUCAL, LUDAT, LUOUT, LUTERM
             C
             C
             C  "Patterns" common (300,310)
0008                 INTEGER PAT1(128,128), PAT2(128,128), PAT3(128,128),
                  1           PAT4(128,128), PAT5(128,128), PAT6(128,128),
                  2           PAT7(128,128), PAT8(128,128)
             C           Contain the image data.
             C
0009                 COMMON/PATS/
                  1       PAT1, PAT2, PAT3, PAT4, PAT5, PAT6,
                  2       PAT7, PAT8
             C
             C
0010                CALL CLRTBL( PAT1, 128, 128 )
0011                CALL CLRTBL( PAT2, 128, 128 )
0012                CALL CLRTBL( PAT3, 128, 128 )
0013                CALL CLRTBL( PAT4, 128, 128 )
0014                CALL CLRTBL( PAT5, 128, 128 )
0015                CALL CLRTBL( PAT6, 128, 128 )
0016                CALL CLRTBL( PAT7, 128, 128 )
0017                CALL CLRTBL( PAT8, 128, 128 )
             C
```

```
               C      TO SIGNAL END OF PATTERN
               C
0018            1      READ (LUDAT, 2000, END=100) I,J
0019                   IF ((I .EQ.-1) .AND. (J.EQ.-1)) GO TO 2
0020                     PAT1(I,J) = 1
0021                   GO TO 1
               C
0022            2      READ (LUDAT, 2000,END=100) I,J
0023                   IF ((I .EQ. -1) .AND. (J.EQ.-1)) GO TO 3
0024                    PAT2(I,J) = 1
0025                   GO TO 2
               C
0026            3      READ (LUDAT, 2000, END=100) I,J
0027                   IF ((I .EQ.-1) .AND. (J.EQ.-1)) GO TO 4
0028                    PAT3(I,J) = 1
0029                   GO TO 3
               C
0030            4      READ (LUDAT, 2000, END=100) I,J
0031                   IF ((I .EQ. -1) .AND. (J.EQ.-1)) GO TO 5
0032                    PAT4(I,J) = 1
0033                   GO TO 4
               C
0034            5      READ (LUDAT, 2000, END=100) I,J
0035                   IF ((I .EQ. -1) .AND. (J.EQ.-1)) GO TO 6
0036                    PAT5(I,J) = 1
0037                   GO TO 5
               C
0038            6      READ (LUDAT, 2000, END=100) I,J
0039                   IF ((I .EQ. -1) .AND. (J.EQ.-1)) GO TO 7
0040                    PAT6(I,J) = 1
0041                   GO TO 6
               C
0042            7      READ (LUDAT, 2000, END=100) I,J
0043                   IF ((I .EQ. -1) .AND. (J.EQ.-1)) GO TO 8
0044                    PAT7(I,J) = 1
0045                   GO TO 7
               C
0046            8      READ (LUDAT, 2000, END=100) I,J
0047                   IF ((I .EQ. -1) .AND. (J.EQ.-1)) GO TO 100
0048                    PAT8(I,J) = 1
0049                   GO TO 8
               C
               C100    WRITE( LUTERM, 3000 ) (( PAT1(I,J), J=1,CCOL), I=1,CROW )
               C
0050            100    RETURN
0051            2000   FORMAT( 2I6 )
               C3000   FORMAT( 16I2 )
               C
0052                   END
 *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
 *OPTIONS IN EFFECT*  NAME = IMGIN    , LINECNT =       57
 *STATISTICS*    SOURCE STATEMENTS =        52,PROGRAM SIZE =      1702
 *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
               C                                                             235.000
               C                                                             236.000
               C---------------------------------------------------------    237.000
 0001                SUBROUTINE SUROUT( SURPNT )                             239.000
               C                                                             241.000
               C       OUTPUT ROUTINE FOR ESTIMATED LOCATION                 242.000
               C       OF POINTS                                             243.000
               C---------------------------------------------------------    244.000
               C                                                             245.000
               C   IO Commons included here                                  246.000
               C  "IO" common (200, 210 )                                    200.000
 0002                INTEGER LUCAL, LUDAT, LUOUT, LUTERM                     201.000
               C         Logical units for the calibration file, data, output,  202.000
               C          and the terminal.                                  203.000
               C                                                             204.000
 0003                COMMON/IO/                                              205.000
                         LUCAL, LUDAT, LUOUT, LUTERM                         206.000
               C                                                             207.000
               C                                                             208.000
               C                                                             248.000
               C                                                             249.000
               C  PARAMETERS                                                 250.000
 0004                REAL SURPNT(3)                                          251.000
               C                                                             252.000
 0005                WRITE (LUOUT, 2000) SURPNT(1), SURPNT(2), SURPNT(3)     253.000
 0006           2000 FORMAT(3F10.3)                                          254.000
               C                                                             255.000
 0007                RETURN                                                  256.000
 0008                END                                                     257.000
  *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
  *OPTIONS IN EFFECT*  NAME = SUROUT   , LINECNT =       57
  *STATISTICS*    SOURCE STATEMENTS =        8,PROGRAM SIZE =      332
  *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
      C
      C
      C
      C-----------------------------------------------------------------
      C
0001        SUBROUTINE TMAT ( CUB, DTA, TM )
      C
      C  calibration of T matrix by least square method
      C
      C  Here T matrix is normalized by T44
      C       NOTE T matrix is used only in (3-49) of Rogers in Eye.
      C            But in Phantom, T matrix should not be mormalized.
      C
      C-----------------------------------------------------------------
      C
      C  Include global and IO common here
      C  "GLOBAL" common (100,120)
      C
0002          INTEGER SROW, SCOL
      C         Dimensions of the shutter grid
      C
0003          INTEGER CROW, CCOL
      C         Dimensions of the camera grid
      C
0004          INTEGER NIMAGE, NUMCAL
      C         Number of patterns needed
      C         Number of calibration data
      C
0005          COMMON/GLOBAL/
           1      SROW, SCOL, CROW, CCOL, NIMAGE, NUMCAL
      C
      C  "IO" common (200, 210 )
0006          INTEGER LUCAL, LUDAT, LUOUT, LUTERM
      C           Logical units for the calibration file, data, output,
      C            and the terminal.
      C
0007          COMMON/IO/
           1      LUCAL, LUDAT, LUOUT, LUTERM
      C
      C
      C
      C   Solve BX = C
      C
0008        REAL CUB(100,3),TM(4,4)
0009        INTEGER DTA(100,2), IWK(11),IER
0010        REAL A(200,12),B(200,11),C(200),X(11),COMMU(4),WK(3000)
      C
      C    More least square parms
0011         REAL EPS/4.768E-07/,QL, QLMAX/1.E07/
0012         INTEGER ITS/10/, PRT1, PRT2
      C
      C  begin
      C
0013        CALL CLRTBL(A,200,12)
0014        CALL CLRTBL(TM,4,4)
```

```
           C
0015             NDATA = 2 * NUMCAL
0016             J=0
0017             DO 1005 I=1, NDATA, 2
0018                J=J+1
0019                A(I,1) = CUB(J,1)
0020                A(I+1,2) = CUB(J,1)
0021                A(I,4) = CUB(J,2)
0022                A(I+1,5) = CUB(J,2)
0023                A(I,7) = CUB(J,3)
0024                A(I+1,8) = CUB(J,3)
0025                A(I,10) = 1.
0026                A(I+1,11) = 1.
           C
0027                A(I,12) = - FLOAT( DTA(J,1) )
0028                A(I+1,12) = - FLOAT( DTA(J,2) )
0029                A(I,3) = - CUB(J,1) * FLOAT( DTA(J,1) )
0030                A(I+1,3) = - CUB(J,1) * FLOAT( DTA(J,2) )
0031                A(I,6) = - CUB(J,2) * FLOAT( DTA(J,1) )
0032                A(I+1,6) = - CUB(J,2) * FLOAT( DTA(J,2) )
0033                A(I,9) = - CUB(J,3) * FLOAT( DTA(J,1) )
0034                A(I+1,9) = - CUB(J,3) * FLOAT( DTA(J,2) )
0035        1005 CONTINUE
           C
0036             DO 1006 I=1,200
0037                DO 1007 J=1,11
0038                   B(I,J) = A(I,J)
0039        1007    CONTINUE
0040                C(I) = - A(I,12)
0041        1006 CONTINUE
           C
           C%%%% This is a call to IMSL package
           C%%%%
           C%%%% CALL LLBQF ( B,200,NDATA,11,C,200,1,0,COMMU,X,11,IWK,WK,IER )
           C%%%%
           C%%%% IF (IER .EQ. 0 ) GO TO 1010
           C
           C     This is a call to L2
           C
0042             PRT1 = 1
0043             PRT2 = 0
0044             CALL L2(NDATA,11,B,C,X,EPS,ITS,PRT1,PRT2,200,QL)
           C
0045             IF ( QL .LT. QLMAX ) GO TO 1010
           C
0046             WRITE(LUTERM,2001) QL
0047             STOP
           C
0048        1010 CONTINUE
0049             TM(1,1) = X(1)
0050             TM(1,2) = X(2)
0051             TM(1,4) = X(3)
0052             TM(2,1) = X(4)
0053             TM(2,2) = X(5)
```

```
0055              TM(3,1) = X(7)
0056              TM(3,2) = X(8)
0057              TM(3,4) = X(9)
0058              TM(4,1) = X(10)
0059              TM(4,2) = X(11)
0060              TM(4,4) = 1.
            C
0061              WRITE(LUTERM,2000) ((TM(I,J),J=1,4),I=1,4)
            C
0062              RETURN
            C
0063         2000 FORMAT(/,1X,'T Matrix determined',/,4(4F14.7,/))
            C
0064         2001 FORMAT(1X,'Abnormal termination --MESSAGE FROM SUBROUTINE TMAT',/,
                1        1X,'CONDITION NUMBER OF T MATRIX = ', E20.5 )
            C
0065              END
 *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
 *OPTIONS IN EFFECT*  NAME = TMAT    , LINECNT =       57
 *STATISTICS*    SOURCE STATEMENTS =       65,PROGRAM SIZE =    33300
 *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
              C
              C-----------------------------------------------------------------
              C
0001               SUBROUTINE LMAT ( SURPTS, COLPTS, LM )
              C
              C  calibration of L matrix by least square method
              C
              C  Here L matrix is normalized by L44
              C       NOTE L matrix is used only in (3-49) of Rogers in Eye.
              C            But in Phantom, L matrix should not be mormalized.
              C
              C-----------------------------------------------------------------
              C
              C  Include global and IO common here
              C  "GLOBAL" common (100,120)
              C
0002                INTEGER SROW, SCOL
              C       Dimensions of the shutter grid
              C
0003                INTEGER CROW, CCOL
              C       Dimensions of the camera grid
              C
0004                INTEGER NIMAGE, NUMCAL
              C       Number of patterns needed
              C       Number of calibration data
              C
0005                COMMON/GLOBAL/
                 1      SROW, SCOL, CROW, CCOL, NIMAGE, NUMCAL
              C
              C  "IO" common (200, 210 )
0006                INTEGER LUCAL, LUDAT, LUOUT, LUTERM
              C        Logical units for the calibration file, data, output,
              C          and the terminal.
              C
0007                COMMON/IO/
                 1      LUCAL, LUDAT, LUOUT, LUTERM
              C
              C
              C
              C    Solve BX = C
              C
0008               REAL SURPTS(100,3),LM(4,4)
0009               INTEGER COLPTS(100), IWK(11), IER
0010               REAL A(100,8),B(100,7),C(100),X(7),COMMU(4),WK(3000)
              C
              C    More least square parms
0011                REAL EPS/4.768E-07/,QL, QLMAX/1.E07/,WKAREA(200)
0012                INTEGER ITS/10/, PRT1, PRT2
              C
              C
              C  begin
              C
0013               CALL CLRTBL(A,100,8)
0014               CALL CLRTBL(LM,4,4)
```

```
      DO 1005 I=1, NUMCAL
         A(I,1) = SURPTS(I,1)
         A(I,2) = - SURPTS(I,1) * FLOAT( COLPTS(I) )
         A(I,3) = SURPTS(I,2)
         A(I,4) = - SURPTS(I,2) * FLOAT( COLPTS(I) )
         A(I,5) = SURPTS(I,3)
         A(I,6) = - SURPTS(I,3) * FLOAT( COLPTS(I) )
         A(I,7) = 1.
         A(I,8) = - FLOAT ( COLPTS(I) )
 1005 CONTINUE
C
      DO 1006 I=1,100
         DO 1007 J=1,7
            B(I,J) = A(I,J)
 1007    CONTINUE
         C(I) = - A(I,8)
 1006 CONTINUE
C
C%%%% This is a call to IMSL package
C%%%%
C%%%% CALL LLBQF ( B,100,NUMCAL,7,C,100,1,0,COMMU,X,7,IWK,WK,IER )
C%%%%
C%%%% IF( IER .EQ. 0 ) GO TO 1010
C
C     This is a call to L2
C
      PRT1 = 1
      PRT2 = 0
      CALL L2(NUMCAL,7,B,C,X,EPS,ITS,PRT1,PRT2,100,QL)
C
      IF ( QL .LT. QLMAX ) GO TO 1010
C
      WRITE(LUTERM,2001) QL
      STOP
C
 1010 CONTINUE
      LM(1,1) = X(1)
      LM(1,4) = X(2)
      LM(2,1) = X(3)
      LM(2,4) = X(4)
      LM(3,1) = X(5)
      LM(3,4) = X(6)
      LM(4,1) = X(7)
      LM(4,4) = 1.
C
      WRITE(LUTERM,2000) ((LM(I,J),J=1,4),I=1,4)
C
      RETURN
C
 2000 FORMAT(/,1X,'L Matrix determined',/,4(4F14.7,/))
C
 2001 FORMAT(1X,'Abnormal termination --MESSAGE FROM SUBROUTINE LMAT',
     1       /, 1X,'CONDITION NUMBER OF L MATRIX = ', E20.5 )
C
      END
```

```
*OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
*OPTIONS IN EFFECT*  NAME = LMAT     , LINECNT =       57
*STATISTICS*    SOURCE STATEMENTS =        50,PROGRAM SIZE =    20628
*STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
      C
      C
      C***************************************************************
      C
      C  MAP Module
      C
      C  This module does the mapping of the unknown surface.
      C
      C  Procedures:
      C    Map - Drives module
      C    Column - Determines the shutter column, u, for a given
      C             Camera grid point.
      C    Setcoef - Sets the coefficients for least sqares equations
      C
      C***************************************************************
      C
      C---------------------------------------------------------------
      C
0001        SUBROUTINE MAP (TM,LM)
      C
      C  This is the driver routine for the module.
      C  For each point imaged, it determines the
      C  shutter column which created the image,
      C  sets the coefficients for the equations
      C  which determine the surface point, and does
      C  the least squares procedure which determines
      C  the surface point.
      C
      C  Parameters:
      C     TM -  Transformation matrix for camera geometry
      C     LM -  Equation of the plane for each column of the shutter
      C
      C----------------------------------------------------------------
      C
      C  Include all commons here
      C  "GLOBAL" common (100,120)
      C
0002         INTEGER SROW, SCOL
      C        Dimensions of the shutter grid
      C
0003         INTEGER CROW, CCOL
      C        Dimensions of the camera grid
      C
0004         INTEGER NIMAGE, NUMCAL
      C        Number of patterns needed
      C        Number of calibration data
      C
0005         COMMON/GLOBAL/
           1      SROW, SCOL, CROW, CCOL, NIMAGE, NUMCAL
      C
      C  "IO" common (200, 210 )
0006         INTEGER LUCAL, LUDAT, LUOUT, LUTERM
      C          Logical units for the calibration file, data, output,
      C           and the terminal.
      C
```

