

**SOFTWARE SUPPORT FOR THE STEREO CAMERA:
PHANTOM AND MEASUREMENT PROGRAMS¹**

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**

*THE UNIVERSITY OF MICHIGAN
ENGINEERING LIBRARY*

¹Supported in part by USAF/SAM via SCEEE Contract No. F-33616-78-D-0817 Task 81 and by CRIM via AFOSR Contract No. F49620-82-C-0089.

ENGM
VMR
1619

TABLE OF CONTENTS

1.	Introduction	1
2.	The Phantom Program	2
2.1.	The Coordinate System	2
2.2.	Program Data	3
2.3.	Set Up of the Projector and the Camera	5
2.4.	The Phantom Program Algorithm	5
2.5.	Calibration	6
2.6.	The Phantom Program	7
2.6.1.	Read_Parameters	8
2.6.2.	Read_Cal_Para	8
2.6.3.	Initialize	8
2.6.4.	Find_Tmat	8
2.6.5.	Find_Lmat	9
2.6.6.	Find_Beam_Eqs	9
2.6.7.	Find_Ray_Eqs	10
2.6.8.	Find_Beam_Plane	10
2.6.9.	Map_to_Camera	11
2.6.10.	Map_to_Cam_Tmat	11
2.6.11.	Map_to_Cam_Geo	12
2.6.12.	Make_Image_Pattern	12
2.6.13.	Output_Pattern	13
2.6.14.	Find_Cal_Face	13
2.6.15.	Output_Cube_Image	14
2.6.16.	Main Program	14
3.	Surface Mapping Program	15
3.1.	Init	15
3.2.	Calin	15
3.3.	Imgin	16
3.4.	Tmat	16
3.5.	Lmat	16
3.6.	Map	17

3.7. Column	17
3.8. Setcof	17
3.9. Space Coding	18
4. The Analysis Program	19
5. General Description of the Program Files	20
6. Simulations	21
7. References	25
8. Program Listings	26
8.1. Phantom Program	26
8.2. Eye.Cal Program Listing	27
8.3. Eye.Tlmat Program Listing	28
8.4. Support Program Listing	29
8.5. Analysis Program Listing	30
9. Sample Run	31

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 the 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.

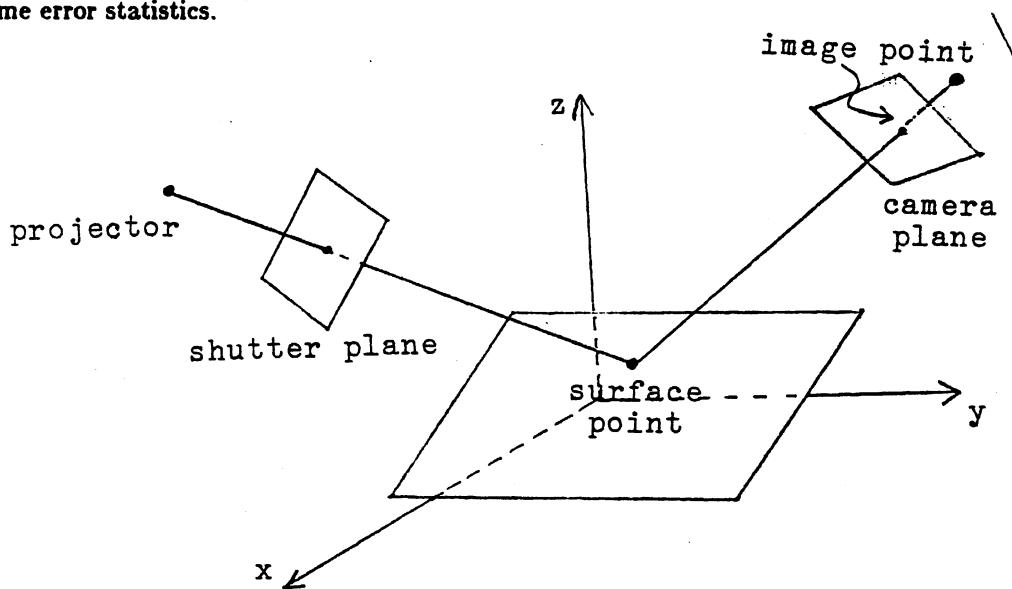


Figure 1. Stereo Camera System

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_c and y_c 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 D_sh	$r \theta, \phi$ of the focal point of the projector Distance from the focal point of the projector to the center of the shutter plane
Su, Sv N_u, N_v	Half width and half length of the shutter plane 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 D_ca	$r \theta, \phi$ of the focal point of the camera Distance from the focal point of the camera to the center of the camera plane
Sx, Sy N_x, N_y	Half width and half length of the camera plane Number of sample points on the camera in x, and y, axis At present, $N_x, N_y \leq 128$ and are powers of 2

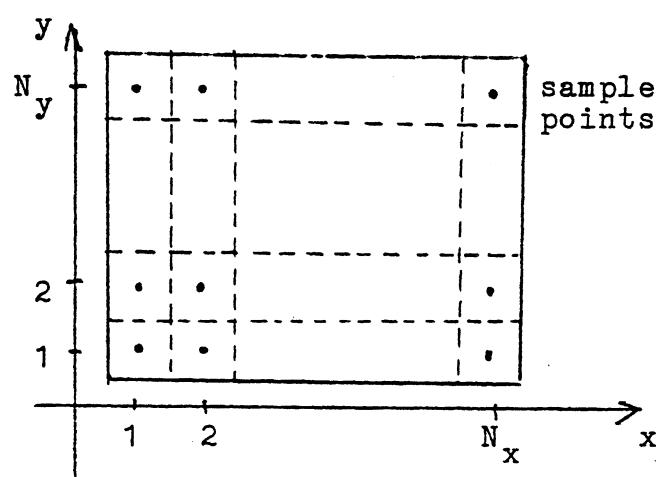
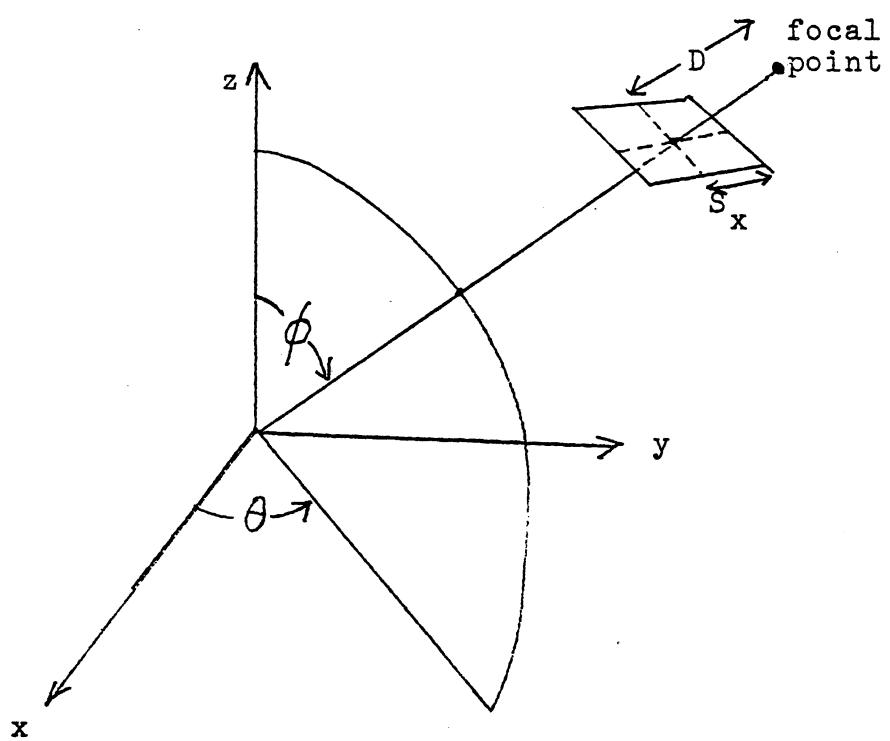
For the calibration cube

Length Step	Length of the side of the cube 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 xt, yt, zt	Rotation of the cube around the world "axis" 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 $a_\theta, -a_\phi$, 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]:

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

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 catastrophic cancellations in floating point arithmetic do not provide 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. $(u + m \text{ step}, v + n \text{ step})$, where $m, n = 1, 2, \dots$. 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].

$$\left[\begin{array}{cccc} -F_x \sin(\theta) & -F_y \cos(\theta) \cos(\phi) & -\cos(\theta) \cos(\phi) & 0 \\ F_x \cos(\theta) & -F_y \sin(\theta) \cos(\phi) & -\sin(\theta) \sin(\phi) & 0 \\ 0 & F_y \sin(\phi) & -\cos(\phi) & 0 \\ F_x(x \sin(\theta) - y \cos(\theta)) & F_y(-z \sin(\phi) + x \cos(\theta) \cos(\phi) + y \sin(\theta) \cos(\phi)) & z \cos(\phi) + x \cos(\theta) \sin(\phi) + y \sin(\theta) \sin(\phi) & 1 \end{array} \right]$$

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

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

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

where \vec{i} , \vec{j} , and \vec{k} are the unit vectors of the Cartesian world coordinate system, and \vec{a}_u , and \vec{a}_v

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

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

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

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

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

$$A = b_1 c_n - b_n c_1$$

$$B = c_1 a_n - c_n a_1$$

$$C = a_1 b_n - a_n b_1$$

$$D = -x_p A - y_p B - z_p C$$

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 .