```
0007              COMMON/10/
              1       LUCAL, LUDAT, LUOUT, LUTERM
           C
           C
           C  "Patterns" common (300,310)
0008              INTEGER PAT1(128,128), PAT2(128,128), PAT3(128,128),
              1           PAT4(128,128), PAT5(128,128), PAT6(128,128),
              2           PAT7(128,128), PAT8(128,128)
           C          Contain the image data.
           C
0009              COMMON/PATS/
              1       PAT1, PAT2, PAT3, PAT4, PAT5, PAT6,
              2       PAT7, PAT8
           C
0010             REAL  TM(4,4), LM(4,4)
           C
           C  Local variables:
           C
           C    Matrices for least squares AX=B
0011             REAL LSA(3,3), LSB(3), LSX(3)
           C
           C    Known values in least squares equations
0012             REAL COL, XSTAR, YSTAR
0013             INTEGER U
           C
           C    More least square parms
0014             REAL EPS/4.768E-07/,QL, QLMAX/1.E07/,WKAREA(200)
0015             INTEGER ITS/10/, PRT1, PRT2
           C
           C  Begin
           C
0016             DO 20 J = 1, CCOL
           C
0017               DO 10 I = 1, CROW
           C
0018                 IF (PAT1(I,J) .EQ. O) GO TO 10
           C
0019                 CALL CLRVC (LSB,3)
0020                 CALL CLRVC (LSX,3)
0021                 CALL CLRTBL (LSA,3,3)
           C
0022                 CALL COLUMN(I,J,U)
           C
0023                 XSTAR = FLOAT(I)
0024                 YSTAR = FLOAT(J)
0025                 COL = FLOAT(U)
           C
0026                 CALL SETCOF(LSA,LSB,XSTAR,YSTAR,COL,TM,LM)
           C
0027                 PRT1 = O
0028                 PRT2 = O
0029                 CALL L2(3,3,LSA,LSB,LSX,EPS,ITS,PRT1,PRT2,3,QL)
           C
           C%%%%%      CALL LEQT2F(LSA,1,3,3, LSB,O,WKAREA,QL)
```

```
0030                IF (QL .GT. QLMAX) GO TO 10
           C
           C%%%%%   CALL SUROUT( LSB )
           C
0031                WRITE (LUOUT, 100) U, I, J, LSX(1), LSX(2), LSX(3)
 /         C
0032          10   CONTINUE
0033          20  CONTINUE
           C
0034         100  FORMAT (1X, 3I6, 3F14.7 )
           C
0035              RETURN
0036              END
 *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
 *OPTIONS IN EFFECT*  NAME = MAP     , LINECNT =       57
 *STATISTICS*    SOURCE STATEMENTS =       36,PROGRAM SIZE =      1832
 *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
              C
              C-------------------------------------------------------------------
              C
 0001                 SUBROUTINE COLUMN (RW,CL,U)
              C
              C   Column finds the shutter column (u) for the given point by looking
              C   at the images the point appeared in.  It is assumed that pattern two
              C   illuminates the right half of the shutter, etc..
              C
              C   Input : RW -  camera row  for point.
              C           CL -  camera column for point.
              C   Output: U  -  shutter column number for point
              C
              C-------------------------------------------------------------------
              C
              C  Include GLOBAL and PATTERN commons here
              C  "GLOBAL" common (100,120)
              C
 0002                 INTEGER SROW, SCOL
              C         Dimensions of the shutter grid
              C
 0003                 INTEGER CROW, CCOL
              C         Dimensions of the camera grid
              C
 0004                 INTEGER NIMAGE, NUMCAL
              C         Number of patterns needed
              C         Number of calibration data
              C
 0005                 COMMON/GLOBAL/
                   1      SROW, SCOL, CROW, CCOL, NIMAGE, NUMCAL
              C
              C  "Patterns" common (300,310)
 0006                 INTEGER PAT1(128,128), PAT2(128,128), PAT3(128,128),
                   1          PAT4(128,128), PAT5(128,128), PAT6(128,128),
                   2          PAT7(128,128), PAT8(128,128)
              C           Contain the image data.
              C
 0007                 COMMON/PATS/
                   1      PAT1, PAT2, PAT3, PAT4, PAT5, PAT6,
                   2      PAT7, PAT8
              C
 0008                 INTEGER RW,CL,U
              C
              C  begin
              C
 0009                  U          = PAT2(RW,CL) * ( 2 ** (NIMAGE-2))
                   1              + PAT3(RW,CL) * ( 2 ** (NIMAGE-3))
                   2              + PAT4(RW,CL) * ( 2 ** (NIMAGE-4))
                   3              + PAT5(RW,CL) * ( 2 ** (NIMAGE-5))
                   4              + PAT6(RW,CL) * ( 2 ** (NIMAGE-6))
                   5              + PAT7(RW,CL) * ( 2 ** (NIMAGE-7))
                   6              + PAT8(RW,CL) * ( 2 ** (NIMAGE-8))
                   7              + 1
 0010                 RETURN
```

```
0011                 END                                                                              548.000
 *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
 *OPTIONS IN EFFECT*  NAME = COLUMN  , LINECNT =       57
 *STATISTICS*    SOURCE STATEMENTS =        11,PROGRAM SIZE =       986
 *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
               C
               C-------------------------------------------------------------
               C
 0001                 SUBROUTINE SETCOF(A, B, XS, YS, U, T, L)
               C
               C   Setcoef sets the coefficients on the following three
               C   equations which are used to solve for the point (x,y,z):
               C
               C  (T11-T14x*)x + (T21-T24x*)y + (T31-T34x*)z + (T41-T44x*) = 0
               C  (T12-T14y*)x + (T22-T24y*)y + (T32-T34y*)z + (T42-T44y*) = 0
               C  (L11-L14u )x + (L21-L24u )y + (L31-L34u )z + (L41-L44u ) = 0
               C
               C  Parameters:
               C
               C    Input:  XS - x* in the equations
               C            YS - y* in the equations
               C            U  - column of the shutter
               C            T  - Transformation matrix for camera geometry
               C            L  - Equation of the plane for each shutter column
               C
               C    Output:  A, B:  Least square matrices
               C
               C---------------------------------------------------------------
               C
 0002                 REAL XS, YS, U, T(4,4), L(4,4), A(3,3), B(3)
               C
               C  Begin...
 0003                 A(1,1) = T(1,1) - T(1,4) * XS
 0004                 A(1,2) = T(2,1) - T(2,4) * XS
 0005                 A(1,3) = T(3,1) - T(3,4) * XS
 0006                 B(1)   = - ( T(4,1) - T(4,4) * XS )
               C
 0007                 A(2,1) = T(1,2) - T(1,4) * YS
 0008                 A(2,2) = T(2,2) - T(2,4) * YS
 0009                 A(2,3) = T(3,2) - T(3,4) * YS
 0010                 B(2)   = - ( T(4,2) - T(4,4) * YS )
               C
 0011                 A(3,1) = L(1,1) - L(1,4) * U
 0012                 A(3,2) = L(2,1) - L(2,4) * U
 0013                 A(3,3) = L(3,1) - L(3,4) * U
 0014                 B(3)   = - ( L(4,1) - L(4,4) * U )
               C
               C
 0015                 RETURN
 0016                 END
  *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
  *OPTIONS IN EFFECT*  NAME = SETCOF   , LINECNT =      57
  *STATISTICS*    SOURCE STATEMENTS =       16,PROGRAM SIZE =      614
  *STATISTICS*  NO DIAGNOSTICS GENERATED

NO STATEMENTS FLAGGED IN THE ABOVE COMPILATIONS.
```

```
C
C      Mapping Program : Eye.TLmat
C
C
C
C      Version : This is based on the program of Dr. Martin AltschulerC
C                Calibration of T matrix and L matrix are not
C                performed.
C
C                Instead, T and L matrix are calculated by geometric
C                consideration.
C
C
C
C
C  DIRECTORY
C     SROW       # OF ROWS OF CONTOUR GRID
C     SCOL       # OF COLUMNS OF CONTOUR GRID
C     CROW       # OF ROWS IN ENTIRE RASTER SCREEN
C     CCOL       # OF COLUMNS IN ENTIRE RASTER SCREEN
C     NIMAGE       # OF INAGES NEEDED TO CODE SCENE
C
C     LUCAL        LOGICAL UNIT FOR CALIBRATION INFO
C     LUDAT        LOGICAL UNIT FOR INPUT DATA
C     LUOUT        LOGICAL UNIT FOR FILE OUTPUT
C     LUTERM       LOGICAL UNIT FOR TERMINAL OUTPUT
C
C     PAT1       MATRIX OF RASTER IMAGE # 1
C       :
C       :
C     PAT7       MATRIX OF RASTER IMAGE # 7
C
C
C  LOGICAL DEVICES
C
C
C     LOGICAL UNIT #3   CALIBRATION INFO
C     LOGICAL UNIT #4   INPUT FILE
C     LOGICAL UNIT #5   OUTPUT FILE
C     LOGICAL UNIT #6   TERMINAL
C
C
C------------------------------------------------------------
C
C             Main program
C
C------------------------------------------------------------
C
C   Local variables
C
C   Transformation matrices
C
0001      REAL T(4,4), L(4,4)
C
C  Begin
```

```
             C    Acquire and check the initial parameters.                          56.000
 0002               CALL INIT                                                        57.000
             C                                                                       58.000
             C    Calibrate the system.                                              59.000
 0003               CALL TMAT( T )                                                   60.000
 0004               CALL LMAT( L )                                                   61.000
             C                                                                       62.000
             C    Solve for the unknown surface.                                     63.000
 0005               CALL IMGIN                                                       64.000
 0006               CALL MAP( T, L )                                                 65.000
             C                                                                       66.000
 0007               STOP                                                             67.000
 0008            .  END                                                              68.000
  *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
  *OPTIONS IN EFFECT*  NAME = MAIN    , LINECNT =       57
  *STATISTICS*    SOURCE STATEMENTS =        8,PROGRAM SIZE =      484
  *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
            C
            C-----------------------------------------------------------
            C
0001                 SUBROUTINE INIT
            C
            C-----------------------------------------------------------
            C
            C  include GLOBAL and IO common
            C
            C
            C  "GLOBAL" common (1,15)
            C
0002                 REAL SDIST, STHETA, SPHI
            C
0003                 REAL SSEP, SWIDTH, SHIGH
            C
0004                 INTEGER SROW, SCOL
            C          Dimensions of the shutter grid
            C
0005                 REAL CDIST, CTHETA, CPHI
            C
0006                 REAL CSEP, CWIDTH, CHIGH
            C
0007                 INTEGER CROW, CCOL
            C          Dimensions of the camera grid
            C
0008                 INTEGER NIMAGE
            C          Number of patterns needed.
            C
0009                 COMMON/GLOBAL/
                 1       SDIST, STHETA, SPHI, SSEP, SWIDTH, SHIGH, SROW, SCOL,
                 2       CDIST, CTHETA, CPHI, CSEP, CWIDTH, CHIGH, CROW, CCOL,
                 3       NIMAGE
            C
            C  "IO" COMMON  (16,25)
            C
0010                 INTEGER LUCAL, LUDAT, LUOUT, LUTERM
            C           Logical units for the calibration file, data, output,
            C            and the terminal.
            C
0011                 COMMON/IO/
                 1       LUCAL, LUDAT, LUOUT, LUTERM
            C
            C
            C   Begin initialization
            C
0012               LUCAL =  3
0013               LUDAT =  4
0014               LUOUT =  5
0015               LUTERM = 6
            C
            C  read global variables
            C
0016               READ (LUCAL, 2000) SDIST, STHETA, SPHI, SSEP, SWIDTH, SHIGH,
                 1                    SCOL, SROW
```

```
 0017             READ (LUCAL, 2000) CDIST, CTHETA, CPHI, CSEP, CWIDTH, CHIGH,           92.000
                 1                  CCOL, CROW                                             93.000
 0018        2000 FORMAT ( F9.4, 5F8.4, 2I6 )                                               94.000
           C                                                                                95.000
 0019             NIMAGE = INT( 1.001 + ALOG( FLOAT( SCOL ) ) / ALOG(2.) )                 96.000
           C                                                                                97.000
           C convert to radian                                                              98.000
           C                                                                                99.000
 0020             ANGRAD = 3.141592 / 180                                                  100.000
 0021             CTHETA = CTHETA * ANGRAD                                                 101.000
 0022             CPHI   = CPHI * ANGRAD                                                   102.000
           C                                                                               103.000
           C                                                                               104.000
 0023             RETURN                                                                   105.000
 0024             END                                                                      106.000
  *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
  *OPTIONS IN EFFECT*  NAME = INIT    , LINECNT =       57
  *STATISTICS*    SOURCE STATEMENTS =       24,PROGRAM SIZE =      692
  *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
            C
            C
            C -------------------------------------------------------------
            C
0001                SUBROUTINE IMGIN
            C
            C--------------------------------------------------------------
            C
            C      INPUT ROUTINE FOR RASTER DATA
            C
            C   COMMONS  included here
            C
            C   "GLOBAL" common (1,15)
            C
0002                  REAL SDIST, STHETA, SPHI
            C
0003                  REAL SSEP, SWIDTH, SHIGH
            C
0004                  INTEGER SROW, SCOL
            C           Dimensions of the shutter grid
            C
0005                  REAL CDIST, CTHETA, CPHI
            C
0006                  REAL CSEP, CWIDTH, CHIGH
            C
0007                  INTEGER CROW, CCOL
            C           Dimensions of the camera grid
            C
0008                  INTEGER NIMAGE
            C           Number of patterns needed.
            C
0009                  COMMON/GLOBAL/
                 1        SDIST, STHETA, SPHI, SSEP, SWIDTH, SHIGH, SROW, SCOL,
                 2        CDIST, CTHETA, CPHI, CSEP, CWIDTH, CHIGH, CROW, CCOL,
                 3        NIMAGE
            C
            C   "IO" COMMON  (16,25)
            C
0010                  INTEGER LUCAL, LUDAT, LUOUT, LUTERM
            C            Logical units for the calibration file, data, output,
            C              and the terminal.
            C
0011                  COMMON/IO/
                 1        LUCAL, LUDAT, LUOUT, LUTERM
            C
            C   "Patterns" common (26,37)
            C
0012                  INTEGER PAT1(128,128), PAT2(128,128), PAT3(128,128),
                 1            PAT4(128,128), PAT5(128,128), PAT6(128,128),
                 2            PAT7(128,128), PAT8(128,128)
            C              Contain the image data.
            C
0013                  COMMON/PATS/
                 1        PAT1, PAT2, PAT3, PAT4, PAT5, PAT6,
                 2        PAT7, PAT8
```

```
               C
               C
0014                 CALL CLRTBL( PAT1, 128, 128 )
0015                 CALL CLRTBL( PAT2, 128, 128 )
0016                 CALL CLRTBL( PAT3, 128, 128 )
0017                 CALL CLRTBL( PAT4, 128, 128 )
0018                 CALL CLRTBL( PAT5, 128, 128 )
0019                 CALL CLRTBL( PAT6, 128, 128 )
0020                 CALL CLRTBL( PAT7, 128, 128 )
0021                 CALL CLRTBL( PAT8, 128, 128 )
               C
               C     LOOP THRU DATA UNTIL YOU GET A (-1,-1) ENTRY
               C     TO SIGNAL END OF PATTERN
               C
0022             1   READ (LUDAT, 2000, END=100) I,J
0023                 IF ((I .EQ.-1) .AND. (J.EQ.-1)) GO TO 2
0024                   PAT1(I,J) = 1
0025                 GO TO 1
               C
0026             2   READ (LUDAT, 2000,END=100) I,J
0027                 IF ((I .EQ. -1) .AND. (J.EQ.-1)) GO TO 3
0028                  PAT2(I,J) = 1
0029                 GO TO 2
               C
0030             3   READ (LUDAT, 2000, END=100) I,J
0031                 IF ((I .EQ.-1) .AND. (J.EQ.-1)) GO TO 4
0032                  PAT3(I,J) = 1
0033                 GO TO 3
               C
0034             4   READ (LUDAT, 2000, END=100) I,J
0035                 IF ((I .EQ. -1) .AND. (J.EQ.-1)) GO TO 5
0036                  PAT4(I,J) = 1
0037                 GO TO 4
               C
0038             5   READ (LUDAT, 2000, END=100) I,J
0039                 IF ((I .EQ. -1) .AND. (J.EQ.-1)) GO TO 6
0040                  PAT5(I,J) = 1
0041                 GO TO 5
               C
0042             6   READ (LUDAT, 2000, END=100) I,J
0043                 IF ((I .EQ. -1) .AND. (J.EQ.-1)) GO TO 7
0044                  PAT6(I,J) = 1
0045                 GO TO 6
               C
0046             7   READ (LUDAT, 2000, END=100) I,J
0047                 IF ((I .EQ. -1) .AND. (J.EQ.-1)) GO TO 8
0048                  PAT7(I,J) = 1
0049                 GO TO 7
               C
0050             8   READ (LUDAT, 2000, END=100) I,J
0051                 IF ((I .EQ. -1) .AND. (J.EQ.-1)) GO TO 100
0052                  PAT8(I,J) = 1
0053                 GO TO 8
```

```
              C
 0054            100  RETURN
 0055           2000  FORMAT( 2I6 )
 0056           3000  FORMAT( 16I4 )
              C
 0057                 END
  *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
  *OPTIONS IN EFFECT*  NAME = IMGIN    , LINECNT =       57
  *STATISTICS*     SOURCE STATEMENTS =        57,PROGRAM SIZE =      1710
  *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
              C
              C
              C-----------------------------------------------------------
 0001               SUBROUTINE SUROUT( SURPNT )
              C
              C       OUTPUT ROUTINE FOR ESTIMATED LOCATION
              C       OF POINTS
              C-----------------------------------------------------------
              C
              C  Commons included here
              C
              C  "IO" COMMON  (16,25)
              C
 0002                INTEGER LUCAL, LUDAT, LUOUT, LUTERM
              C         Logical units for the calibration file, data, output,
              C          and the terminal.
              C
 0003                COMMON/IO/
                 1        LUCAL, LUDAT, LUOUT, LUTERM
              C
              C
              C  PARAMETERS
 0004               REAL SURPNT(3)
              C
 0005               WRITE (LUOUT, 2000) SURPNT(1), SURPNT(2), SURPNT(3)
 0006          2000 FORMAT(3F10.3)
              C
 0007               RETURN
 0008               END
  *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
  *OPTIONS IN EFFECT*  NAME = SUROUT   , LINECNT =       57
  *STATISTICS*    SOURCE STATEMENTS =        8,PROGRAM SIZE =      332
  *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
            C
            C
            C
            C-------------------------------------------------------------
            C
0001              SUBROUTINE TMAT ( TM )
            C
            C  read in T matrix from phantom
            C-------------------------------------------------------------
            C
            C  Include IO common here
            C
            C  "IO" COMMON  (16,25)
            C
0002                INTEGER LUCAL, LUDAT, LUOUT, LUTERM
            C          Logical units for the calibration file, data, output,
            C           and the terminal.
            C
0003                COMMON/IO/
                 1      LUCAL, LUDAT, LUOUT, LUTERM
            C
            C
0004              REAL TM(4, 4)
            C
0005              CALL CLRTBL(TM, 4, 4)
            C
0006              READ (LUCAL,2000) (( TM(I, J), J = 1, 4), I = 1, 4)
            C     WRITE (LUTERM,3000) (( I, J, TM(I, J), J = 1, 4), I = 1, 4)
            C
0007         2000 FORMAT( F15.7, 3F14.7 )
            C3000 FORMAT( 4( 2I4, F14.7 ))
            C
0008              RETURN
0009              END
 *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
 *OPTIONS IN EFFECT*  NAME = TMAT    , LINECNT =       57
 *STATISTICS*    SOURCE STATEMENTS =         9,PROGRAM SIZE =      460
 *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
                C
                C
                C-------------------------------------------------------
                C
 0001                 SUBROUTINE LMAT ( LM )
                C
                C  read in L matrix from phantom
                C-------------------------------------------------------
                C
                C  Include GLOBAL and IO Common here
                C
                C  "GLOBAL" common (1,15)
                C
 0002                   REAL SDIST, STHETA, SPHI
                C
 0003                   REAL SSEP, SWIDTH, SHIGH
                C
 0004                   INTEGER SROW, SCOL
                C         Dimensions of the shutter grid
                C
 0005                   REAL CDIST, CTHETA, CPHI
                C
 0006                   REAL CSEP, CWIDTH, CHIGH
                C
 0007                   INTEGER CROW, CCOL
                C         Dimensions of the camera grid
                C
 0008                   INTEGER NIMAGE
                C         Number of patterns needed.
                C
 0009                   COMMON/GLOBAL/
                     1      SDIST, STHETA, SPHI, SSEP, SWIDTH, SHIGH, SROW, SCOL,
                     2      CDIST, CTHETA, CPHI, CSEP, CWIDTH, CHIGH, CROW, CCOL,
                     3      NIMAGE   *
                C
                C  "IO" COMMON  (16,25)
                C
 0010                   INTEGER LUCAL, LUDAT, LUOUT, LUTERM
                C          Logical units for the calibration file, data, output,
                C            and the terminal.
                C
 0011                   COMMON/IO/
                            LUCAL, LUDAT, LUOUT, LUTERM
                C
                C
 0012                 REAL LM(4,4)
                C
 0013                 CALL CLRTBL(LM, 4, 4)
                C
 0014                 READ (LUCAL, 2000) (( LM(I, J), J = 1, 4 ), I = 1, 4 )
                C     WRITE (LUTERM, 3000) (( I, J, LM(I, J), J = 1, 4 ), I = 1, 4 )
                C
 0015            2000 FORMAT( F15.7, 3F14.7 )
                C3000 FORMAT( 4( 2I4, F14.7 ) )
```

```
0016               RETURN                                                                         252.000
0017               END                                                                            253.000
 *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
 *OPTIONS IN EFFECT*  NAME = LMAT     , LINECNT =       57
 *STATISTICS*    SOURCE STATEMENTS =       17,PROGRAM SIZE =      468
 *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
              C
              C----------------------------------------------------------------
              C
              C  MAP Module
              C
              C  This module does the mapping of the unknown surface.
              C
              C  Procedures:
              C    Map - Drives module
              C    Column - Determines the shutter column, u, for a given
              C             Camera grid point.
              C    Setcoef - Sets the coefficients for least sqares equations
              C
              C----------------------------------------------------------------
              C------------------------------------------------------------------
              C
0001                SUBROUTINE MAP (TM,LM)
              C
              C  This is the driver routine for the module.
              C  For each point imaged, it determines the
              C  shutter column which created the image,
              C  sets the coefficients for the equations
              C  which determine the surface point, and does
              C  the least squares procedure which determines
              C  the surface point.
              C
              C  Parameters:
              C     TM -  Transformation matrix for camera geometry
              C     LM -  Equation of the plane for each column of the shutter
              C
              C-------------------------------------------------------------------
              C
              C  Include all commons here
              C
              C  "GLOBAL" common (1,15)
              C
0002                 REAL SDIST, STHETA, SPHI
              C
0003                 REAL SSEP, SWIDTH, SHIGH
              C
0004                 INTEGER SROW, SCOL
              C        Dimensions of the shutter grid
              C
0005                 REAL CDIST, CTHETA, CPHI
              C
0006                 REAL CSEP, CWIDTH, CHIGH
              C
0007                 INTEGER CROW, CCOL
              C        Dimensions of the camera grid
              C
0008                 INTEGER NIMAGE
              C        Number of patterns needed.
              C
0009                 COMMON/GLOBAL/
```

```
                 2      CDIST, CTHETA, CPHI, CSEP, CWIDTH, CHIGH, CROW, CCOL,
                 3      NIMAGE
               C
               C  "IO" COMMON  (16,25)
               C
0010                  INTEGER LUCAL, LUDAT, LUOUT, LUTERM
               C         Logical units for the calibration file, data, output,
               C          and the terminal.
               C
0011                  COMMON/IO/
                 1      LUCAL, LUDAT, LUOUT, LUTERM
               C
               C  "Patterns" common (26,37)
               C
0012                  INTEGER PAT1(128,128), PAT2(128,128), PAT3(128,128),
                 1          PAT4(128,128), PAT5(128,128), PAT6(128,128),
                 2          PAT7(128,128), PAT8(128,128)
               C          Contain the image data.
               C
0013                  COMMON/PATS/
                 1      PAT1, PAT2, PAT3, PAT4, PAT5, PAT6,
                 2      PAT7, PAT8
               C
0014                 REAL  TM(4,4), LM(4,4)
               C
               C  Local variables:
               C
               C    Matrices for least squares AX=B
0015                 REAL LSA(3,3), LSB(3), LSX(3)
               C
               C    Known values in least squares equations
0016                 REAL COL, XSTAR, YSTAR
0017                 INTEGER U
               C
               C    More least square parms
0018                 REAL EPS/4.768E-07/,QL, QLMAX/1.E07/,WKAREA(200)
0019                 INTEGER ITS/10/, PRT1, PRT2
               C
               C    Begin...
               C
0020                 DO 20 J = 1, CCOL
               C
0021                   DO 10 I = 1, CROW
               C
0022                     IF (PAT1(I,J) .EQ. 0) GO TO 10
               C
0023                     CALL CLRVC (LSB,3)
0024                     CALL CLRVC (LSX,3)
0025                     CALL CLRTBL (LSA,3,3)
               C
0026                     CALL COLUMN(I,J,U)
               C
0027                     XSTAR = FLOAT(I)
0028                     YSTAR = FLOAT(J)
0029                     COL = FLOAT(U)
```

```
            C                                                                        322.000
 0030                  CALL SETCOF(LSA,LSB,XSTAR,YSTAR,COL,TM,LM)                    323.000
            C                                                                        324.000
 0031                  PRT1 = 0                                                      325.000
 0032                  PRT2 = 0                                                      326.000
 0033                  CALL L2(3,3,LSA,LSB,LSX,EPS,ITS,PRT1,PRT2,3,QL)               327.000
            C                                                                        328.000
            C%%%%%     CALL LEQT2F(LSA,1,3,3, LSB,0,WKAREA,QL)                       329.000
            C                                                                        330.000
 0034                  IF (QL .GT. QLMAX) GO TO 10                                   331.000
            C                                                                        332.000
            C%%%%%     CALL SUROUT( LSB )                                            333.000
            C                                                                        334.000
 0035                  WRITE (LUOUT, 100) U, I, J, LSX(1), LSX(2), LSX(3)            336.000
            C                                                                        337.000
 0036          10    CONTINUE                                                        338.000
 0037          20  CONTINUE                                                          339.000
            C                                                                        340.000
 0038         100  FORMAT (1X, 3I6, 3F14.7 )                                         342.000
            C                                                                        343.000
 0039              RETURN                                                            344.000
 0040              END                                                               345.000
  *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
  *OPTIONS IN EFFECT*  NAME = MAP     , LINECNT =       57
  *STATISTICS*    SOURCE STATEMENTS =       40,PROGRAM SIZE =      1832
  *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
        C
        C------------------------------------------------------------------
        C
0001          SUBROUTINE COLUMN (RW,CL,U)
        C
        C   Column finds the shutter column (u) for the given point by looking
        C   at the images the point appeared in.  It is assumed that pattern two
        C   illuminates the right half of the shutter, etc..
        C
        C   Input : RW -  camera row  for point.
        C           CL -  camera column for point.
        C   Output: U  -  shutter column number for point
        C
        C------------------------------------------------------------------
        C
        C  Include GLOBAL and PATTERN commons here
        C
        C  "GLOBAL" common (1,15)
        C
0002          REAL SDIST, STHETA, SPHI
        C
0003          REAL SSEP, SWIDTH, SHIGH
        C
0004          INTEGER SROW, SCOL
        C       Dimensions of the shutter grid
        C
0005          REAL CDIST, CTHETA, CPHI
        C
0006          REAL CSEP, CWIDTH, CHIGH
        C
0007          INTEGER CROW, CCOL
        C       Dimensions of the camera grid
        C
0008          INTEGER NIMAGE
        C       Number of patterns needed.
        C
0009          COMMON/GLOBAL/
             1    SDIST, STHETA, SPHI, SSEP, SWIDTH, SHIGH, SROW, SCOL,
             2    CDIST, CTHETA, CPHI, CSEP, CWIDTH, CHIGH, CROW, CCOL,
             3    NIMAGE
        C
        C  "Patterns" common (26,37)
        C
0010          INTEGER PAT1(128,128), PAT2(128,128), PAT3(128,128),
             1        PAT4(128,128), PAT5(128,128), PAT6(128,128),
             2        PAT7(128,128), PAT8(128,128)
        C         Contain the image data.
        C
0011          COMMON/PATS/
             1    PAT1, PAT2, PAT3, PAT4, PAT5, PAT6,
             2    PAT7, PAT8
        C
0012          INTEGER RW,CL,U
        C
        C  begin
```