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

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

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

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

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

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

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 acquisition 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 $V_{n,l}$ of the moved vertices are found by

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

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 $V \cdot F$ 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. Imgini

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],

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

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

where $i = 1, 2, \dots, N$, 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,

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

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

where u^1 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 $\log_2(\text{number of columns}) + 1$. 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		standard deviation	
θ	ϕ	average	
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: $r = 20$, $\theta = 30$, $\phi = 45$, $D = 4$, $S_u, S_v = 2 N_u, N_v = 16$
 camera: $r = 20$, $D = 4$, $S_x, S_y = 2 N_x, N_y = 32$
 equation of the plane: $z = -1$

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

Table 2. Effect of Number of Sampling Points

shutter: $r = 20, \theta = 30, \phi = 45, D = 4, S_u, S_v = 2$ camera: $r = 20, \theta = 60, \phi = 60, D = 4, S_x, S_y = 2$

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: $r = 20$, $\theta = 30$, $\phi = 45$, $D = 4$, S_u , $S_v = 2 N_u N_v = 16$
 camera: $r = 20$, $\theta = 60$, $\phi = 60$, $D = 4$, S_x , $S_y = 2 N_x N_y = 32$
 calibration: size = 8, rotation = x 10, y 45, z 30, translation = 0, 0, 0
 equation of the plane: $z = -1$

rotations			average	standard deviation	
x	y	z			
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: $r = 20$, $\theta = 30$, $\phi = 45$, $D = 4$, $S_u, S_v = 2$ camera: $r = 20$, $\theta = 60$, $\phi = 60$, $D = 4$, $S_x, S_y = 2$

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 # PHAN_CAL

SOURCE PROGRAM

```
V-----+-----+-----+-----+-----+-----+-----+-----+-----+
      1-----+-----+-----+-----+-----+-----+-----+-----+
      2-----+-----+-----+-----+-----+-----+-----+-----+-----+
      3-----+-----+-----+-----+-----+-----+-----+-----+-----+
      4-----+-----+-----+-----+-----+-----+-----+-----+-----+
      5-----+-----+-----+-----+-----+-----+-----+-----+-----+
      6-----+-----+-----+-----+-----+-----+-----+-----+-----+
      7-----+-----+-----+-----+-----+-----+-----+-----+-----+
      8-----+-----+-----+-----+-----+-----+-----+-----+-----+
      9-----+-----+-----+-----+-----+-----+-----+-----+-----+
      10-----+-----+-----+-----+-----+-----+-----+-----+-----+
      11-----+-----+-----+-----+-----+-----+-----+-----+-----+
      12-----+-----+-----+-----+-----+-----+-----+-----+-----+
      13-----+-----+-----+-----+-----+-----+-----+-----+-----+
      14-----+-----+-----+-----+-----+-----+-----+-----+-----+
      15-----+-----+-----+-----+-----+-----+-----+-----+-----+
      16-----+-----+-----+-----+-----+-----+-----+-----+-----+
      17-----+-----+-----+-----+-----+-----+-----+-----+-----+
      18-----+-----+-----+-----+-----+-----+-----+-----+-----+
      19-----+-----+-----+-----+-----+-----+-----+-----+-----+
      20-----+-----+-----+-----+-----+-----+-----+-----+-----+
      21-----+-----+-----+-----+-----+-----+-----+-----+-----+
      22-----+-----+-----+-----+-----+-----+-----+-----+-----+
      23-----+-----+-----+-----+-----+-----+-----+-----+-----+
      24-----+-----+-----+-----+-----+-----+-----+-----+-----+
      25-----+-----+-----+-----+-----+-----+-----+-----+-----+
      26-----+-----+-----+-----+-----+-----+-----+-----+-----+
      27-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

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

```

28.000
29.000
30.000
31.000      u_max = 64 ; { maximum number of sample points along u axis }
32.000      v_max = 64 ; { maximum number of sample points along v axis }
33.000      xs_max = 128 ; { maximum number of sample points along xs axis }
34.000      ys_max = 128 ; { maximum number of sample points along ys axis }

35.000
36.000      unobserved = -999 ; { number of decimal places of real number }
37.000      pre = 7 ; { length of the real field }
38.000      wid = 14 ; { for FORTRAN interface }
39.000      wdf = wid - 1 ; { length of the integer field }
40.000      len = 6 ; { for FORTRAN interface }
41.000      lendif = len - 1 ; { for FORTRAN interface }

42.000
43.000
44.000
45.000
46.000      Vector = ARRAY (. 1..3 .) OF REAL ;
47.000      Mat_4 = ARRAY (. 1..4, 1..4 .) OF REAL ;
48.000
49.000
50.000
51.000      Cal_f,          { Calibration file }
52.000      Img_f,          { Image file }
53.000      Ana_f : TEXT; { File for Analysis program }

54.000
55.000      Tmat,           { T matrix }
56.000      Lmat,           { L matrix }
57.000      View_xform : ARRAY (. 1..4, 1..4 .) OF REAL ; { transforms world coordinates into clip coordinates }

58.000
59.000
60.000      Ray : ARRAY (. 1..xs_max, 1..ys_max, 1..3 .) OF REAL ; { direction numbers of each ray passing through the focal point }
61.000      { of the camera and sample points on the camera plane }
62.000
63.000
64.000      Beam : ARRAY (. 1..u_max, 1..v_max, 1..3 .) OF REAL ; { direction numbers of each beam passing through the focal point }
65.000      { of the shutter and sample points on the shutter plane }

66.000
67.000
68.000      Beam_Plane : ARRAY (. 1..u_max, 1..4 .) OF REAL ; { equation of the planes passing through the focal point of the shutter }
69.000      { and two sample points on the same column of the shutter plane }
70.000
71.000      Screen : ARRAY (. 1..xs_max, 1..ys_max .) OF BOOLEAN ; { If two different shutter points imaged on the same camera point }
72.000      { Space Coding gives a spurious column }
73.000      { To avoid this, the camera point is recorded true when it is imaged }

74.000
75.000
76.000      Pattern : ARRAY (. 1..u_max, 1..v_max, 1..2 .) OF INTEGER ; { image point for each shutter point }
77.000
78.000      H_mat,          { calibration cube transformation matrix }
79.000

```

PASCAL/VIS RELEASE 2.1
LINE NUMBER B P C I STMT #

10/10/83 11:28:19

PAGE 3

SOURCE PROGRAM

```
V-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
81.000 |  
82.000 | { Every name ending with "sh" is associated with shutter }  
83.000 | { "u" and "v" are coordinate system in the shutter plane }  
84.000 |  
85.000 | { Every name ending with "ca" is associated with camera }  
86.000 | { "x" and "y" are coordinate system in the camera plane }  
87.000 |  
88.000 |  
89.000 |  
90.000 |  
91.000 |  
92.000 |  
93.000 |  
94.000 |  
95.000 |  
96.000 |  
97.000 |  
98.000 |  
99.000 |  
100.000 |  
101.000 |  
102.000 |  
103.000 |  
104.000 |  
105.000 |  
106.000 |  
107.000 |  
108.000 |  
109.000 |  
110.000 |  
111.000 |  
112.000 |  
113.000 |  
114.000 |  
115.000 |  
116.000 |  
117.000 |  
118.000 | { names are }  
119.000 | { same as }  
120.000 | { above }  
121.000 | { with camera }  
122.000 | { instead of }  
123.000 | { shutter }  
124.000 |  
125.000 |  
126.000 |  
127.000 |  
128.000 |  
129.000 |  
130.000 |  
131.000 |  
132.000 |  
133.000 | { 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 #

S O U R C E P R O G R A M

```
134.000
135.000
136.000
137.000
138.000
139.000
140.000
141.000
```

```
      PHAN_CAL :          ;           S O U R C E P R O G R A M
      V          +-----+-----+
      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 }
```