```
            C
 0013                    U          = PAT2(RW,CL) * ( 2 ** (NIMAGE-2))
                    1               + PAT3(RW,CL) * ( 2 ** (NIMAGE-3))
                    2               + PAT4(RW,CL) * ( 2 ** (NIMAGE-4))
                    3               + PAT5(RW,CL) * ( 2 ** (NIMAGE-5))
                    4               + PAT6(RW,CL) * ( 2 ** (NIMAGE-6))
                    5               + PAT7(RW,CL) * ( 2 ** (NIMAGE-7))
                    6               + PAT8(RW,CL) * ( 2 ** (NIMAGE-8))
                    7               + 1
 0014                    RETURN
            C
 0015                    END
  *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
  *OPTIONS IN EFFECT*  NAME = COLUMN   , LINECNT =       57
  *STATISTICS*    SOURCE STATEMENTS =       15,PROGRAM SIZE =      986
  *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
          C
          C--------------------------------------------------------------
          C
0001              SUBROUTINE SETCOF(A, B, XS, YS, U, T, L)
          C
          C   Setcoef sets the coefficients on the following three
          C   equations which are used to solve for the point (x,y,z):
          C
          C  (T11-T14x*)x + (T21-T24x*)y + (T31-T34x*)z + (T41-T44x*) = 0
          C  (T12-T14y*)x + (T22-T24y*)y + (T32-T34y*)z + (T42-T44y*) = 0
          C  (L11-L14u )x + (L21-L24u )y + (L31-L34u )z + (L41-L44u ) = 0
          C
          C  Parameters:
          C
          C    Input:  XS - x* in the equations
          C            YS - y* in the equations
          C            U  - column of the shutter
          C            T  - Transformation matrix for camera geometry
          C            L  - Equation of the plane for each shutter column
          C
          C    Output:  A, B:  Least square matrices
          C
          C--------------------------------------------------------------
          C
0002              REAL XS, YS, U, T(4,4), L(4,4), A(3,3), B(3)
          C
          C  Begin...
0003              A(1,1) = T(1,1) - T(1,4) * XS
0004              A(1,2) = T(2,1) - T(2,4) * XS
0005              A(1,3) = T(3,1) - T(3,4) * XS
0006              B(1)   = - ( T(4,1) - T(4,4) * XS )
          C
0007              A(2,1) = T(1,2) - T(1,4) * YS
0008              A(2,2) = T(2,2) - T(2,4) * YS
0009              A(2,3) = T(3,2) - T(3,4) * YS
0010              B(2)   = - ( T(4,2) - T(4,4) * YS )
          C
0011              A(3,1) = L(1,1) - L(1,4) * U
0012              A(3,2) = L(2,1) - L(2,4) * U
0013              A(3,3) = L(3,1) - L(3,4) * U
0014              B(3)   = - ( L(4,1) - L(4,4) * U )
          C
          C
0015              RETURN
0016              END
  *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
  *OPTIONS IN EFFECT*  NAME = SETCOF   , LINECNT =       57
  *STATISTICS*    SOURCE STATEMENTS =       16,PROGRAM SIZE =      614
  *STATISTICS*  NO DIAGNOSTICS GENERATED

NO STATEMENTS FLAGGED IN THE ABOVE COMPILATIONS.
```

8.3. Eye.Tlmat Program

```
      C
      C      Mapping Program : Eye.TLmat
      C
      C
      C
      C      Version : This is based on the program of Dr. Martin AltschulerC
      C                Calibration of T matrix and L matrix are not
      C                performed.
      C
      C                Instead, T and L matrix are calculated by geometric
      C                consideration.
      C
      C
      C
      C
      C  DIRECTORY
      C     SROW       # OF ROWS OF CONTOUR GRID
      C     SCOL       # OF COLUMNS OF CONTOUR GRID
      C     CROW       # OF ROWS IN ENTIRE RASTER SCREEN
      C     CCOL       # OF COLUMNS IN ENTIRE RASTER SCREEN
      C     NIMAGE       # OF INAGES NEEDED TO CODE SCENE
      C                  |
      C     LUCAL        LOGICAL UNIT FOR CALIBRATION INFO
      C     LUDAT        LOGICAL UNIT FOR INPUT DATA
      C     LUOUT        LOGICAL UNIT FOR FILE OUTPUT
      C     LUTERM       LOGICAL UNIT FOR TERMINAL OUTPUT
      C
      C     PAT1       MATRIX OF RASTER IMAGE # 1
      C       :
      C       :
      C     PAT7       MATRIX OF RASTER IMAGE # 7
      C
      C
      C  LOGICAL DEVICES
      C
      C
      C     LOGICAL UNIT #3   CALIBRATION INFO
      C     LOGICAL UNIT #4   INPUT FILE
      C     LOGICAL UNIT #5   OUTPUT FILE
      C     LOGICAL UNIT #6   TERMINAL
      C
      C
      C-----------------------------------------------------------
      C
      C            Main program
      C
      C-----------------------------------------------------------
      C
      C   Local variables
      C
      C   Transformation matrices
      C
0001        REAL T(4,4), L(4,4)
```

```
              C   Acquire and check the initial parameters.
0002                CALL INIT
              C
              C   Calibrate the system.
0003                CALL TMAT( T )
0004                CALL LMAT( L )
              C
              C   Solve for the unknown surface.
0005                CALL IMGIN
0006                CALL MAP( T, L )
              C
0007                STOP
0008                END
 *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
 *OPTIONS IN EFFECT*  NAME = MAIN    , LINECNT =       57
 *STATISTICS*    SOURCE STATEMENTS =        8,PROGRAM SIZE =      484
 *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
            C
            C-----------------------------------------------------
            C
 0001              SUBROUTINE INIT
            C
            C-----------------------------------------------------
            C
            C   include GLOBAL and IO common
            C
            C
            C   "GLOBAL" common (1,15)
            C
 0002              REAL SDIST, STHETA, SPHI
            C
 0003              REAL SSEP, SWIDTH, SHIGH
            C
 0004              INTEGER SROW, SCOL
            C        Dimensions of the shutter grid
            C
 0005              REAL CDIST, CTHETA, CPHI
            C
 0006              REAL CSEP, CWIDTH, CHIGH
            C
 0007              INTEGER CROW, CCOL
            C        Dimensions of the camera grid
            C
 0008              INTEGER NIMAGE
            C        Number of patterns needed.
            C
 0009              COMMON/GLOBAL/
                1      SDIST, STHETA, SPHI, SSEP, SWIDTH, SHIGH, SROW, SCOL,
                2      CDIST, CTHETA, CPHI, CSEP, CWIDTH, CHIGH, CROW, CCOL,
                3      NIMAGE
            C
            C   "IO" COMMON  (16,25)
            C
 0010              INTEGER LUCAL, LUDAT, LUOUT, LUTERM
            C         Logical units for the calibration file, data, output,
            C          and the terminal.
            C
 0011              COMMON/IO/
                       LUCAL, LUDAT, LUOUT, LUTERM
            C
            C
            C   Begin initialization
            C
 0012            LUCAL  =  3
 0013            LUDAT  =  4
 0014            LUOUT  =  5
 0015            LUTERM = 6
            C
            C  read global variables
            C
 0016            READ (LUCAL, 2000) SDIST, STHETA, SPHI, SSEP, SWIDTH, SHIGH,
```

```
0017           READ (LUCAL, 2000) CDIST, CTHETA, CPHI, CSEP, CWIDTH, CHIGH,
              1                    CCOL, CROW
0018      2000 FORMAT ( F9.4, 5F8.4, 2I6 )
         C
0019           NIMAGE = INT( 1.001 + ALOG( FLOAT( SCOL ) ) / ALOG(2.) )
         C
         C convert to radian
         C
0020           ANGRAD = 3.141592 / 180
0021           CTHETA = CTHETA * ANGRAD
0022           CPHI   = CPHI * ANGRAD
         C
         C
0023           RETURN
0024           END
 *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
 *OPTIONS IN EFFECT*  NAME = INIT    , LINECNT =       57
 *STATISTICS*    SOURCE STATEMENTS =       24,PROGRAM SIZE =      692
 *STATISTICS* 'NO DIAGNOSTICS GENERATED
```

```
             C
             C
             C ----------------------------------------------------------------
             C
  0001                SUBROUTINE IMGIN
             C
             C-----------------------------------------------------------------
             C
             C       INPUT ROUTINE FOR RASTER DATA
             C
             C  COMMONS  included here
             C
             C  "GLOBAL" common (1,15)
             C
  0002                REAL SDIST, STHETA, SPHI
             C
  0003                REAL SSEP, SWIDTH, SHIGH
             C
  0004                INTEGER SROW, SCOL
             C          Dimensions of the shutter grid
             C
  0005                REAL CDIST, CTHETA, CPHI
             C
  0006                REAL CSEP, CWIDTH, CHIGH
             C
  0007                INTEGER CROW, CCOL
             C          Dimensions of the camera grid
             C
  0008                INTEGER NIMAGE
             C          Number of patterns needed.
             C
  0009                COMMON/GLOBAL/
                  1       SDIST, STHETA, SPHI, SSEP, SWIDTH, SHIGH, SROW, SCOL,
                  2       CDIST, CTHETA, CPHI, CSEP, CWIDTH, CHIGH, CROW, CCOL,
                  3       NIMAGE
             C
             C  "IO" COMMON  (16,25)
             C
  0010                INTEGER LUCAL, LUDAT, LUOUT, LUTERM
             C           Logical units for the calibration file, data, output,
             C            and the terminal.
             C
  0011                COMMON/IO/
                  1       LUCAL, LUDAT, LUOUT, LUTERM
             C
             C  "Patterns" common (26,37)
             C
  0012                INTEGER PAT1(128,128), PAT2(128,128), PAT3(128,128),
                  1           PAT4(128,128), PAT5(128,128), PAT6(128,128),
                  2           PAT7(128,128), PAT8(128,128)
             C            Contain the image data.
             C
  0013                COMMON/PATS/
                  1       PAT1, PAT2, PAT3, PAT4, PAT5, PAT6,
```

```
C
C
      CALL CLRTBL( PAT1, 128, 128 )
      CALL CLRTBL( PAT2, 128, 128 )
      CALL CLRTBL( PAT3, 128, 128 )
      CALL CLRTBL( PAT4, 128, 128 )
      CALL CLRTBL( PAT5, 128, 128 )
      CALL CLRTBL( PAT6, 128, 128 )
      CALL CLRTBL( PAT7, 128, 128 )
      CALL CLRTBL( PAT8, 128, 128 )
C
C     LOOP THRU DATA UNTIL YOU GET A (-1,-1) ENTRY
C     TO SIGNAL END OF PATTERN
C
 1    READ (LUDAT, 2000, END=100) I,J
      IF ((I .EQ.-1) .AND. (J.EQ.-1)) GO TO 2
        PAT1(I,J) = 1
      GO TO 1
C
 2    READ (LUDAT, 2000,END=100) I,J
      IF ((I .EQ. -1) .AND. (J.EQ.-1)) GO TO 3
       PAT2(I,J) = 1
      GO TO 2
C
 3    READ (LUDAT, 2000, END=100) I,J
      IF ((I .EQ.-1) .AND. (J.EQ.-1)) GO TO 4
       PAT3(I,J) = 1
      GO TO 3
C
 4    READ (LUDAT, 2000, END=100) I,J
      IF ((I .EQ. -1) .AND. (J.EQ.-1)) GO TO 5
       PAT4(I,J) = 1
      GO TO 4
C
 5    READ (LUDAT, 2000, END=100) I,J
      IF ((I .EQ. -1) .AND. (J.EQ.-1)) GO TO 6
       PAT5(I,J) = 1
      GO TO 5
C
 6    READ (LUDAT, 2000, END=100) I,J
      IF ((I .EQ. -1) .AND. (J.EQ.-1)) GO TO 7
       PAT6(I,J) = 1
      GO TO 6
C
 7    READ (LUDAT, 2000, END=100) I,J
      IF ((I .EQ. -1) .AND. (J.EQ.-1)) GO TO 8
       PAT7(I,J) = 1
      GO TO 7
C
 8    READ (LUDAT, 2000, END=100) I,J
      IF ((I .EQ. -1) .AND. (J.EQ.-1)) GO TO 100
       PAT8(I,J) = 1
      GO TO 8
C
C100  WRITE( LUTERM, 3000 ) (( PAT1(I,J), J=1,CCOL), I=1,CROW )
```

```
             C                                                          174.000
0054           100  RETURN                                              175.000
0055          2000  FORMAT( 2I6 )                                       176.000
0056          3000  FORMAT( 16I4 )                                      177.000
             C                                                          178.000
0057                END                                                 179.000
 *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
 *OPTIONS IN EFFECT*  NAME = IMGIN    , LINECNT =       57
 *STATISTICS*    SOURCE STATEMENTS =        57,PROGRAM SIZE =      1710
 *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
            C
            C
            C------------------------------------------------------------
0001              SUBROUTINE SUROUT( SURPNT )
            C
            C        OUTPUT ROUTINE FOR ESTIMATED LOCATION
            C        OF POINTS
            C------------------------------------------------------------
            C
            C  .Commons included here
            C
            C   "IO" COMMON  (16,25)
            C
0002                INTEGER LUCAL, LUDAT, LUOUT, LUTERM
            C          Logical units for the calibration file, data, output,
            C           and the terminal.
            C
0003                COMMON/IO/
                        LUCAL, LUDAT, LUOUT, LUTERM
            C
            C
            C  PARAMETERS
0004                REAL SURPNT(3)
            C
0005                WRITE (LUOUT, 2000) SURPNT(1), SURPNT(2), SURPNT(3)
0006         2000   FORMAT(3F10.3)
            C
0007                RETURN
0008                END
 *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
 *OPTIONS IN EFFECT*  NAME = SUROUT   , LINECNT =       57
 *STATISTICS*    SOURCE STATEMENTS =        8,PROGRAM SIZE =      332
 *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
             C
             C
             C
             C-----------------------------------------------------
             C
0001               SUBROUTINE TMAT ( TM )
             C
             C  read in T matrix from phantom
             C-----------------------------------------------------
             C
             C  Include IO common here
             C
             C  "IO" COMMON  (16,25)
             C
0002                 INTEGER LUCAL, LUDAT, LUOUT, LUTERM
             C          Logical units for the calibration file, data, output,
             C           and the terminal.
             C
0003                 COMMON/IO/
                  1       LUCAL, LUDAT, LUOUT, LUTERM
             C
             C
0004               REAL TM(4, 4)
             C
0005               CALL CLRTBL(TM, 4, 4)
             C
0006               READ (LUCAL,2000) (( TM(I, J), J = 1, 4), I = 1, 4)
             C     WRITE (LUTERM,3000) (( I, J, TM(I, J), J = 1, 4), I = 1, 4)
             C
0007          2000 FORMAT( F15.7, 3F14.7 )
             C3000 FORMAT( 4( 2I4, F14.7 ))
             C
0008               RETURN
0009               END
 *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
 *OPTIONS IN EFFECT*  NAME = TMAT    , LINECNT =       57
 *STATISTICS*    SOURCE STATEMENTS =        9,PROGRAM SIZE =      460
 *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
        C
        C
        C-----------------------------------------------------
        C
0001          SUBROUTINE LMAT ( LM )
        C
        C  read in L matrix from phantom
        C-----------------------------------------------------
        C
        C  Include GLOBAL and IO Common here
        C
        C  "GLOBAL" common (1,15)
        C
0002            REAL SDIST, STHETA, SPHI
        C
0003            REAL SSEP, SWIDTH, SHIGH
        C
0004            INTEGER SROW, SCOL
        C         Dimensions of the shutter grid
        C
0005            REAL CDIST, CTHETA, CPHI
        C
0006            REAL CSEP, CWIDTH, CHIGH
        C
0007            INTEGER CROW, CCOL
        C         Dimensions of the camera grid
        C
0008            INTEGER NIMAGE
        C         Number of patterns needed.
        C
0009            COMMON/GLOBAL/
             1      SDIST, STHETA, SPHI, SSEP, SWIDTH, SHIGH, SROW, SCOL,
             2      CDIST, CTHETA, CPHI, CSEP, CWIDTH, CHIGH, CROW, CCOL,
             3      NIMAGE
        C
        C  "IO" COMMON  (16,25)
        C
0010            INTEGER LUCAL, LUDAT, LUOUT, LUTERM
        C          Logical units for the calibration file, data, output,
        C           and the terminal.
        C
0011            COMMON/IO/
             1      LUCAL, LUDAT, LUOUT, LUTERM
        C
        C
0012          REAL LM(4,4)
        C
0013          CALL CLRTBL(LM, 4, 4)
        C
0014          READ (LUCAL, 2000) ( ( LM(I, J), J = 1, 4 ), I = 1, 4 )
        C     WRITE (LUTERM, 3000) (( I, J, LM(I, J), J = 1, 4 ), I = 1, 4 )
        C
0015     2000 FORMAT( F15.7, 3F14.7 )
        C3000 FORMAT( 4( 2I4, F14.7 ) )
        C
```