PHAN_CAL

SOURCE PROGRAM

LINE NUMBER BPC1 STATM #

```
142.000
143.000
144.000
145.000
146.000
147.000
148.000
149.000
150.000
151.000
152.000
153.000
154.000
155.000
156.000
157.000
158.000
159.000
160.000
161.000
162.000
163.000
164.000
165.000
166.000
167.000
168.000

V-----1-----2-----3-----4-----5-----6-----7-----8-----9-----10-----11V

142.000
/*
  Make_Identity
  *
  * This makes a 4 by 4 matrix an identity matrix
  */
PROCEDURE Make_Identity ( VAR Matrix : mat_4 ) ;
  VAR
    i, j : INTEGER; { indices }
BEGIN
  { Initialize 4 by 4 Matrix as Identity matrix }
  FOR i := 1 TO 4 DO
    FOR j := 1 TO 4 DO
      Matrix (i, j) := 0;
    FOR i := 1 TO 4 DO
      Matrix (i, i) := 1;
  END;
```

PHAN_CAL

SOURCE PROGRAM

LINE NUMBER	B P C I	STMT #	V
169.000			1
170.000			2
171.000			3
172.000			4
173.000			5
174.000			6
175.000			7
176.000			8
177.000			9
178.000	1		1V
179.000	1		
180.000	1		
181.000	1		
182.000	1		
183.000	1		
184.000	1		
185.000	1		
186.000	1		
187.000	1		
188.000	1	1	
189.000	1	2	
190.000	1	2	
191.000	1	2	
192.000	1	3	
193.000	1	3	
194.000	1	2	
195.000	1	2	
196.000			

```

/*
 *      Mult_Mat4
 *
 *      This multiplies two 4 by 4 matrix
 */
PROCEDURE Mult_Mat4 ( VAR Mat : mat_4 ;
                      Mat_a, Mat_b : mat_4 ) ;

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

BEGIN
  { multiply 4 by 4 Matrix Mat_a and Mat_b }
  FOR i := 1 TO 4 DO
    FOR j := 1 TO 4 DO
      BEGIN
        Mat ( .i, .j ) := 0 ;
        FOR k := 1 TO 4 DO
          Mat ( .i, .j ) := Mat ( .i, .k ) + Mat_a ( .i, .k ) * Mat_b ( .k, .j ) ;
      END ;
END ;

```

S O U R C E P R O G R A M

```
197.000
198.000
199.000
200.000
201.000
202.000
203.000
204.000
205.000
206.000
207.000
208.000
209.000
210.000
211.000
212.000
213.000
214.000
215.000
216.000
217.000
218.000
219.000
220.000
221.000
222.000
223.000
224.000
225.000
226.000
227.000
228.000
229.000
230.000
231.000
232.000
233.000
234.000
235.000
236.000
237.000
238.000
239.000
240.000
241.000
242.000
243.000
244.000
245.000
246.000
247.000
248.000
249.000

/* *
 * Read_Parameters
 * *
 * This reads parameters for phantom program
 */

PROCEDURE Read_Parameters;

BEGIN
    WRITELN( ' Enter Parameters ' );
    WRITELN( ' ---- shutter parameters ---- ' );
    WRITELN( ' ---- focal point -- r ' );
    READLN( Dist_sh );
    WRITELN( ' ---- dist_sh ' );
    READLN( Theta_sh );
    WRITELN( ' ---- focal point -- theta ' );
    READLN( Phi_sh );
    WRITELN( ' ---- shutter plane -- D ' );
    READLN( D_sh );
    WRITELN( ' ---- shutter plane -- Su ' );
    READLN( Su );
    WRITELN( ' ---- shutter plane -- Sv ' );
    READLN( Sv );
    WRITELN( ' ---- shutter plane -- Nu ' );
    READLN( Nu );
    WRITE( ca1_f, Nu :1enf );
    WRITELN( ' ---- shutter plane -- Nv ' );
    READLN( Nv );
    WRITE( ca1_f, Nv :1en );
    WRITELN( ' ---- camera parameters ---- ' );
    READLN( Dist_ca );
    WRITELN( ' ---- focal point -- r ' );
    READLN( Dist_ca );
    WRITELN( ' ---- focal point -- theta ' );
```

LINE NUMBER	BPC1	STMT #	SOURCE PROGRAM
		V	V
250.000	1	29	READLN (Theta_ca) ;
251.000	1	30	WRITELN (' focal point -- phi ') ;
252.000	1	31	READLN (Phi_ca) ;
253.000	1		
254.000	1	32	WRITELN (' camera plane -- D ') ;
255.000	1	33	READLN (D_ca) ;
256.000	1		
257.000	1	34	WRITELN (' camera plane -- Sx') ;
258.000	1	35	READLN (Sx) ;
259.000	1		
260.000	1	36	WRITELN (' camera plane -- Sy ') ;
261.000	1	37	READLN (Sy) ;
262.000	1		
263.000	1	38	WRITELN (' camera plane -- Nx ') ;
264.000	1	39	READLN (Nx) ;
265.000	1	40	WRITE (cal_f, Nx :lenf) ;
266.000	1		
267.000	1	41	WRITELN (' camera plane -- Ny') ;
268.000	1	42	READLN (Ny) ;
269.000	1	43	WRITE (cal_f, Ny :len) ;
270.000	1		
271.000	1	44	WRITELN (' Equation of the plane to be mapped -- A, B, C, D') ;
272.000	1	45	READLN (Map_plane(.1.), Map_plane(.2.), Map_plane(.3.), Map_plane(.4.)) ;
273.000	1	46	WRITELN ;
274.000	1		
275.000	1		
276.000	1		END ;

PASCAL/V RELEASE 2.1 PHAN_CAL :
 LINE NUMBER B P C I STMT #

```

 277.000      1
 278.000      1
 279.000      1
 280.000      1    /* Initialize
 281.000      1    */
 282.000      1
 283.000      1
 284.000      1
 285.000      1
 286.000      1
 287.000      1
 288.000      1
 289.000      1
 290.000      1
 291.000      1
 292.000      1
 293.000      1
 294.000      1
 295.000      1
 296.000      1
 297.000      1
 298.000      1
 299.000      1
 300.000      1
 301.000      1
 302.000      1
 303.000      1
 304.000      1
 305.000      1
 306.000      1
 307.000      1
 308.000      1
 309.000      1
 310.000      1
 311.000      1
 312.000      1
 313.000      1
 314.000      1
 315.000      1
 316.000      1
 317.000      1
 318.000      1
 319.000      1
 320.000      1
 321.000      1
 322.000      1
 323.000      1
 324.000      1
 325.000      1
 326.000      1
 327.000      1
 328.000      1
 329.000      1

```

10/10/83 11:28:19 S O U R C E P R O G R A M
 PAGE 9

```

 1      PROCEDURE Initialize ;
 2          { convert angles to radian }
 3          ang_rad : REAL ; { angle to radian conversion }
 4          i, j : INTEGER ; { indices }
 5
 6      BEGIN
 7          { convert angles to radian }
 8          ang_rad := 3.141592 / 180 ;
 9          Theta_sh := Theta_sh * ang_rad ;
10          Phi_sh := Phi_sh * ang_rad ;
11          Theta_ca := Theta_ca * ang_rad ;
12          Phi_ca := Phi_ca * ang_rad ;
13
14          { calculate direction numbers }
15          Dir_no_sh (.1.) := COS ( Theta_sh )* SIN ( phi_sh ) ;
16          Dir_no_sh (.2.) := SIN ( Theta_sh )* SIN ( phi_sh ) ;
17          Dir_no_sh (.3.) := COS ( phi_sh ) ;
18
19          Dir_no_ca (.1.) := COS ( Theta_ca )* SIN ( phi_ca ) ;
20          Dir_no_ca (.2.) := SIN ( Theta_ca )* SIN ( phi_ca ) ;
21          Dir_no_ca (.3.) := COS ( phi_ca ) ;
22
23          { calculate Cartesian coordinate of focal points and center of shutter and camera }
24
25          FOR i := 1 TO 3 DO
26             BEGIN
27              FP_sh (.1.) := Dist_sh * Dir_no_sh (.1.) ;
28              FP_ca (.1.) := Dist_ca * Dir_no_ca (.1.) ;
29
30              Center_sh (.1.) := ( Dist_sh - D_sh ) * Dir_no_sh (.1.) ;
31              Center_ca (.1.) := ( Dist_ca - D_ca ) * Dir_no_ca (.1.) ;
32             END ;
33
34          { initialize screen }
35          FOR i := 1 TO Ny DO
36             FOR j := 1 TO Nx DO
37              Screen ( .1, j .) := FALSE
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329

```

PASCAL/VIS RELEASE 2.1 PHAN_CAL : 10/10/83 11:28:19
PAGE 10

LINE NUMBER B P C I STMT # SOURCE PROGRAM

V +-----+-----+-----+-----+-----+-----+-----+-----+-----+IV.