```
 0016             RETURN                                                     252.000
 0017             END                                                        253.000
  *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
  *OPTIONS IN EFFECT*  NAME = LMAT     , LINECNT =       57
  *STATISTICS*    SOURCE STATEMENTS =       17,PROGRAM SIZE =       468
  *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
C
C---------------------------------------------------------------
C
C  MAP Module
C
C  This module does the mapping of the unknown surface.
C
C  Procedures:
C    Map - Drives module
C    Column - Determines the shutter column, u, for a given
C             Camera grid point.
C    Setcoef - Sets the coefficients for least sqares equations
C
C---------------------------------------------------------------
C-----------------------------------------------------------------
C
      SUBROUTINE MAP (TM,LM)
C
C  This is the driver routine for the module.
C  For each point imaged, it determines the
C  shutter column which created the image,
C  sets the coefficients for the equations
C  which determine the surface point, and does
C  the least squares procedure which determines
C  the surface point.
C
C  Parameters:
C     TM -  Transformation matrix for camera geometry
C     LM -  Equation of the plane for each column of the shutter
C
C-----------------------------------------------------------------
C
C  Include all commons here
C
C  "GLOBAL" common (1,15)
C
      REAL SDIST, STHETA, SPHI
C
      REAL SSEP, SWIDTH, SHIGH
C
      INTEGER SROW, SCOL
C       Dimensions of the shutter grid
C
      REAL CDIST, CTHETA, CPHI
C
      REAL CSEP, CWIDTH, CHIGH
C
      INTEGER CROW, CCOL
C       Dimensions of the camera grid
C
      INTEGER NIMAGE
C       Number of patterns needed.
C
      COMMON/GLOBAL/
     1    SDIST, STHETA, SPHI, SSEP, SWIDTH, SHIGH, SROW, SCOL,
```

```
                2      CDIST, CTHETA, CPHI, CSEP, CWIDTH, CHIGH, CROW, CCOL,
                3      NIMAGE
             C
             C  "IO" COMMON  (16,25)
             C
 0010                INTEGER LUCAL, LUDAT, LUOUT, LUTERM
             C          Logical units for the calibration file, data, output,
             C           and the terminal.
             C
 0011                COMMON/IO/
                1      LUCAL, LUDAT, LUOUT, LUTERM
             C
             C  "Patterns" common (26,37)
             C
 0012                INTEGER PAT1(128,128), PAT2(128,128), PAT3(128,128),
                1          PAT4(128,128), PAT5(128,128), PAT6(128,128),
                2          PAT7(128,128), PAT8(128,128)
             C            Contain the image data.
             C
 0013                COMMON/PATS/
                1      PAT1, PAT2, PAT3, PAT4, PAT5, PAT6,
                2      PAT7, PAT8
             C
 0014              REAL  TM(4,4), LM(4,4)
             C
             C Local variables:
             C
             C   Matrices for least squares AX=B
 0015              REAL LSA(3,3), LSB(3), LSX(3)
             C
             C   Known values in least squares equations
 0016              REAL COL, XSTAR, YSTAR
 0017              INTEGER U
             C
             C   More least square parms
 0018              REAL EPS/4.768E-07/,QL, QLMAX/1.E07/,WKAREA(200)
 0019              INTEGER ITS/10/, PRT1, PRT2
             C
             C   Begin...
             C
 0020              DO 20 J = 1, CCOL
             C
 0021                DO 10 I = 1, CROW
             C
 0022                  IF (PAT1(I,J) .EQ. 0) GO TO 10
             C
 0023                  CALL CLRVC (LSB,3)
 0024                  CALL CLRVC (LSX,3)
 0025                  CALL CLRTBL (LSA,3,3)
             C
 0026                  CALL COLUMN(I,J,U)
             C
 0027                  XSTAR = FLOAT(I)
 0028                  YSTAR = FLOAT(J)
```

```
                C                   CALL SETCOF(LSA,LSB,XSTAR,YSTAR,COL,TM,LM)          322.000
0030                                CALL SETCOF(LSA,LSB,XSTAR,YSTAR,COL,TM,LM)          323.000
                C                                                                       324.000
0031                                PRT1 = 0                                            325.000
0032                                PRT2 = 0                                            326.000
0033                                CALL L2(3,3,LSA,LSB,LSX,EPS,ITS,PRT1,PRT2,3,QL)     327.000
                C                                                                       328.000
                C%%%%%              CALL LEQT2F(LSA,1,3,3, LSB,0,WKAREA,QL)             329.000
                C                                                                       330.000
0034                                IF (QL .GT. QLMAX) GO TO 10                         331.000
                C                                                                       332.000
                C%%%%%              CALL SUROUT( LSB )                                  333.000
                C                                                                       334.000
0035                                WRITE (LUOUT, 100) U, I, J, LSX(1), LSX(2), LSX(3)  336.000
                C                                                                       337.000
0036               10     CONTINUE                                                      338.000
0037               20   CONTINUE                                                        339.000
                C                                                                       340.000
0038              100   FORMAT (1X, 3I6, 3F14.7 )                                       342.000
                C                                                                       343.000
0039                    RETURN                                                          344.000
0040                    END                                                             345.000
 *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
 *OPTIONS IN EFFECT*  NAME = MAP      , LINECNT =       57
 *STATISTICS*    SOURCE STATEMENTS =        40,PROGRAM SIZE =      1832
 *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
            C
            C---------------------------------------------------------------------
            C
 0001             SUBROUTINE COLUMN (RW,CL,U)
            C
            C   Column finds the shutter column (u) for the given point by looking
            C   at the images the point appeared in.  It is assumed that pattern two
            C   illuminates the right half of the shutter, etc..
            C
            C   Input : RW -  camera row  for point.
            C           CL -  camera column for point.
            C   Output: U  -  shutter column number for point
            C
            C---------------------------------------------------------------------
            C
            C  Include GLOBAL and PATTERN commons here
            C
            C  "GLOBAL" common (1,15)
            C
 0002             REAL SDIST, STHETA, SPHI
            C
 0003             REAL SSEP, SWIDTH, SHIGH
            C
 0004             INTEGER SROW, SCOL
            C       Dimensions of the shutter grid
            C
 0005             REAL CDIST, CTHETA, CPHI
            C
 0006             REAL CSEP, CWIDTH, CHIGH
            C
 0007             INTEGER CROW, CCOL
            C       Dimensions of the camera grid
            C
 0008             INTEGER NIMAGE
            C       Number of patterns needed.
            C
 0009             COMMON/GLOBAL/
                1     SDIST, STHETA, SPHI, SSEP, SWIDTH, SHIGH, SROW, SCOL,
                2     CDIST, CTHETA, CPHI, CSEP, CWIDTH, CHIGH, CROW, CCOL,
                3     NIMAGE
            C
            C  "Patterns" common (26,37)
            C
 0010             INTEGER PAT1(128,128), PAT2(128,128), PAT3(128,128),
                1         PAT4(128,128), PAT5(128,128), PAT6(128,128),
                2         PAT7(128,128), PAT8(128,128)
            C          Contain the image data.
            C
 0011             COMMON/PATS/
                1     PAT1, PAT2, PAT3, PAT4, PAT5, PAT6,
                2     PAT7, PAT8
            C
 0012             INTEGER RW,CL,U
            C
```

```
            C
0013                  U          = PAT2(RW,CL) * ( 2 ** (NIMAGE-2))
                   1             + PAT3(RW,CL) * ( 2 ** (NIMAGE-3))
                   2             + PAT4(RW,CL) * ( 2 ** (NIMAGE-4))
                   3             + PAT5(RW,CL) * ( 2 ** (NIMAGE-5))
                   4             + PAT6(RW,CL) * ( 2 ** (NIMAGE-6))
                   5             + PAT7(RW,CL) * ( 2 ** (NIMAGE-7))
                   6             + PAT8(RW,CL) * ( 2 ** (NIMAGE-8))
                   7             + 1
0014                  RETURN
            C
0015                  END
 *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
 *OPTIONS IN EFFECT*  NAME = COLUMN  , LINECNT =      57
 *STATISTICS*    SOURCE STATEMENTS =        15,PROGRAM SIZE =       986
 *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
                C
                C---------------------------------------------------------------
                C
 0001                 SUBROUTINE SETCOF(A, B, XS, YS, U, T, L)
                C
                C   Setcoef sets the coefficients on the following three
                C   equations which are used to solve for the point (x,y,z):
                C
                C  (T11-T14x*)x + (T21-T24x*)y + (T31-T34x*)z + (T41-T44x*) = 0
                C  (T12-T14y*)x + (T22-T24y*)y + (T32-T34y*)z + (T42-T44y*) = 0
                C  (L11-L14u )x + (L21-L24u )y + (L31-L34u )z + (L41-L44u ) = 0
                C
                C  Parameters:
                C
                C    Input:  XS - x* in the equations
                C            YS - y* in the equations
                C            U  - column of the shutter
                C            T  - Transformation matrix for camera geometry
                C            L  - Equation of the plane for each shutter column
                C
                C    Output:  A, B:  Least square matrices
                C
                C------------------------------------------------------------------
                C
 0002                 REAL XS, YS, U, T(4,4), L(4,4), A(3,3), B(3)
                C
                C  Begin...
 0003                 A(1,1) = T(1,1) - T(1,4) * XS
 0004                 A(1,2) = T(2,1) - T(2,4) * XS
 0005                 A(1,3) = T(3,1) - T(3,4) * XS
 0006                 B(1)   = - ( T(4,1) - T(4,4) * XS )
                C
 0007                 A(2,1) = T(1,2) - T(1,4) * YS
 0008                 A(2,2) = T(2,2) - T(2,4) * YS
 0009                 A(2,3) = T(3,2) - T(3,4) * YS
 0010                 B(2)   = - ( T(4,2) - T(4,4) * YS )
                C
 0011                 A(3,1) = L(1,1) - L(1,4) * U
 0012                 A(3,2) = L(2,1) - L(2,4) * U
 0013                 A(3,3) = L(3,1) - L(3,4) * U
 0014                 B(3)   = - ( L(4,1) - L(4,4) * U )
                C
                C
 0015                 RETURN
 0016                 END
  *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
  *OPTIONS IN EFFECT*  NAME = SETCOF   , LINECNT =       57
  *STATISTICS*    SOURCE STATEMENTS =        16,PROGRAM SIZE =      614
  *STATISTICS*  NO DIAGNOSTICS GENERATED

NO STATEMENTS FLAGGED IN THE ABOVE COMPILATIONS.
```

```
            C -----------------------------------------------------------
            C
0001              SUBROUTINE L2(M,N,A,B,X,EPS,I1,I2,I3,M1,QL)
            C
            C -----------------------------------------------------------
            C
            C   THIS SUBROUTINE SOLVES THE LINEAR EQUATIONS AX = B.
            C
            C    A    --  M X N matrix
            C    EPS  --  threshold for the singular vaules.
            C             Those S.V.'s (in ARRAY Q) smaller than EPS * QMAX
            C             are set to zero.
            C    EPS1 --  precision desired for the solution.
            C    I1   --  maximum number of iterations allowed to obtain the
            C             precision desired.
            C    I2, I3 - flags for various printouts.
            C    M1    -- row dimension of matrix A as declared in the calling
            C             routine.
            C    TOL   -- Smallest REAL number in the particular machine used.
            C
            C
            C    REFERENCE: G.H. GOLUB & C. REINSCH
            C               NUMER. MATH. 14, 403-420, (1970).
            C               FORTRAN BY ALAN CLINE, NCAR, BOULDER, CO., (1970).
            C
            C   SUBROUTINES CALLED: SVD S1
            C
            C
0002              COMMON/BLKA/LDTA,LCAL,LGC1,LUNOUT,INFL,IM1,IM2,LCOD
0003              COMMON/SUB/U(200,200),V(11,11),Q(11),DX(11),R(200)
0004              REAL A(M1,N),B(M),X(N)
0005              DOUBLE PRECISION S
0006              DATA EPS1/1.E-05/
            C
            C   begin
            C
            C  LUNOUT ALONE INITIALIZED HERE TO MINIMIZE INTERFACE
            C    WITH REST OF PROGRAM AND MINIMIZE CHANGES TO CANNED PROGRAM.
            C
0007              LUNOUT=6
            C
0008              CALL SVD(A,M,N,1,1,M1)
            C
0009              QL=Q(1)
0010              QH=Q(1)
            C
0011              DO 2 J=2,N
0012                 QH=AMAX1(Q(J),QH)
0013                 QL=AMIN1(Q(J),QL)
0014          2   CONTINUE
            C
0015              QL=QH/QL
            C
0016              IF (I2.EQ.1) WRITE(LUNOUT,104) QL,(Q(I),I=1,N)
```

```
      QH=QH*EPS
      DO 3 J=1,N
         IF(Q(J).LT.QH)Q(J)=0.
    3 CONTINUE
C
      IF (I2.EQ.1) WRITE(LUNOUT,104) QL,(Q(I),I=1,N)
C
      CALL S1(X,B,M,N)
C
      IF(I3.NE.0)WRITE(LUNOUT,100)(X(J),J=1,N)
      IF(I1.LE.0)RETURN
C
C  iterative improvement
C
      XNORM=0.
      DO 5 J=1,N
         XNORM=AMAX1(XNORM,ABS(X(J)))
    5 CONTINUE
C
      IF(XNORM.EQ.0.)RETURN
C
      DO 10 K=1,I1
C
         DO 7 I=1,M
            S=0.
            DO 6 J=1,N
               S=S+DBLE(A(I,J))*DBLE(X(J))
    6       CONTINUE
            R(I) = DBLE( B(I) ) - S
    7    CONTINUE
C
         CALL S1(DX,R,M,N)
C
         DXNORM=0.
         DO 8 J=1,N
            T=X(J)
            X(J)=X(J)+DX(J)
            DXNORM=AMAX1(DXNORM,ABS(X(J)-T))
    8    CONTINUE
C
         IF (I3 .NE. 0) WRITE(LUNOUT,100) (X(J),J=1,N)
         IF ( .NOT. ( DXNORM .LE. EPS1*XNORM ) ) GO TO 11
         IF (I3 .NE. 0) WRITE(LUNOUT,102) K
C
         RETURN
C
   11    CONTINUE
   10 CONTINUE
C
      WRITE(LUNOUT,101)I1
      RETURN
C
  100 FORMAT(1H0,(5E12.3))
  101 FORMAT(' CONVERGENCE IN ITERATIVE IMPROVEMENT DID NOT OCCUR IN ',
     1       I2,' ITERATIONS.')
```

```
  0055          102  FORMAT(I3,' STEPS OF ITERATIVE REFINEMENT WERE TAKEN.')                97.000
  0056          103  FORMAT(' ITERATION',I3,' L2 NORM OF RESIDUAL = ',E12.3,/,               98.000
                    1         (5E12.3))                                                     99.000
  0057          104  FORMAT(/,' CONDITION OF MATRIX = ',E12.3,/,                           100.000
                    1         ' SINGULAR VALUES ',/,(5E12.3))                              101.000
               C                                                                           102.000
  0058               END                                                                   103.000
   *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
   *OPTIONS IN EFFECT*  NAME = L2       , LINECNT =       57
   *STATISTICS*    SOURCE STATEMENTS =       58,PROGRAM SIZE =      2284
   *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
C
C
C ------------------------------------------------------------------
C
C
      SUBROUTINE SVD(A,M,N,IU,IV,M1)
C
C-----------------------------------------------------------------
C
C     COMPUTATION OF THE SINGULAR VALUES AND COMPLETE ORTHOGONAL
C       DECOMPOSITION OF A RECTANGULAR MATRIX A.
C
C           A = U * DIAG(G) * V(TRANSPOSE)
C           U(TRANSPOSE) * U = V(TRANSPOSE) * V = I
C
C     IF IU=1, THEN U IS COMPUTED.
C     IF IV=1, THEN V IS COMPUTED.
C
C     dimensions  A(M,N),U(M,N),V(N,N),G(N)
C
      REAL A(M1,N)
      COMMON/SUB/U(200,200),V(11,11),Q(11),E(200)
      DATA TOL/1.102E-30/
C
C   begin
C
      EPS=4.768E-07
C
      DO 1 I=1,M
         DO 1001 J=1,N
            U(I,J)=A(I,J)
 1001    CONTINUE
 1    CONTINUE
C
      G=0.
      X=0.
C
      DO 11 I=1,N
         E(I)=G
         S=0.
         L=I+1
C
         DO 2 J=I,M
            S=S+U(J,I)*U(J,I)
 2       CONTINUE
C
C  BEGIN HOUSEHOLDER'S REDUCTION TO BIDIAGONAL FORM.
C
         G=0.
         IF(.NOT.(S.GE.TOL))GO TO 5
            F=U(I,I)
            G=SQRT(S)
            IF(F.GE.O.)G=-G
            H=F*G-S
            U(I,I)=F-G
```

```
                 IF(.NOT.(L.LE.N))GO TO 1005
                    DO 4 J=L,N
                       S=0.
                       DO 3 K=I,M
                          S=S+U(K,I)*U(K,J)
         3             CONTINUE
                       F=S/H
                       DO 1004 K=I,M
                          U(K,J)=U(K,J)+F*U(K,I)
         1004          CONTINUE
         4          CONTINUE
         1005    CONTINUE
         5    CONTINUE
        C
              Q(I)=G
              G=0.
              IF(.NOT.(L.LE.N))GO TO 10
                 S=0.
                 DO 6 J=L,N
                    S=S+U(I,J)*U(I,J)
         6       CONTINUE
                 IF(.NOT.(S.GE.TOL))GO TO 1010
                    F=U(I,I+1)
                    G=SQRT(S)
                    IF(F.GE.0.)G=-G
                    H=F*G-S
                    U(I,I+1)=F-G
                  . DO 7 J=L,N
                       E(J)=U(I,J)/H
         7          CONTINUE
                    DO 9 J=L,M
                       S=0.
                       DO 8 K=L,N
                          S=S+U(J,K)*U(I,K)
         8             CONTINUE
                       DO 1009 K=L,N
                          U(J,K)=U(J,K)+S*E(K)
         1009          CONTINUE
         9          CONTINUE
         1010    CONTINUE
         10   CONTINUE
              Y=ABS(Q(I))+ABS(E(I))
              IF(Y.GT.X)X=Y
         11 CONTINUE
        C
        C   END HOUSEHOLDER'S REDUCTION
        C
        C BEGIN COMPUTATION OF VECTOR V
        C
            IF(.NOT.(IV.EQ.1))GO TO 18
              DO 17 I1=1,N
                 I=N+1-I1
                 IF(.NOT.(L.LE.N))GO TO 165
                    IF(.NOT.(G.NE.0.))GO TO 15
```

```
                DO 12 J=L,N
                   V(J,I)=U(I,J)/H
      12        CONTINUE
C
                DO 14 J=L,N
                   S=0.
                   DO 13 K=L,N
                      S=S+U(I,K)*V(K,J)
      13            CONTINUE
                   DO 1014 K=L,N
                      V(K,J)=V(K,J)+S*V(K,I)
      1014         CONTINUE
      14        CONTINUE
C
      15     CONTINUE
C
             DO 16 J=L,N
                V(I,J)=0.
                V(J,I)=0.
      16     CONTINUE
C
      165 CONTINUE
C
          V(I,I)=1.
          G=E(I)    .
          L=I
      17   CONTINUE
      18 CONTINUE
C
C      END COMPUTATION OF V
C
C     BEGIN COMPUTATION OF VECTOR U
C
        IF(.NOT.(IU.EQ.1))GO TO 26
           DO 25 I1=1,N
              I=N+1-I1
              L=I+1
              G=Q(I)
              IF(.NOT.(L.LE.N))GO TO 195
                 DO 19 J=L,N
                    U(I,J)=0.
      19         CONTINUE
      195     CONTINUE
C
              IF(.NOT.(G.NE.0.))GO TO 23
              H=U(I,I)*G
              IF(.NOT.(L.LE.N))GO TO 215
                 DO 21 J=L,N
                    S=0.
                    DO 20 K=L,M
                       S=S+U(K,I)*U(K,J)
      20            CONTINUE
                    F=S/H
                    DO 1021 K=I,M
                       U(K,J)=U(K,J)+F*U(K,I)
```

```
0120        1021              CONTINUE
0121        21              CONTINUE
0122        215           CONTINUE
           C
0123                      DO 22 J=I,M
0124                         U(J,I)=U(J,I)/G
0125        22            CONTINUE
           C
0126                      GO TO 1025
           C              ELSE
0127        23            CONTINUE
           C
0128                      DO 24 J=I,M
0129                         U(J,I)=O.
0130        24            CONTINUE
           C
0131        1025       CONTINUE
0132                   U(I,I)=U(I,I)+1.
0133        25         CONTINUE
0134        26      CONTINUE
           C
           C        END COMPUTATION OF U
           C
           C        BEGIN DIAGONALIZATION OF THE BIDIAGONAL FORM G USING QR ALGORITHM
           C
0135                 EPS=EPS*X
0136                 DO 41 K1=1,N
0137                    K=N+1-K1
           C            BEGIN BIG LOOP
0138        27          CONTINUE
           C
0139                       DO 28 L1=1,K
0140                          L=K+1-L1
0141                          IF(.NOT.(ABS(E(L)).GT.EPS))GO TO 32
0142                          IF(.NOT.(ABS(Q(L-1)).GT.EPS))GO TO 29
0143        28             CONTINUE
           C
0144        29             CONTINUE
           C
0145                       C=O.
0146                       S=1.
0147                       L1=L-1
0148                       DO 31 I=L,K
0149                          F=S*E(I)
0150                          E(I)=C*E(I)
0151                          IF(ABS(F).LE.EPS)GO TO 32
0152                          G=Q(I)
0153                          H=SQRT(F*F+G*G)
0154                          Q(I)=H
0155                          C=G/H
0156                          S=-F/H
0157                          IF(.NOT.(IU.EQ.1))GO TO 1031
0158                             DO 30 J=1,M
0159                                Y=U(J,I)
```

```
                  U(J,L1)=Y*C+Z*S
                  U(J,I)=-Y*S+Z*C
   30          CONTINUE
 1031       CONTINUE
   31    CONTINUE
   32    CONTINUE
C
         Z=Q(K)
         IF(L.EQ.K)GO TO 39
C
C        ACCELERATION SCHEME FOR CONVERGENCE BY SHIFTING TO
C           DISPLACED EIGENVALUES
C
         X=Q(L)
         Y=Q(K-1)
         G=E(K-1)
         H=E(K)
         F=((Y-Z)*(Y+Z)+(G-H)*(G+H))/(2.*H*Y)
         G=SQRT(F*F+1.)
         IF(.NOT.(F.LT.O.))GO TO 33
            F=((X-Z)*(X+Z)+H*(Y/(F-G)-H))/X
            GO TO 34
   33    CONTINUE
            F=((X-Z)*(X+Z)+H*(Y/(F+G)-H))/X
   34    CONTINUE
C
         C=1.
         S=1.
         L2=L+1
         DO 38 I=L2,K
            G=E(I)
            Y=Q(I)
            H=S*G
            G=C*G
            Z=SQRT(F*F+H*H)
            E(I-1)=Z
            C=F/Z
            S=H/Z
            F=X*C+G*S
            G=-X*S+G*C
            H=Y*S
            Y=Y*C
            IF(.NOT.(IV.EQ.1))GO TO 36
               DO 35 J=1,N
                  X=V(J,I-1)
                  Z=V(J,I)
                  V(J,I-1)=X*C+Z*S
                  V(J,I)=-X*S+Z*C
   35          CONTINUE
   36       CONTINUE
C
            Z=SQRT(F*F+H*H)
            Q(I-1)=Z
            C=F/Z
            S=H/Z
```

```
0209                 F=C*G+S*Y
0210                 X=-S*G+C*Y
0211                 IF(.NOT.(IU.EQ.1))GO TO 1038
0212                    DO 37 J=1,M
0213                       Y=U(J,I-1)
0214                       Z=U(J,I)
0215                       U(J,I-1)=Y*C+Z*S
0216                       U(J,I)=-Y*S+Z*C
0217       37           CONTINUE
0218       1038      CONTINUE
0219       38     CONTINUE
          C
0220                 E(L)=0.
0221                 E(K)=F
0222                 Q(K)=X
0223              GO TO 27
          C
          C     END BIG LOOP
          C
0224       39     CONTINUE
          C
          C       CHANGE SIGN OF NEGATIVE SQUARE ROOTS
          C
0225              IF(Z.GE.0.)GO TO 41
0226              Q(K)=-Z
0227              IF(IV.NE.1)GO TO 41
          C
0228              DO 40 J=1,N
0229                 V(J,K)=-V(J,K)
0230       40     CONTINUE
          C
0231       41   CONTINUE
          C
0232           RETURN
0233           END
  *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
  *OPTIONS IN EFFECT*  NAME = SVD     , LINECNT =      57
  *STATISTICS*    SOURCE STATEMENTS =      233,PROGRAM SIZE =      5466
  *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
            C
            C
            C-----------------------------------------------------------------
            C
            C
0001                SUBROUTINE S1(X,B,M,N)
            C
            C-----------------------------------------------------------------
            C
            C       COMPUTES SOLUTION VECTOR X.
            C            X = V * DIAG(G INVERSE) * U(TRANSPOSE)
            C            A = U * DIAG(G) * V(TRANSPOSE)
            C
0002                REAL X(N),B(M),Y(11)
0003                COMMON/SUB/U(200,200),V(11,11),Q(11),DX(11),R(200)
            C
0004                DO 3 J=1,N
0005                   IF(.NOT.(Q(J).LE.O))GO TO 1
0006                      Y(J)=O.
0007                      GO TO 3
0008         1         CONTINUE
            C          ELSE
0009                      S=O.
0010                      DO 2 I=1,M
0011                         S=S+U(I,J)*B(I)
0012         2            CONTINUE
0013                      Y(J)=S/Q(J)
0014         3      CONTINUE
            C
0015                DO 6 I=1,N
0016                   S=O.
0017                   DO 5 J=1,N
0018                      S=S+V(I,J)*Y(J)
0019         5         CONTINUE
0020                   X(I)=S
0021         6      CONTINUE
            C
0022                RETURN .
0023                END
 *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
 *OPTIONS IN EFFECT*  NAME = S1      , LINECNT =       57
 *STATISTICS*    SOURCE STATEMENTS =       23,PROGRAM SIZE =       804
 *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
               C
               C-------------------------------------------------------------------
               C
               C
 0001                 SUBROUTINE CLRTBL(MATRIX,NROW,NCOL)
               C
               C-------------------------------------------------------------------
               C
               C
               C   THIS SUBROUTINE CLEARS A 2-D TABLE (CALLED 'MATRIX') OF
               C      DIMENSION NROW X NCOL.
               C
 0002                 REAL MATRIX(NROW,NCOL)
               C
               C   begin
               C
 0003                 IF(.NOT.((NROW.LT.1).OR.(NCOL.LT.1)))GO TO 1001
 0004                      STOP
               C
 0005            1001 CONTINUE
 0006                 DO 1002 J=1,NCOL
 0007                      DO 1003 I=1,NROW
 0008                           MATRIX(I,J)=0.
 0009            1003      CONTINUE
 0010            1002 CONTINUE
               C
 0011                 RETURN
               C
 0012                 END
  *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
  *OPTIONS IN EFFECT*  NAME = CLRTBL   , LINECNT =       57
  *STATISTICS*    SOURCE STATEMENTS =        12,PROGRAM SIZE =       544
  *STATISTICS*  NO DIAGNOSTICS GENERATED
```

```
                C
                C
                C -------------------------------------------------------------
                C
                C
0001                  SUBROUTINE CLRVC(VECTOR,N)
                C
                C -------------------------------------------------------------
                C
                C
                C    THIS SUBROUTINE CLEARS AN ARRAY (CALLED 'VECTOR') OF
                C       DIMENSION N.
                C
0002                  REAL VECTOR(N)
                C
                C    begin
                C
0003                  IF(.NOT.(N.LT.1))GO TO 1001
0004                       STOP
0005             1001 CONTINUE
                C
0006                  DO 1002 I=1,N
0007                       VECTOR(I)=0.
0008             1002 CONTINUE
                C
0009                  RETURN
                C
0010                  END
 *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
 *OPTIONS IN EFFECT*  NAME = CLRVC    , LINECNT =       57
 *STATISTICS*    SOURCE STATEMENTS =       10,PROGRAM SIZE =      430
 *STATISTICS*  NO DIAGNOSTICS GENERATED