330.000
331.000
332.000

END ;

PASCAL/VIS RELEASE 2.1 · PHAN_CAL
LINE NUMBER B P C I Stmt #

10/10/83 11:28:19

PAGE 11

SOURCE PROGRAM

```
333.000
334.000
335.000
336.000
337.000
338.000
339.000
340.000
341.000
342.000
343.000
344.000
345.000
346.000
347.000
348.000
349.000
350.000
351.000
352.000
353.000
354.000
355.000
356.000
357.000
358.000
359.000
360.000
361.000
362.000
363.000
364.000
365.000
366.000
367.000
368.000
369.000
370.000
371.000
372.000
373.000
374.000
375.000
376.000
377.000
378.000
379.000
380.000
381.000
382.000
383.000
384.000
385.000

    /*      Find_Tmat
    *      This calculates T matrix by geometrical consideration
    */

PROCEDURE Find_Tmat :  
VAR  
    Pers_xform : ARRAY (. 1.4, 1..4.) OF REAL ;  
    SFx, SFy : REAL ;  
    x, y, z : REAL ;  
    i, j, k : INTEGER ;  
    temp : REAL ;  
BEGIN  
    1 SFx := D_ca / Sx ;  
    2 SFy := D_ca / Sy ;  
    3 x := FP_ca (.1.) ;  
    4 y := FP_ca (.2.) ;  
    5 z := FP_ca (.3.) ;  
    6 View_xform (. 1, 1) := - SFx * SIN(Theta_ca) ;  
    7 View_xform (. 2, 1) := SFx * COS(Theta_ca) ;  
    8 View_xform (. 3, 1) := 0 ;  
    9 View_xform (. 4, 1) := SFx * ( x * SIN ( Theta_ca ) - y * COS ( Theta_ca ) ) ;  
    10 View_xform (. 1, 2) := - SFy * COS ( Theta_ca ) * COS ( Phi_ca ) ;  
    11 View_xform (. 2, 2) := - SFy * SIN ( Theta_ca ) * COS ( Phi_ca ) ;  
    12 View_xform (. 3, 2) := SFy * SIN ( Phi_ca ) ;  
    13 View_xform (. 4, 2) := SFy * ( - z * SIN(Phi_ca) + x * COS(Theta_ca) * COS(Phi_ca) ) ;  
    14 View_xform (. 1, 3) := - COS ( Theta_ca ) * SIN ( Phi_ca ) ;  
    15 View_xform (. 2, 3) := - SIN ( Theta_ca ) * SIN ( Phi_ca ) ;  
    16 View_xform (. 3, 3) := - COS ( Phi_ca ) ;  
    17 View_xform (. 4, 3) := z * COS(Phi_ca) + x * COS(Theta_ca) * SIN(Phi_ca) ;  
    18 View_xform (. 1, 4) := 0 ;  
    19 View_xform (. 2, 4) := 0 ;  
    20 View_xform (. 3, 4) := 0 ;  
    21 View_xform (. 4, 4) := 1 ;  
    { Perspective transform }  
    FOR i := 1 TO 4 DO
```

PHAN_CAL

S O U R C E P R O G R A M

LINE NUMBER B P C I STMT

```

V-----1-----2-----3-----4-----5-----6-----7-----8-----9-----IV
23   FOR j := 1 TO 4 DO
      Pers_xform (. , i, j ) := 0 ;
24
25   Pers_xform (. , 1, 1 ) := Nx / 2 ;
26   Pers_xform (. , 2, 1 ) := Ny / 2 ;
27   Pers_xform (. , 3, 1 ) := ( 1 + Nx ) / 2 ;
28   Pers_xform (. , 3, 2 ) := ( 1 + Ny ) / 2 ;
29   Pers_xform (. , 3, 4 ) := 1 ;
30
31 { multiply View_xform by Pers_xform }
32   writeln ; writeln ( '---- T matrix ----' ) ;
33   FOR i := 1 TO 4 DO
      BEGIN
34   FOR j := 1 TO 4 DO
      BEGIN
35       temp := 0 ;
36       FOR K := 1 TO 4 DO
          temp := temp + View_xform ( . , k , j . ) * Pers_xform ( . , k , j . ) ;
37       Tmat ( . , j ) := temp ;
38   WRITE ( Tmat ( . , j ) :wid :pre )
39   END ; {for j}
40   writeln ;
41
42   END ; {for i}
43   writeln ;
44   FOR i := 1 TO 4 DO
      BEGIN
45       FOR j := 1 TO 4 DO
          WRITE ( Tmat ( . , i , j ) / Tmat ( . , 4 , 4 . ) :wid :pre ) ;
46       writeln ;
47   END ;
48
49   END ;
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425

```

LINE NUMBER B P C I STMT. #

S O U R C E P R O G R A M

```

426.000
427.000
428.000
429.000
430.000
431.000
432.000
433.000
434.000
435.000
436.000
437.000
438.000
439.000
440.000
441.000
442.000
443.000
444.000
445.000
446.000
447.000
448.000
449.000
450.000
451.000
452.000
453.000
454.000
455.000
456.000
457.000
458.000
459.000
460.000
461.000
462.000
463.000
464.000
465.000
466.000
467.000
468.000
469.000
470.000
471.000
472.000
473.000
474.000
475.000
476.000
477.000
478.000

    /*
    *      Find_Lmat
    *
    *      This calculates L matrix from the geometrical consideration
    */
PROCEDURE Find_Lmat ;
VAR
    View_mat : ARRAY (. 1..4, 1..4) OF REAL ;
        { L matrix equivalent of View_xform of T matrix }
    Pers_mat : ARRAY (. 1..4, 1..4) OF REAL ;
        { Perspective transformation matrix of shutter }
    SFx, SFy : REAL ;
        { Scale Factors }
    x, y, z : REAL ;
        { focal point }
    i, j, k : INTEGER ;
        { Indices }
    temp : REAL ;
        { temporary }

BEGIN
    SFx := D_sh / Su ;
    SFy := D_sh / Sv ;
    x := FP_sh (.1.) ;
    y := FP_sh (.2.) ;
    z := FP_sh (.3.) ;
    View_mat (. 1, 1 ) := - SFx * SIN(Theta_sh) ;
    View_mat (. 1, 2 ) := SFx * COS(Theta_sh) ;
    View_mat (. 2, 1 ) := SFx * (- x * SIN(Theta_sh) - y * COS(Theta_sh) ) ;
    View_mat (. 2, 2 ) := SFx * (- x * SIN(Theta_sh) + y * COS(Theta_sh) ) ;
    View_mat (. 3, 1 ) := 0 ;
    View_mat (. 3, 2 ) := SFy * SIN(Theta_sh) ;
    View_mat (. 3, 3 ) := SFy * (- z * SIN(Phi_sh) + x * COS(Theta_sh) * COS(Phi_sh) +
                                + y * SIN(Theta_sh) * COS(Phi_sh) ) ;
    View_mat (. 3, 4 ) := SFy * (- z * SIN(Phi_sh) + x * COS(Theta_sh) * SIN(Phi_sh) +
                                + y * SIN(Theta_sh) * SIN(Phi_sh) ) ;
    View_mat (. 4, 1 ) := SFx * (- y * SIN(Theta_sh) - z * COS(Theta_sh) ) ;
    View_mat (. 4, 2 ) := SFx * (- y * SIN(Theta_sh) + z * COS(Theta_sh) ) ;
    View_mat (. 4, 3 ) := 0 ;
    View_mat (. 4, 4 ) := SFx * (- z * SIN(Phi_sh) + y * COS(Theta_sh) * COS(Phi_sh) +
                                + y * SIN(Theta_sh) * SIN(Phi_sh) ) ;
    View_mat (. 5, 1 ) := 0 ;
    View_mat (. 5, 2 ) := 0 ;
    View_mat (. 5, 3 ) := 0 ;
    View_mat (. 5, 4 ) := 1 ;
    View_mat (. 6, 1 ) := 0 ;
    View_mat (. 6, 2 ) := 0 ;
    View_mat (. 6, 3 ) := 0 ;
    View_mat (. 6, 4 ) := 0 ;
    View_mat (. 7, 1 ) := 0 ;
    View_mat (. 7, 2 ) := 0 ;
    View_mat (. 7, 3 ) := 0 ;
    View_mat (. 7, 4 ) := 0 ;
    View_mat (. 8, 1 ) := 0 ;
    View_mat (. 8, 2 ) := 0 ;
    View_mat (. 8, 3 ) := 0 ;
    View_mat (. 8, 4 ) := 0 ;
    View_mat (. 9, 1 ) := 0 ;
    View_mat (. 9, 2 ) := 0 ;
    View_mat (. 9, 3 ) := 0 ;
    View_mat (. 9, 4 ) := 0 ;
    View_mat (. 10, 1 ) := 0 ;
    View_mat (. 10, 2 ) := 0 ;
    View_mat (. 10, 3 ) := 0 ;
    View_mat (. 10, 4 ) := 0 ;
    View_mat (. 11, 1 ) := 0 ;
    View_mat (. 11, 2 ) := 0 ;
    View_mat (. 11, 3 ) := 0 ;
    View_mat (. 11, 4 ) := 0 ;
    View_mat (. 12, 1 ) := 0 ;
    View_mat (. 12, 2 ) := 0 ;
    View_mat (. 12, 3 ) := 0 ;
    View_mat (. 12, 4 ) := 0 ;
    View_mat (. 13, 1 ) := 0 ;
    View_mat (. 13, 2 ) := 0 ;
    View_mat (. 13, 3 ) := 0 ;
    View_mat (. 13, 4 ) := 0 ;
    View_mat (. 14, 1 ) := 0 ;
    View_mat (. 14, 2 ) := 0 ;
    View_mat (. 14, 3 ) := 0 ;
    View_mat (. 14, 4 ) := 0 ;
    View_mat (. 15, 1 ) := 0 ;
    View_mat (. 15, 2 ) := 0 ;
    View_mat (. 15, 3 ) := 0 ;
    View_mat (. 15, 4 ) := 0 ;
    View_mat (. 16, 1 ) := 0 ;
    View_mat (. 16, 2 ) := 0 ;
    View_mat (. 16, 3 ) := 0 ;
    View_mat (. 16, 4 ) := 0 ;
    View_mat (. 17, 1 ) := 0 ;
    View_mat (. 17, 2 ) := 0 ;
    View_mat (. 17, 3 ) := 0 ;
    View_mat (. 17, 4 ) := 0 ;
    View_mat (. 18, 1 ) := 0 ;
    View_mat (. 18, 2 ) := 0 ;
    View_mat (. 18, 3 ) := 0 ;
    View_mat (. 18, 4 ) := 0 ;
    View_mat (. 19, 1 ) := 0 ;
    View_mat (. 19, 2 ) := 0 ;
    View_mat (. 19, 3 ) := 0 ;
    View_mat (. 19, 4 ) := 0 ;
    View_mat (. 20, 1 ) := 0 ;
    View_mat (. 20, 2 ) := 0 ;
    View_mat (. 20, 3 ) := 0 ;
    View_mat (. 20, 4 ) := 0 ;
    View_mat (. 21, 1 ) := 0 ;
    View_mat (. 21, 2 ) := 0 ;
    View_mat (. 21, 3 ) := 0 ;
    View_mat (. 21, 4 ) := 0 ;