NO STATEMENTS FLAGGED IN THE ABOVE COMPILATIONS.
```

8.5. Analysis Program

LINE NUMBER B P C I   STMT #   S O U R C E   P R O G R A M

```
                 V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
                 |PROGRAM Phan_Cal ( INPUT, OUTPUT ) ;
                 |
                 |  CONST
                 |
                 |    u_max = 64 ;
                 |    v_max = 64 ;
                 |
                 |
                 |  VAR
                 |
                 |    Pha_f,
                 |    Eye_f : TEXT ;
                 |
                 |    Data : ARRAY (. 1..u_max, 1..v_max, 1..8 .) of REAL ;
                 |
                 |    Xs, Ys, U, V : INTEGER ;
                 |    X1, Y1, Z1,
                 |    X2, Y2, Z2 : REAL ;
                 |
                 |    Dist,
                 |    Dist_sqr,
                 |    Sum,
                 |    Sum_sqr :  REAL ;
                 |
                 |    Num_data : INTEGER ;
                 |
                 |    Mean,
                 |    Stan_dev : REAL ;
                 |
                 | BEGIN
                 |
                 |  { Open files }
                 |
               1 |   RESET ( Pha_f, 'UNIT=2' ) ;
               2 |   RESET ( Eye_f, 'UNIT=5' ) ;
                 |
                 |  { Initialize }
                 |
       1       3 |   FOR U := 1 TO u_max DO
       2       4 |       FOR V := 1 TO v_max DO
               5 |           Data(.u, v, 1.) := 0 ;
                 |
                 |  { read intersection points from phantom file }
                 |
       1       6 |   WHILE NOT EOF ( Pha_f ) DO
       1         |     BEGIN
  1    1       7 |       READLN ( Pha_f, U, V, Xs, Ys, X1, Y1, Z1 ) ;
  1    1       8 |       Data (.U, V, 1.) := Xs ;
  1    1       9 |       Data (.U, V, 2.) := Ys ;
  1    1      10 |       Data (.U, V, 3.) := X1 ;
  1    1      11 |       Data (.U, V, 4.) := Y1 ;
  1    1      12 |       Data (.U, V, 5.) := Z1
                 |     END ;
```

LINE NUMBER B P C I STMT #   S O U R C E   P R O G R A M

```
                            V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
   51.000
   52.000                     { read calculated points from eye file }
   53.000
   54.000          1     13    WHILE NOT EOF ( Eye_f ) DO
   55.000          1             BEGIN
   56.000 1        1     14        READLN ( Eye_f, U, Xs, Ys, X1, Y1, Z1 ) ;
   56.100 1        1
   56.200 1        1               { find V }
   56.300 1        1
   56.400 1        1     15        V := 0 ;
   56.500 1        2     16        REPEAT  V := V + 1
   56.600 1        1     17        UNTIL ( V > v_max ) | (( Data(.U, V, 1.) = Xs ) & ( Data(.U, V, 2.) = Ys )) ;
   56.800 1        1
   56.860 1      1 1     18        IF V <= v_max
   56.920 1      1 1                  THEN BEGIN
   57.000 2      1 1     19                  Data (.U, V, 6.) := X1 ;
   58.000 2      1 1     20                  Data (.U, V, 7.) := Y1 ;
   59.000 2      1 1     21                  Data (.U, V, 8.) := Z1
   59.500 1      1 1                       END
   59.700 1      1 1     22           ELSE WRITELN ( '*** Not Found ***' )
   59.800 1      1 1
   60.000                          END ;
   61.000
   62.000                     { find average and standard deviation of distances between    }
   63.000                     { intersection points from phantom file and calculated points }
   64.000                     { from eye file                                               }
   65.000
   65.100                23    WRITELN ( '    ----- intersection -----       ----- computed point ---- ' ) ;
   65.300                24    WRITELN ( '      X1          Y1         Z1           X2          Y2          Z2        distance  ' ) ;
   65.600                25    WRITELN ;
   65.800
   66.000                26    Num_data := 0 ;
   67.000                27    Sum := 0 ;
   68.000                28    Sum_sqr := 0 ;
   69.000
   70.000          1     29    FOR U := 1 TO u_max DO
   71.000          2     30       FOR V := 1 TO  v_max DO
   72.000        1 2     31           IF Data (.U, V, 1.) > 0  THEN
   73.000        1 2                    BEGIN
   74.000        1 2
   75.000 1      1 2     32               X1 := Data (.U, V, 3.) ;  X2 := Data (.U, V, 6.) ;
   76.000 1      1 2     34               Y1 := Data (.U, V, 4.) ;  Y2 := Data (.U, V, 7.) ;
   77.000 1      1 2     36               Z1 := Data (.U, V, 5.) ;  Z2 := Data (.U, V, 8.) ;
   78.000 1      1 2
   79.000 1      1 2     38               Dist_sqr :=  SQR ( X1 - X2 ) + SQR ( Y1 - Y2 ) + SQR ( Z1 - Z2 ) ;
   80.000 1      1 2     39               Sum_sqr := Sum_sqr + Dist_sqr ;
   81.000 1      1 2
   82.000 1      1 2     40               Dist := SQRT ( Dist_sqr ) ;
   83.000 1      1 2     41               Sum := Sum + Dist ;
   84.000 1      1 2     42               WRITELN ( X1 :10:3, Y1 :10:3, Z1 :10:3, X2 :12:3, Y2 :10:3, Z2 :10:3, Dist :12:3 ) ;
   85.000 1      1 2
   86.000 1      1 2     43               Num_data := Num_data + 1
   87.000 1      1 2
```

```
LINE NUMBER B P C I  STMT #                       S O U R C E   P R O G R A M
                             V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
   88.000 1    1 2          |
   89.000      1 2          |          END ;
   90.000      1 2          |
   91.000             44    |  Mean := Sum / Num_data ;
   92.000             45    |  Stan_dev := SQRT ( Sum_sqr / Num_data - Mean * Mean ) ;
   93.000                   |
   93.500             46    |  WRITELN ;  WRITELN ;
   94.000             48    |  WRITELN ( ' --- Average Distance   : ', mean :7 :3 ) ;
   94.500             49    |  WRITELN ( ' --- Standard Deviation : ', stan_Dev :7 :3 )
   95.000                   |
   96.000                   | END.
```

NO COMPILER DETECTED ERRORS

OPTIONS IN EFFECT: MARGINS(1,100), NOSEQUENCE, LANGLVL(EXTENDED), LINECOUNT(60), PAGEWIDTH(132), CHECK,
DEBUG, GOSTMT, SOURCE, WARNING

SOURCE LINES: 117; COMPILE TIME: 0.13 SECONDS; COMPILE RATE: 53867 LPM

SOURCE LINES. 117; TRANSLATE TIME: 0.12 SECONDS; TRANSLATE RATE: 59338 LPM

TOTAL TIME: 0.31 SECONDS; TOTAL RATE: 22308 LPM

```
B P C I  STMT #                      S O U R C E   P R O G R A M
                 V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
                 PROGRAM Phan_Cal ( INPUT, OUTPUT ) ;

                   CONST

                     u_max = 64 ;
                     v_max = 64 ;


                   VAR

                     Pha_f,
                     Eye_f : TEXT ;

                     Data : ARRAY (. 1..u_max, 1..v_max, 1..8 .) of REAL ;

                     Xs, Ys, U, V : INTEGER ;
                     X1, Y1, Z1,
                     X2, Y2, Z2 : REAL ;

                     Dist,
                     Dist_sqr,
                     Sum,
                     Sum_sqr :  REAL ;

                     Num_data : INTEGER ;

                     Mean,
                     Stan_dev : REAL ;

                  BEGIN

                    { Open files }

          1         RESET ( Pha_f, 'UNIT=2' ) ;
          2         RESET ( Eye_f, 'UNIT=5' ) ;

                    { Initialize }

      1   3         FOR U := 1 TO u_max DO
      2   4             FOR V := 1 TO v_max DO
          5                 Data(.u, v, 1.) := 0 ;

                    { read intersection points from phantom file }

      1   6         WHILE NOT EOF ( Pha_f ) DO
      1               BEGIN
1     1   7             READLN ( Pha_f, U, V, Xs, Ys, X1, Y1, Z1 ) ;
1     1   8             Data (.U, V, 1.) := Xs ;
1     1   9             Data (.U, V, 2.) := Ys ;
1     1  10             Data (.U, V, 3.) := X1 ;
```

```
LINE NUMBER B P C I  STMT #                        S O U R C E   P R O G R A M

                           V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
   51.000                 |
   52.000                 |  { read calculated points from eye file }
   53.000                 |
   54.000       1     13  |   WHILE NOT EOF ( Eye_f ) DO
   55.000       1         |     BEGIN
   56.000 1     1     14  |     . READLN ( Eye_f, U, Xs, Ys, X1, Y1, Z1 ) ;
   56.100 1     1         |
   56.200 1     1         |       { find V }
   56.300 1     1         |
   56.400 1     1     15  |       V := O ;
   56.500 1     2     16  |       REPEAT  V := V + 1
   56.600 1     1     17  |       UNTIL ( V > v_max ) | (( Data(.U, V, 1.) = Xs ) & ( Data(.U, V, 2.) = Ys )) ;
   56.800 1     1         |
   56.860 1   1 1     18  |       IF V <= v_max
   56.920 1   1 1         |          THEN BEGIN
   57.000 2   1 1     19  |                 Data (.U, V, 6.) := X1 ;
   58.000 2   1 1     20  |                 Data (.U, V, 7.) := Y1 ;
   59.000 2   1 1     21  |                 Data (.U, V, 8.) := Z1
   59.500 1   1 1         |               END
   59.700 1   1 1     22  |          ELSE WRITELN ( '*** Not Found ***' )
   59.800 1   1 1         |
   60.000                 |     END ;
   61.000                 |
   62.000                 |  { find average and standard deviation of distances between    }
   63.000                 |  { intersection points from phantom file and calculated points }
   64.000                 |  { from eye file                                               }
   65.000                 |
   65.100             23  |   WRITELN ( '    ----- intersection -----         ----- computed point ---- ' ) ;
   65.300             24  |   WRITELN ( '     X1         Y1         Z1          X2         Y2         Z2        distance  ' ) ;
   65.600             25  |   WRITELN ;
   65.800                 |
   66.000             26  |   Num_data := O ;
   67.000             27  |   Sum := O ;
   68.000             28  |   Sum_sqr := O ;
   69.000                 |
   70.000       1     29  |   FOR U := 1 TO u_max DO
   71.000       2     30  |      FOR V := 1 TO  v_max DO
   72.000     1 2     31  |          IF Data (.U, V, 1.) > O  THEN
   73.000     1 2         |             BEGIN
   74.000     1 2         |
   75.000 1   1 2     32  |                X1 := Data (.U, V, 3.) ;  X2 := Data (.U, V, 6.) ;
   76.000 1   1 2     34  |                Y1 := Data (.U, V, 4.) ;  Y2 := Data (.U, V, 7.) ;
   77.000 1   1 2     36  |                Z1 := Data (.U, V, 5.) ;  Z2 := Data (.U, V, 8.) ;
   78.000 1   1 2         |
   79.000 1   1 2     38  |                Dist_sqr :=  SQR ( X1 - X2 ) + SQR ( Y1 - Y2 ) + SQR ( Z1 - Z2 ) ;
   80.000 1   1 2     39  |                Sum_sqr := Sum_sqr + Dist_sqr ;
   81.000 1   1 2         |
   82.000 1   1 2     40  |                Dist := SQRT ( Dist_sqr ) ;
   83.000 1   1 2     41  |                Sum := Sum + Dist ;
   84.000 1   1 2     42  |                WRITELN ( X1 :10:3, Y1 :10:3, Z1 :10:3, X2 :12:3, Y2 :10:3, Z2 :10:3, Dist :12:3 ) ;
   85.000 1   1 2         |
   86.000 1   1 2     43  |                Num_data := Num_data + 1
   87.000 1   1 2         |
```

```
LINE NUMBER B P C I  STMT #                          S O U R C E   P R O G R A M

                            V---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+---1V
     88.000 1    1 2       |
     89.000      1 2       |            END ;
     90.000      1 2       |
     91.000             44 |   Mean := Sum / Num_data ;
     92.000             45 |   Stan_dev := SQRT ( Sum_sqr / Num_data - Mean * Mean ) ;
     93.000                |
     93.500             46 |   WRITELN ;  WRITELN ;
     94.000             48 |   WRITELN ( ' --- Average Distance   : ', mean :7 :3 ) ;
     94.500             49 |   WRITELN ( ' --- Standard Deviation : ', stan_Dev :7 :3 )
     95.000                |
     96.000                | END.
```

NO COMPILER DETECTED ERRORS

OPTIONS IN EFFECT: MARGINS(1,100), NOSEQUENCE, LANGLVL(EXTENDED), LINECOUNT(60), PAGEWIDTH(132), CHECK,
DEBUG, GOSTMT, SOURCE, WARNING

SOURCE LINES: 117;   COMPILE TIME: 0.14 SECONDS;   COMPILE RATE: 50289 LPM

SOURCE LINES: 117; TRANSLATE TIME: 0.12 SECONDS; TRANSLATE RATE: 58792 LPM

TOTAL TIME: 0.33 SECONDS;   TOTAL RATE: 21596 LPM

9. Sample Run

```
#so exec.e
#   $.00,  $13.97T
#empty -5
#Done.
#   $.00,  $13.97T
#$run lamp:eye.obj+lamp:sup.obj.2+naas:imsl 3=-3 4=-4 5=-5 t=3
#Execution begins

 CONDITION OF MATRIX =     0.375E+03
 SINGULAR VALUES
   0.540E+03   0.331E+03   0.390E+03   0.217E+02   0.172E+02
   0.159E+02   0.590E+01   0.302E+01   0.219E+01   0.144E+01
   0.546E+01

 CONDITION OF MATRIX =     0.375E+03
 SINGULAR VALUES
   0.540E+03   0.331E+03   0.390E+03   0.217E+02   0.172E+02
   0.159E+02   0.590E+01   0.302E+01   0.219E+01   0.144E+01
   0.546E+01

 T Matrix determined
    -1.7496386   -0.8302813    0.0            -0.0260207
     0.0814062   -1.3786592    0.0            -0.0421989
    -0.4219891    0.9700564    0.0            -0.0260956
    16.3746338   16.4649811    0.0             1.0000000


 CONDITION OF MATRIX =     0.658E+02
 SINGULAR VALUES
   0.100E+03   0.644E+02   0.770E+02   0.737E+01   0.337E+01
   0.261E+01   0.152E+01

 CONDITION OF MATRIX =     0.658E+02
 SINGULAR VALUES
   0.100E+03   0.644E+02   0.770E+02   0.737E+01   0.337E+01
   0.261E+01   0.152E+01

 L Matrix determined
    -0.3377837    0.0          0.0            -0.0306186
     0.2668607    0.0          0.0            -0.0176777
    -0.1590991    0.0          0.0            -0.0353554
     4.5000000    0.0          0.0             1.0000000

#Execution terminated
#   $.15,  $14.12T
```

2	2	Tmat	3	12
2	3	Geomet	6	15
2	3	Tmat	6	15
2	4	Geomet	8	17
2	4	Tmat	8	17
2	5	Geomet	11	19
2	5	Tmat	11	19
2	6	Geomet	13	21
2	6	Tmat	13	21
2	7	Geomet	16	24
2	7	Tmat	16	24
2	8	Geomet	18	26
2	8	Tmat	18	26
3	1	Geomet	3	9
3	1	Tmat	3	9
3	2	Geomet	6	11
3	2	Tmat	6	11
3	3	Geomet	8	14
3	3	Tmat	8	14
3	4	Geomet	11	16
3	4	Tmat	11	16
3	5	Geomet	14	18
3	5	Tmat	14	18
3	6	Geomet	16	21
3	6	Tmat	16	21
3	7	Geomet	19	23
3	7	Tmat	19	23
3	8	Geomet	21	25
3	8	Tmat	21	25
4	1	Geomet	6	7
4	1	Tmat	6	7
4	2	Geomet	9	10
4	2	Tmat	9	10
4	3	Geomet	11	13
4	3	Tmat	11	13
4	4	Geomet	14	15
4	4	Tmat	14	15
4	5	Geomet	17	18
4	5	Tmat	17	18
4	6	Geomet	20	20
4	6	Tmat	20	20
4	7	Geomet	22	22
4	7	Tmat	22	22
4	8	Geomet	25	25
4	8	Tmat	25	25
5	1	Geomet	9	6
5	1	Tmat	9	6
5	2	Geomet	12	9
5	2	Tmat	12	9
5	3	Geomet	15	11
5	3	Tmat	15	11
5	4	Geomet	18	14
5	4	Tmat	18	14
5	5	Geomet	20	17
5	5	Tmat	20	17
5	6	Geomet	23	19
5	6	Tmat	23	19
5	7	Geomet	26	22

```
       5      8  Tmat       29      24
       6      1  Geomet     12       4
       6      1  Tmat       12       4
       6      2  Geomet     15       7
       6      2  Tmat       15       7
       6      3  Geomet     19      10
       6      3  Tmat       19      10
       6      4  Geomet     22      13
       6      4  Tmat       22      13
       6      5  Geomet     24      16
       6      5  Tmat       24      16
       6      6  Geomet     27      18
       6      6  Tmat       27      18
       6      7  Geomet     30      21
       6      7  Tmat       30      21
  VMAT *** not observed ***             6            8
  GEO  *** not observed ***             6            8
  TMAT *** not observed ***             6            8
       7      1  Geomet     16       2
       7      1  Tmat       16       2
       7      2  Geomet     20       5
       7      2  Tmat       20       5
       7      3  Geomet     23       8
       7      3  Tmat       23       8
       7      4  Geomet     26      11
       7      4  Tmat       26      11
       7      5  Geomet     29      14
       7      5  Tmat       29      14
       7      6  Geomet     32      17
       7      6  Tmat       32      17
  VMAT *** not observed ***             7            7
  GEO  *** not observed ***             7            7
  TMAT *** not observed ***             7            7
  VMAT *** not observed ***             7            8
  GEO  *** not observed ***             7            8
  TMAT *** not observed ***             7            8
  VMAT *** not observed ***             8            1
  GEO  *** not observed ***             8            1
  TMAT *** not observed ***             8            1
       8      2  Geomet     24       3
       8      2  Tmat       24       3
       8      3  Geomet     28       6
       8      3  Tmat       28       6
       8      4  Geomet     31      10
       8      4  Tmat       31      10
  VMAT *** not observed ***             8            5
  GEO  *** not observed ***             8            5
  TMAT *** not observed ***             8            5
  VMAT *** not observed ***             8            6
  GEO  *** not observed ***             8            6
  TMAT *** not observed ***             8            6
  VMAT *** not observed ***             8            7
  GEO  *** not observed ***             8            7
  TMAT *** not observed ***             8            7
  VMAT *** not observed ***             8            8
  GEO  *** not observed ***             8            8
  TMAT *** not observed ***             8            8

  Num_face              3
#Execution terminated
```

```
  length of the side of a cube -- Length
8
  8.0000
  Step in u and v  -- Step
1
       1
 ---- rotation of the cube ----

 Enter axis ( x, y, z, or / ) and angle
x 10
x 10.0000
 Enter axis ( x, y, z, or / ) and angle
y 45
y 45.0000
 Enter axis ( x, y, z, or / ) and angle
z 30
z 30.0000
 Enter axis ( x, y, z, or / ) and angle
/ 0
/      0.0
 ---- Translation of the cube ----

 Enter translation along x, y, and z axis
     e.g.  3.8  10  8.5
0  0  0
                 0.0                 0.0                 0.0


 ---- T matrix ----
   -34.8575208   -15.1447174          0.0    -0.4330128
     3.6250092   -26.2314068          0.0    -0.7499998
    -8.2500031    19.4628063          0.0    -0.5000002
   330.0000000   330.0000000          0.0    20.0000000


 ---- L matrix ----
    -6.7556749    -7.6546563          0.0    -0.6123724
     5.3372140    -4.4194168          0.0    -0.3535533
    -3.1819810     2.4748723          0.0    -0.7071069
    90.0000000    90.0000000          0.0    20.0000000
 VMAT *** not observed ***          1           1
 GEO  *** not observed ***          1           1
 TMAT *** not observed ***          1           1
     1      2  Geomet      1     14
     1      2  Tmat        1     14
     1      3  Geomet      3     16
     1      3  Tmat        3     16
     1      4  Geomet      6     18
     1      4  Tmat        6     18
     1      5  Geomet      8     20
     1      5  Tmat        8     20
     1      6  Geomet     10     22
     1      6  Tmat       10     22
     1      7  Geomet     13     24
     1      7  Tmat       13     24
     1      8  Geomet     15     26
     1      8  Tmat       15     26
     2      1  Geomet     1'     10
```

```
$Log Output: LAMP, 01:32:11 Thu Aug 04/83
#so exec.f
#   $.00,  $13.79T
#empty -2
#Done.
#   $.00,  $13.79T
#empty -3
#Done.
#   $.00,  $13.80T
#empty -4
#Done.
#   $.00,  $13.80T
#r lamp:phan.obj++pascalvslib 2=-2 3=-3 4=-4  t=3
#Execution begins

  Enter Parameters

  ---- shutter parameters ----

   focal point -- r
 20
   focal point -- theta
 30
   focal point -- phi
 45
   shutter plane -- D
4
   shutter plane -- Su
2
   shutter plane -- Sv
2
   shutter plane -- Nu
8
   shutter plane -- Nv
8
 ---- camera parameters ----

   focal point -- r
20
   focal point -- theta
60
   focal point -- phi
60
   camera plane -- D
4
   camera plane -- Sx
2
   camera plane -- Sy
2
   camera plane -- Nx
32
   camera plane -- Ny
32
  Equation of the plane to be mapped -- A, B, C, D
0 0 1 1
           0.0            0.0       1.0000000      1.0000000

 Enter Calibration Parameters
```

```
++++++   1+++++++++2+++++++++3+++++++++4+++++++++5+++++++++6+++++++++7+++++++++8+++++++++9+++++++++0+++++++++1+++++++++2+++++++++3+
--------692303---------------692303---------------692303---------------692303---------------692303---------------692303-------
```

Job number 692303   University of Michigan Terminal System   (Model UL223)   01:35:47 EDT   Thu Aug 01/83

```
MMMMM                   MMMMM     TTTTTTTTTTTTTTTTTTTTTTTTT          SSSSSSSSS
MMMMMM                 MMMMMM     TTTTTTTTTTTTTTTTTTTTTTTTT        SSSSSSSSSSSSS
MMMMMMM               MMMMMMM     TTTTTTTTTTTTTTTTTTTTTTTTT      SSSSSSSSSSSSSSSSS
MMMMMMMM             MMMMMMMM               TTTTT               SSSSSS       SSSSSSS
MMMMMMMMM           MMMMMMMMM               TTTTT              SSSSS           SSSSS
MMMMMMMMMM         MMMMMMMMMM               TTTTT              SSSSS
MMMMM MMMMM       MMMMM MMMMM               TTTTT               SSSSSS
MMMMM  MMMMM     MMMMM  MMMMM               TTTTT                SSSSSSSSSSSS
MMMMM   MMMMM   MMMMM   MMMMM               TTTTT                  SSSSSSSSSSSSS
MMMMM    MMMMMMMMMMM    MMMMM               TTTTT                    SSSSSSSSSSSSS
MMMMM     MMMMMMMMM     MMMMM               TTTTT                            SSSSSSS
MMMMM      MMMMMMM      MMMMM               TTTTT                              SSSSS
MMMMM       MMMMM       MMMMM               TTTTT                              SSSSS
MMMMM        MMM        MMMMM               TTTTT              SSSSS           SSSSS
MMMMM                   MMMMM               TTTTT              SSSSSSS       SSSSSSS
MMMMM                   MMMMM               TTTTT                SSSSSSSSSSSSSSSSS
MMMMM                   MMMMM               TTTTT                  SSSSSSSSSSSSS
MMMMM                   MMMMM               TTTTT                    SSSSSSSSS

LL             AAAAAAAAAA   MM        MM  PPPPPPPPPPP
LL            AAAAAAAAAAAA  MMM      MMM  PPPPPPPPPPPP
LL            AA        AA  MMMM    MMMM  PP        PP
LL            AA        AA  MM MM  MM MM  PP        PP
LL            AA        AA  MM  MMMM  MM  PP        PP
LL            AAAAAAAAAAAA  MM   MM   MM  PPPPPPPPPPPP
LL            AAAAAAAAAAAA  MM        MM  PPPPPPPPPPP
LL            AA        AA  MM        MM  PP
LL            AA        AA  MM        MM  PP
LL            AA        AA  MM        MM  PP
LLLLLLLLLLLL  AA        AA  MM        MM  PP
LLLLLLLLLLLL  AA        AA  MM        MM  PP
```

```
#so exec.a
#   $.01,  $14.13T
#r lamp:analy.obj+*pascalvslib 2=-2 5=-5 t=3
#Execution begins
```

----- intersection -----			----- computed point ----			
X1	Y1	Z1	X2	Y2	Z2	distance
8.948	-3.076	-1.000	8.828	-3.004	-0.538	0.483
7.220	-4.942	-1.000	7.522	-4.298	-0.270	1.019
5.086	-7.246	-1.000	5.144	-7.038	-0.667	0.397
2.382	-10.164	-1.000	2.966	-9.240	-0.326	1.284
-1.153	-13.980	-1.000	0.200	-12.039	0.107	2.613
-5.974	-19.184	-1.000	-6.096	-19.072	-0.437	0.587
-12.937	-26.701	-1.000	-12.493	-25.686	0.233	1.658
9.444	0.077	-1.000	9.174	-0.221	-1.175	0.438
7.928	-1.310	-1.000	8.065	-1.195	-1.036	0.183
6.093	-2.989	-1.000	6.010	-3.000	-0.777	0.238
3.826	-5.064	-1.000	4.282	-4.517	-0.559	0.837
0.955	-7.691	-1.000	0.864	-7.818	-1.149	0.216
-2.800	-11.127	-1.000	-2.309	-10.648	-0.899	0.693
-7.920	-15.812	-1.000	-9.297	-16.882	-0.348	1.861
-15.316	-22.580	-1.000	-16.639	-23.430	0.232	1.997
8.513	1.690	-1.000	8.351	1.611	-0.729	0.325
6.908	0.456	-1.000	6.775	0.324	-1.180	0.260
4.966	-1.037	-1.000	5.303	-0.677	-0.395	0.781
2.566	-2.882	-1.000	2.736	-2.736	-0.906	0.243
-0.473	-5.219	-1.000	-0.999	-5.730	-1.649	0.980
-4.448	-8.274	-1.000	-4.290	-8.059	-0.429	0.630
-9.867	-12.441	-1.000	-12.269	-14.350	-1.379	3.092
-17.696	-18.459	-1.000	-20.681	-20.783	-1.177	3.787
7.582	3.302	-1.000	7.358	3.154	-1.110	0.291
5.888	2.222	-1.000	5.620	2.048	-1.057	0.325
3.839	0.915	-1.000	4.130	1.133	-0.376	0.722
1.307	-0.700	-1.000	1.462	-0.597	-0.930	0.199
-1.901	-2.746	-1.000	-2.332	-3.011	-0.815	0.539
-6.095	-5.421	-1.000	-8.233	-6.826	-1.787	2.677
-11.814	-9.069	-1.000	-14.065	-10.548	-1.840	2.821
-20.075	-14.338	-1.000	-29.069	-20.126	-1.975	10.740
6.651	4.915	-1.000	6.404	4.771	-0.691	0.421
4.869	3.988	-1.000	4.678	3.873	-0.671	0.398
2.712	2.867	-1.000	2.493	2.766	-1.256	0.351
0.047	1.482	-1.000	-0.533	1.197	-1.326	0.723
-3.328	-0.273	-1.000	-3.147	-0.200	-0.578	0.465
-7.742	-2.568	-1.000	-8.830	-3.108	-1.517	1.320
-13.761	-5.697	-1.000	-18.048	-7.889	-1.730	4.870
-22.454	-10.217	-1.000	-37.726	-17.993	-4.267	17.446
5.720	6.527	-1.000	5.717	6.452	-0.472	0.533
3.849	5.755	-1.000	4.055	5.770	-0.509	0.533
1.584	4.819	-1.000	1.107	4.641	-1.133	0.527
-1.213	3.663	-1.000	-2.072	3.351	-1.301	0.963
-4.756	2.200	-1.000	-4.841	2.122	-0.703	0.318
-9.389	0.285	-1.000	-10.885	-0.224	-1.766	1.756
-15.707	-2.325	-1.000	-20.841	-4.262	-2.292	5.637
4.788	8.140	-1.000	4.823	8.003	-0.345	0.670
2.829	7.521	-1.000	2.560	7.417	-0.919	0.300
0.457	6.771	-1.000	0.214	6.724	-1.130	0.280
-2.472	5.845	-1.000	-2.951	5.788	-1.414	0.636
-6.184	4.672	-1.000	-7.456	4.455	-1.819	1.529
-11.037	3.138	-1.000	-14.380	2.408	-2.441	3.714
1.809	9.287	-1.000	2.062	9.167	-0.415	0.648

-0.670	8.723	-1.000	-0.970	8.707	-1.171	0.346
-3.732	8.027	-1.000	-4.269	7.926	-1.069	0.551

```
 --- Average Distance   :   1.597
 --- Standard Deviation :   2.780
#Execution terminated
#   $.04,  $14.17T
:mts
#   $.13,  $14.32T
#copy -log2 'print*
```