```

LINE NUMBER B P C I

STMT #

```

V-----1-----2-----3-----4-----5-----6-----7-----8-----9-----IV
479.000   1
480.000   1
481.000   1
482.000   2
483.000   1
484.000   1
485.000   1
486.000   1
487.000   1
488.000   1
489.000   1
490.000   1
491.000   1
492.000   1
493.000   1
494.000   1
495.000   1
496.000   1
497.000   1
498.000   1
499.000   1
500.000   1
501.000   2
502.000   2
503.000   2
504.000   2
505.000   2
506.000   1
507.000   1
508.000   1
509.000   1
510.000   1
511.000   1
512.000   1
513.000   1
514.000   1
515.000   1
516.000   1
517.000   1
518.000   1
519.000   1
520.000   1
521.000   1
522.000   1
523.000   1
      { Perspective transform }

22  FOR i := 1 TO 4 DO
23    FOR j := 1 TO 4 DO
24      Pers_mat (. . i .) := 0 ;
25      Pers_mat (. . 1 . 1 .) := Nu / 2 ;
26      Pers_mat (. . 2 . 2 .) := Nv / 2 ;
27      Pers_mat (. . 3 . 1 .) := ( 1 + Nu ) / 2 ;
28      Pers_mat (. . 3 . 2 .) := ( 1 + Nv ) / 2 ;
29      Pers_mat (. . 3 . 4 .) := 1 ;

{ multiply View_mat by Pers_mat }

30  WRITELN ;
31  WRITELN ( ' --- L matrix ----- ' );
32  FOR i := 1 TO 4 DO
33    BEGIN
34      FOR j := 1 TO 4 DO
35        BEGIN
36          temp := 0 ;
37          FOR K := 1 TO 4 DO
38            temp := temp + View_mat (. . i . k .) * Pers_mat (. . k . j .) ;
39            Lmat (. . i . j .) := temp ;
40            WRITE ( Lmat (. . i . j .) :wid :pre )
41          END ; {for j}
42          WRITELN ;
43        END ; {for i}
44        WRITELN ;
45      END ;

```

LINE NUMBER B P C I Stmt

SOURCE PROGRAM

```

524.000
525.000
526.000
527.000
528.000
529.000
530.000
531.000
532.000
533.000
534.000
535.000
536.000
537.000
538.000
539.000
540.000
541.000
542.000
543.000
544.000
545.000
546.000
547.000
548.000
549.000
550.000
551.000
552.000
553.000
554.000
555.000
556.000
557.000
558.000
559.000
560.000
561.000
562.000
563.000
564.000
565.000
566.000
567.000
568.000
569.000
570.000
571.000
572.000
573.000
574.000
575.000
576.000

/*
 * 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 verifying the T matrix calculated by
 * geometrical consideration.
 */

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

Camera : ARRAY (. .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 }

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

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

{ find the line eq of each ray }

  7  FOR xs := 1 TO Nx DO
  8    FOR ys := 1 TO Ny DO
  BEGIN
  9      FOR k := 1 TO 3 DO
  10        Camera (.xs,ys,k.) := Center_ca (.k.) + ( 2 * xs - Nx - 1 ) * ( Sx / Nx ) * axe(.k.) +
  11          ( 2 * ys - Ny - 1 ) * ( Sy / Ny ) * aye(.k.) ;
  12      Ray (.xs, ys, k.) := Camera (.xs, ys, k.) - FP_ca (.k.) ;
  END ;
  13    END
  14  END ; { find_ray_eq }
```

PASCAL/VIS RELEASE 2.1 PHAN_CAL

10/10/83 11:28:19
PAGE 16

LINE NUMBER	B P C I	STMT #	SOURCE PROGRAM
	V-	1-----	7-----
		2-----	8-----
		3-----	9-----
		4-----	1V
577.000			
578.000			
579.000			

PHAN_CAL

LINE NUMBER BPC1 SIMT #

S O U R C E P R O G R A M

```

580.000
581.000
582.000
583.000
584.000
585.000
586.000
587.000
588.000
589.000
590.000
591.000
592.000
593.000
594.000
595.000
596.000
597.000
598.000
599.000
600.000
601.000
602.000
603.000
604.000
605.000
606.000
607.000
608.000
609.000
610.000
611.000
612.000
613.000
614.000
615.000
616.000
617.000
618.000
619.000
620.000
621.000
622.000
623.000
624.000
625.000
626.000
627.000
628.000
629.000
630.000
631.000
632.000

      /* 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,           { unit vectors in the shutter plane }
  aye : Vector; { unit vectors in the shutter plane }

Shutter : ARRAY (. . . 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 }

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

  4  aye (.1.) := - COS ( Phi_sh ) * COS ( Theta_sh ) ;
  5  aye (.2.) := - COS ( Phi_sh ) * SIN ( Theta_sh ) ;
  6  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
        7  FOR k := 1 TO 3 DO
          8  Shutter (.u,v,k.) := Center_sh (.k.) + ( 2 * u - Nu - 1 ) * ( Su / Nu ) * axe (.k.) +
          9  + ( 2 * v - Nv - 1 ) * ( Sv / Nv ) * aye (.k.) ;
        10 FOR k := 1 TO 3 DO
          11 Beam (.u, v, k.) := Shutter (.u, v, k.) - FP_sh (.k.) ;
        12 END;
      END;
    END;
  END : { find_beam_eq }

```

PASCAL/VIS RELEASE 2.1 PHAN_CAL
LINE NUMBER B P C I STMT #
633.000 |

10/10/83 11:28:19

PAGE 18

SOURCE PROGRAM

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

LINE NUMBER BPC1 Stmt #

SOURCE PROGRAM

```

634.000
635.000
636.000
637.000      /* Find_Beam_Planes
638.000      */
639.000      * This calculates equations of the lines from the focal point
640.000      * of the shutter to the sample points on the shutter plane.
641.000      * This is used for verifying the L matrix calculated by
642.000      * geometrical consideration.
643.000      */

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
  FOR u := 1 TO Nu DO
    BEGIN
      a1 := Beam (.u, 1, 1) ;
      b1 := Beam (.u, 1, 2) ;
      c1 := Beam (.u, 1, 3) ;
      an := Beam (.u, Nv, 1) ;
      bn := Beam (.u, Nv, 2) ;
      cn := Beam (.u, Nv, 3) ;
      Beam_Plane (.u, 1, 1) := b1 * cn - bn * c1 ; { A }
      Beam_Plane (.u, 2, 1) := c1 * an - cn * a1 ; { B }
      Beam_Plane (.u, 3, 1) := a1 * bn - an * b1 ; { C }
      Beam_Plane (.u, 4, 1) := - FP_sh(.2.) * Beam_plane (.u, 1, 1) - FP_sh(.3.) * Beam_plane (.u, 3, 1) ; { D }
    END
  END ;

```

LINE NUMBER B P C I STMT

S O U R C E P R O G R A M

```

673.000
674.000
675.000
676.000
677.000
678.000
679.000
680.000
681.000
682.000
683.000
684.000
685.000
686.000
687.000
688.000
689.000
690.000
691.000
692.000
693.000
694.000
695.000
696.000
697.000
698.000
699.000
700.000
701.000
702.000
703.000
704.000
705.000
706.000
707.000
708.000
709.000
710.000
711.000
712.000
713.000
714.000
715.000
716.000
717.000
718.000
719.000
720.000
721.000
722.000
723.000
724.000

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

```

SOURCE PROGRAM

```

725.000
726.000
727.000
728.000
729.000
730.000
731.000
732.000
733.000
734.000
735.000
736.000
737.000
738.000
739.000
740.000
741.000
742.000
743.000
744.000
745.000
746.000
747.000
748.000
749.000
750.000
751.000
752.000
753.000
754.000
755.000
756.000
757.000
758.000
759.000
760.000
761.000
762.000
763.000
764.000
765.000
766.000
767.000
768.000
769.000
770.000
771.000
772.000
773.000

    /*
     *      Map_to_Cam_Tmat
     *      *
     *      This finds the sample points on the camera plane
     *      using the T matrix.
     */

FUNCTION Map_to_Cam_Tmat ( Pt : Vector ;
                           VAR Xs, Ys : INTEGER ) : BOOLEAN ;

VAR
  Homo : ARRAY (.1..4.) OF REAL ; { homogeneous coordinates }

  sum : REAL ; { sum }
  i, k : INTEGER ; { indices }

BEGIN
  { find homogeneous coordinates by multiplying Tmat matrix }

  FOR i := 1 TO 4 DO
    BEGIN
      sum := 0 ;
      FOR k := 1 TO 3 DO
        sum := sum + Pt(k) * Tmat (.k, i) ;
      Homo (.i) := sum + Tmat (.4, i)
    END;

  { find screen coordinates }

  Xs := ROUND ( Homo (.1.) / Homo (.4.) ) ;
  Ys := ROUND ( Homo (.2.) / Homo (.4.) ) ;

  { clip }

  IF ( 1 <= Xs ) & ( Xs <= Nx ) &
     ( 1 <= Ys ) & ( Ys <= Ny )
  THEN Map_to_Cam_Tmat := TRUE
  ELSE BEGIN
    Map_to_Cam_Tmat := FALSE ;
    Xs := unobserved ;
    Ys := unobserved ;
  END;

END;

```

SOURCE PROGRAM

```

774.000
775.000
776.000
777.000
778.000
779.000
780.000
781.000
782.000
783.000
784.000
785.000
786.000
787.000
788.000
789.000
790.000
791.000
792.000
793.000
794.000
795.000
796.000
797.000
798.000
799.000
800.000
801.000
802.000
803.000
804.000
805.000
806.000
807.000
808.000
809.000
810.000
811.000
812.000
813.000
814.000
815.000
816.000
817.000
818.000
819.000
820.000
821.000
822.000
823.000
824.000
825.000
826.000

    /*
     *      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 }
  a1, b1, c1 : REAL ;          { equation 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 }

  Ap := Dir_no_ca (.1.) ;
  Bp := Dir_no_ca (.2.) ;
  Cp := Dir_no_ca (.3.) ;
  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 }

  a1 := Pt (.1.) - FP_ca (.1.) ;
  b1 := Pt (.2.) - FP_ca (.2.) ;
  c1 := Pt (.3.) - FP_ca (.3.) ;

  { intersection of a camera plane and the line }

  t := - ( Ap * FP_ca (.1.) + Bp * FP_ca (.2.) + Cp * FP_ca (.3.) + Dp ) /
        ( Ap * a1 + Bp * b1 + Cp * c1 ) ;

  Cam_pt (.1.) := FP_ca (.1.) + t * a1 ;
  Cam_pt (.2.) := FP_ca (.2.) + t * b1 ;
  Cam_pt (.3.) := FP_ca (.3.) + t * c1 ;

  { 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 ) ;

```

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-----W
 16 | aye (.2.) := - COS ( Phi_ca ) * SIN ( Theta_ca ) ;
 17 | aye (.3.) := SIN ( Phi_ca ) ;
 18 | { find camera coordinates for the world coordinate point Cam_pt }
 19 | S1 := 0 ; S2 := 0 ;
 20 | FOR i := 1 TO 3 DO
 21 | BEGIN S1 := S1 + ( Cam_pt(.1.) - Center_ca(.1.) ) * aye (.1.) ;
 22 | S2 := S2 + ( Cam_pt(.1.) - Center_ca(.1.) ) * aye (.1.) ;
 23 | END ;
 24 | Xs := ROUND ( ( S1 * ( Nx / Sx ) + 1 + Nx ) / 2 ) ;
 25 | Ys := - ROUND ( ( - S2 * ( Ny / Sy ) + 1 + Ny ) / 2 ) + Ny + 1 ;
 26 | { Clip }
 27 | IF ( 1 <= Xs ) & ( Xs <= Nx ) &
 28 | ( 1 <= Ys ) & ( Ys <= Ny )
 29 | THEN Map_to_Cam_Geo := TRUE
 30 | ELSE BEGIN Map_to_Cam_Geo := FALSE ;
 31 | Xs := unobserved ;
 32 | Ys := unobserved
 33 | END ;
 34 | END ;
 35 | END ;
 36 | END ;
 37 | END ;
 38 | END ;
 39 | END ;
 40 | END ;
 41 | END ;
 42 | END ;
 43 | END ;
 44 | END ;
 45 | END ;
 46 | END ;
 47 | END ;
 48 | END ;
 49 | END ;
 50 | END ;
 51 | END ;
 52 | END ;
 53 | END ;
 54 | 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

855.000
856.000
857.000
858.000      /* * Make_Image_pattern
859.000      * This makes a pattern of images to be passed to the
860.000      * Surface Mapping Program.
861.000      * If two different points mapped into the same point
862.000      * on the camera, Surface Mapping Program generates a
863.000      * spurious output. This routine delete all such points
864.000      * from the pattern and flags that point.
865.000
866.000      */

867.000      PROCEDURE Make_Image_Pattern :
868.000      VAR   Intersect : Vector ; { intersection of the plane and the beam of the shutter }
869.000      A, B, C, D : REAL ; { Ax + By + Cz + D = 0 }
870.000      Xs1, Ys1,
871.000      Xs2, Ys2,
872.000      Xs3, Ys3 : INTEGER ; { Screen coordinates }
873.000      t : REAL ; { temporary }
874.000      dummy : BOOLEAN ;
875.000      u, v, k : INTEGER ; { indices }
876.000      BEGIN
877.000      1 A := Map_plane (.1.) ;
878.000      2 B := Map_plane (.2.) ;
879.000      3 C := Map_plane (.3.) ;
880.000      4 D := Map_plane (.4.) ;
881.000      5 FOR u := 1 TO NU DO
882.000      6   FOR v := 1 TO Nv DO
883.000      7     t := - ( A * FP_sh (.1.) + B * FP_sh (.2.) + C * FP_sh (.3.) + D ) /
884.000      8       ( A * Beam (.u, v, 1.) + B * Beam (.u, v, 2.) + C * Beam (.u, v, 3.) ) ;
885.000      9
886.000      10    IF Map_to_Camera ( Intersect, Xs1, Ys1 ) = TRUE
887.000      11      THEN BEGIN
888.000      12        IF Screen (.Xs1, Ys1) = FALSE
889.000      13          WRITELN ( ana_f, u : lenf, v : len, Xs1:len, Ys1:len, Intersect (.1.) :wid
890.000      14            :pre, Intersect (.2.) :wid :pre, Intersect (.3.) :wid :pre ) ;
891.000      15        Screen (.Xs1, Ys1) := TRUE ;
892.000      16
893.000      17
894.000      18
895.000      19
896.000      20
897.000      21
898.000      22
899.000      23
900.000      24
901.000      25
902.000      26
903.000      27
904.000      28
905.000      29
906.000      30
907.000      31

```

LINE NUMBER	B P C I	STMT #	SOURCE PROGRAM
		V	
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	16	{ Different beams map into same camera point }
913.000	3 1 2 2	17	Pattern (.u, v, 1.) := unobserved ;
914.000	3 1 2 2		WRITELN ('; *** Duplicate points--- ', u :len, v :len)
915.000	2 1 2 2		END
916.000	1 1 2		
917.000	1 1 2		
918.000	1 1 2	18	ELSE BEGIN Pattern (.u, v, 1.) := unobserved ;
919.000	2 1 2	19	WRITELN ('; VMAT *** not observed *** ', u, v)
920.000	2 1 1		END :
921.000	1 1 2		{ comparison of 3 methods }
922.000	1 1 2		
923.000	1 1 2		
924.000	1 1 2	20	dummy := Map_to_Cam_Geo (Intersect, xs2, ys2) ;
925.000	1 1 2	21	dummy := Map_to_Cam_Imat (Intersect, xs3, ys3) ;
926.000	1 1 2		
927.000	1 1 2		If (xs1 <> xs2) (xs2 <> xs3) (ys1 <> ys2) (ys2 <> ys3)
928.000	1 1 2	22	THEN BEGIN
929.000	1 1 2	23	WRITE ('##### Inconsistency ', u :len, v :len);
930.000	2 1 1 2	24	WRITELN (' VMAT ', xs1 :len, ys1 :len) ;
931.000	2 1 1 2	25	WRITELN (' GEO ', xs2 :len, ys2 :len) ;
932.000	2 1 1 2	26	WRITELN (' Tmat ', xs3 :len, ys3 :len)
933.000	2 1 1 2		
934.000	1 1 2		
935.000	1 1 2		
936.000	1 1 2		END {for u,v}
937.000	1 2		END :
938.000			
939.000			

PHAN_CAL

SOURCE PROGRAM

```

LINE NUMBER B P C I      STMT #      PHAN_CAL      SOURCE PROGRAM
V-----1-----2-----3-----4-----5-----6-----7-----8-----9-----1V

940.000
941.000
942.000
943.000
944.000
945.000
946.000
947.000
948.000
949.000
950.000
951.000
952.000
953.000
954.000
955.000
956.000
957.000
958.000
959.000
960.000
961.000
962.000
963.000
964.000
965.000
966.000
967.000
968.000
969.000
970.000
971.000
972.000
973.000
974.000
975.000
976.000
977.000
978.000
979.000
980.000
981.000
982.000
983.000
984.000
985.000
986.000
987.000
988.000
989.000
990.000
991.000
aaan mnn

      /*          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 Nu DO
                          IF Pattern (.Column, v, 1.) = unobserved
                            THEN WRITELN ( Img_f, Pattern (.Column, v, 1.) :1enf,
                                         Pattern (.Column, v, 2.) :1en ) ;

```

```

          END ; {for k}
          Mask_val := NOT Mask_val
        END
      END
    END
  END
END

```

PASCAL/VIS RELEASE 2.1 PHAN_CAL :OUTPUT_PATTERN 10/10/83 11:28:19

LINE NUMBER	B	P	C	I	STMT #		SOURCE PROGRAM
993.000	1	1	1		1	V	1
994.000	1	1	1		1		END : {for j}
995.000	1	1	1		15		Pat := 2 * Pat ;
996.000	1	1	1		1		16 WRITELN (Img_f, -1 :lenf , -1 :len)
997.000	1	1	1		1		END {for i}
998.000	1	1	1		1		END ;
999.000	1	1	1				
1000.000	1	1					
1001.000							
1002.000							
1003.000							
1004.000							

LINE NUMBER B P C I STMT

PHAN_CAL

SOURCE PROGRAM

```

1005.000
1006.000
1007.000
1008.000
1009.000
1010.000
1011.000
1012.000
1013.000
1014.000
1015.000
1016.000
1017.000
1018.000
1019.000
1020.000
1021.000
1022.000
1023.000
1024.000
1025.000
1026.000
1027.000
1028.000
1029.000
1030.000
1031.000
1032.000
1033.000
1034.000
1035.000
1036.000
1037.000
1038.000
1039.000
1040.000
1041.000
1042.000
1043.000
1044.000
1045.000
1046.000
1047.000
1048.000
1049.000
1050.000
1051.000
1052.000
1053.000
1054.000
1055.000
1056.000
1057.000

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

PROCEDURE Read_Cal_Para;
VAR
  R_mat,          { Rotation matrix
    Rot,           { temporary rotation matrix
    T_inv, { inverse translation matrix
    Temp_mat : mat_4 ; { temporary matrix
  }
  Axis : CHAR;   { axis of rotation of the cube }
  Angle,          { amount of rotation of the cube in degree
  Length : REAL; { length of the side of the cube
  i, j : INTEGER; { indices
}

BEGIN
  WRITELN(1, ' ----- Enter Calibration Parameters ' );
  WRITELN(2, '-----');
  WRITELN(3, '-----');

  WRITELN(4, ' length of the side of a cube -- Length ' );
  READLN(Length);

  WRITELN(5, ' Step in u and v -- Step ' );
  READLN(Step);

  a := Length / 2;

  WRITELN(6, '----- rotation of the cube ----' );
  WRITELN(7, '-----');
  WRITELN(8, '-----');

  WRITELN(9, ' Enter axis ( x, y, z, or / ) and angle ' );
  READLN(Axis, Angle);
  Angle := Angle * 3.141592 / 180;

  { Find rotation matrix R_mat }

  Make_Identity(R_mat);

  WHILE ( Axis = 'x' ) | ( Axis = 'y' ) | ( Axis = 'z' ) DO
    BEGIN
      WRITELN(10, '-----');
      WRITELN(11, '-----');
      WRITELN(12, '-----');
      WRITELN(13, '-----');
      WRITELN(14, '-----');
      WRITELN(15, '-----');
    END;
  END;
END;

```

PASCAL/VIS RELEASE 2.1 PHAN_CAL :READ_CAL PARA 10/10/83 11:28:19

PAGE 29

LINE NUMBER	B P C I	STMT #	V
1058.000	1	16	Make_Identity (Rot) ;
1059.000	1	17	CASE Axis OF
1060.000	1		'x' : BEGIN
1061.000	1		Rot(.2, .2) := COS (Angle) ;
1062.000	1		Rot(.2, .3) := - SIN (Angle) ;
1063.000	1		Rot(.3, .2) := SIN (Angle) ;
1064.000	2	18	Rot(.3, .3) := COS (Angle) ;
1065.000	2	19	END ;
1066.000	2	20	'y' : BEGIN
1067.000	2	21	Rot(.1, 1.) := COS (Angle) ;
1068.000	1		Rot(.1, 3.) := SIN (Angle) ;
1069.000	1		Rot(.3, 1.) := - SIN (Angle) ;
1070.000	1		Rot(.3, 3.) := COS (Angle) ;
1071.000	2	22	END ;
1072.000	2	23	'z' : BEGIN
1073.000	2	24	Rot(.1, 1.) := COS (Angle) ;
1074.000	2	25	Rot(.1, 2.) := - SIN (Angle) ;
1075.000	1		Rot(.2, 1.) := SIN (Angle) ;
1076.000	1		Rot(.2, 2.) := COS (Angle) ;
1077.000	1		END ; {case}
1078.000	2	26	Temp_mat := R_mat ;
1079.000	2	27	Mult_Mat4 (R_mat, Temp_mat, Rott) ;
1080.000	2	28	WRITELN (' Enter axis (x, y, z, or /) and angle ') ;
1081.000	2	29	READLN (Axis, Angle) ;
1082.000	1		Angle := Angle * 3.141592 / 180
1083.000	1		END : {while}
1084.000	1		{ Translation of the cube }
1085.000	1	30	WRITELN (' ----- Translation of the cube ----- ') ;
1086.000	1	31	WRITELN :
1087.000	1		WRITELN (' Enter translation along x, y, and z axis ') ;
1088.000	1		READLN (e.g. 3 8 10 0.5) ;
1089.000	1		WRITELN (READLN (xt, yt, zt) ;
1090.000	1		WRITELN (T_inv (.4, 1.) := - xt ;
1092.000	1		T_inv (.4, 2.) := - yt ;
1093.000	1		T_inv (.4, 3.) := - zt ;
1094.000	1		Make_Identity (T_inv) ;
1095.000	1		T_inv (.4, 1.) := - xt ;
1096.000	1		T_inv (.4, 2.) := - yt ;
1097.000	1		T_inv (.4, 3.) := - zt ;
1098.000		35	WRITELN (
1099.000		36	' ----- Translation of the cube ----- ') ;
1100.000			WRITELN :
1101.000			WRITELN (
1102.000		37	' Enter translation along x, y, and z axis ') ;
1103.000		38	WRITELN (
1104.000		39	e.g. 3 8 10 0.5) ;
1105.000			READLN (xt, yt, zt) ;
1106.000			WRITELN (T_inv (.4, 1.) := - xt ;
1107.000		40	T_inv (.4, 2.) := - yt ;
1108.000		41	T_inv (.4, 3.) := - zt ;
1109.000		42	Make_Identity (T_inv) ;
1110.000			T_inv (.4, 1.) := - xt ;
1111.000		43	T_inv (.4, 2.) := - yt ;

PASCAL/V RELEASE 2.1 PHAN_CAL :READ_CAL PARA 10/10/83 11:28:19

PAGE 30

LINE NUMBER	B P C I	STMT #	SOURCE PROGRAM
		V	
1112.000	1		{ find H_mat and inverse H_inv }
1113.000	1		H_mat := R_mat ;
1114.000	1		H_mat (.4, 1.) := xt ;
1115.000	1	44	H_mat (.4, 2.) := yt ;
1116.000	1	45	H_mat (.4, 3.) := zt ;
1117.000	1	46	
1118.000	1	47	
1119.000	1		
1120.000	1	48	FOR I := 1 TO 4 DO
1121.000	1	49	FOR J := 1 TO 4 DO
1122.000	1	50	Temp_mat (.i, .j.) := R_mat (.j, .i) ; { transpose }
1123.000	1		Mult_Mat4 (H_inv, T_inv, Temp_mat) ;
1124.000	1		
1125.000	1		
1126.000	1		END ;

LINE NUMBER B P C I Stmt #

S O U R C E P R O G R A M

```

1127.000
1128.000
1129.000
1130.000
1131.000
1132.000
1133.000
1134.000
1135.000
1136.000
1137.000
1138.000
1139.000
1140.000
1141.000
1142.000
1143.000
1144.000
1145.000
1146.000
1147.000
1148.000
1149.000
1150.000
1151.000
1152.000
1153.000
1154.000
1155.000
1156.000
1157.000
1158.000
1159.000
1160.000
1161.000
1162.000
1163.000
1164.000
1165.000
1166.000
1167.000
1168.000
1169.000
1170.000
1171.000
1172.000
1173.000
1174.000
1175.000
1176.000
1177.000
1178.000
1179.000

/* 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(.1, j.) := 0 ;
        FOR k := 1 TO 4 DO

```

LINE NUMBER B P C I STMT #

```

1180.000 1 1 3 61 V-----1-----2-----3-----4-----5-----6-----7-----8-----9-----+---1V
1181.000 1
1182.000 1
1183.000 1
1184.000 1
1185.000 1
1186.000 1
1187.000 1
1188.000 1
1189.000 1
1190.000 1
1191.000 1
1192.000 1
1193.000 1
1194.000 1
1195.000 1
1196.000 1
1197.000 1
1198.000 1
1199.000 1
1200.000 1
1201.000 1
1202.000 1
1203.000 1
1204.000 1
1205.000 1
1206.000 1
1207.000 1
1208.000 1
1209.000 1
1210.000 1
1211.000 1
1212.000 1
1213.000 1
1214.000 1
1215.000 1
1216.000 1
1217.000 1
1218.000 1
1219.000 1
1220.000 1
1221.000 1
1222.000 1
1223.000 2
1224.000 2
1225.000 2
1226.000 1
1227.000 1
1228.000 1
1229.000 1
1230.000 1
1231.000 1
1232.000 1

END ;
{ Move Faces to new position }
FOR i := 1 TO 6 DO
FOR j := 1 TO 4 DO
BEGIN
Ft(.i, j.) := 0 ;
FOR k := 1 TO 4 DO
Ft(.i, j.) := Ft(.i, j.) + H_inv(.j, k.) * F(.i, k.) ;
END ;

{ find faces which are visible by both shutter and camera }
FOR i := 1 TO 6 DO
BEGIN
n := 0 ;
{ By the sign of sign1, we know at which side of the face focal point of shutter is }
sign1 := 0 ;
FOR k := 1 TO 3 DO
sign1 := sign1 + FP_ca(.k.) * Ft(.i, k.) ;
sign1 := sign1 + Ft(.i, 4.) ;

sign2 := 0 ;
FOR k := 1 TO 3 DO
sign2 := sign2 + FP_sh(.k.) * Ft(.i, k.) ;
sign2 := sign2 + Ft(.i, 4.) ;

sign3 := 0 ;
FOR k := 1 TO 4 DO
sign3 := sign3 + Vt(.i+1, k.) * Ft(.i, k.) ;

{ focal points of shutter and camera should be at the same side of the face
{ and they should be at the opposite side of the face to a vertex selected }
IF ( ( sign1 < 0 ) & ( sign2 < 0 ) & ( sign3 > 0 ) ) | |
( ( sign1 > 0 ) & ( sign2 > 0 ) & ( sign3 < 0 ) ) THEN
BEGIN
n := n + 1 ;
FOR k := 1 TO 4 DO
Face (.n, k.) := Ft(.i, k.) ;
END ;

END ;{for}
Num_face := n ;

WRITELN (' Num face ', Num face )
WRITELN (' Num face ', Num face )

```

PASCAL/VIS RELEASE 2.1 PHAN_CAL :FIND_CAL_FACE 10/10/83 11:28:19
PAGE 33

LINE NUMBER	B P C I	STMT #	SOURCE PROGRAM
1233.000	1	V	----- 1----- 2----- 3----- 4----- 5----- 6----- 7----- 8----- 9----- V
1234.000			
1235.000			
1236.000			

END ;

PASCAL/VIS RELEASE 2.1 PHAN_CAL
LINE NUMBER B P C I Stmt #

10/10/83 11:28:19

PAGE 34

SOURCE PROGRAM

```
V---+---1---+---2---+---3---+---4---+---5---+---6---+---7---+---8---+---9---+---10---+
1237.000
1238.000
1239.000
1240.000
1241.000
1242.000
1243.000
1244.000
1245.000
1246.000
1247.000
1248.000
1249.000
1250.000
1251.000
1252.000
1253.000
1254.000
1255.000
1256.000
1257.000
1258.000
1259.000
1260.000
1261.000
1262.000
1263.000
1264.000
1265.000
1266.000
1267.000
1268.000
1269.000
1270.000
1271.000
1272.000
1273.000
1274.000
1275.000
1276.000
1277.000
1278.000
1279.000
1280.000
1281.000
1282.000
1283.000
1284.000
1285.000
1286.000
1287.000
1288.000
1289.000
*/ Output_Cube_Image
* This outputs sample points of the cube on the
camera plane.
*/
PROCEDURE Output_Cube_Image ;
VAR
    A, B, C, D, t : REAL;
    bound, dist, v : Vector;
    Intersect : Vector;
    u, v, i, k, xs, ys : INTEGER;
    Xs, Ys : 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(.1, 1);
                            B := Face(.1, 2);
                            C := Face(.1, 3);
                            D := Face(.1, 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.) );
                        END;
                END;
            END;
        END;
    END;
```

LINE NUMBER	B P C I	STMT #	:OUTPUT_CUBE_IMAG	SOURCE PROGRAM
				V-----1-----2-----3-----4-----5-----6-----7-----8-----9-----10-----11-----12-----13-----14-----15
				1290.000 3 1 4 15 1291.000 3 1 3 16 1292.000 3 1 3 1293.000 3 1 3 1294.000 3 1 3 1295.000 3 1 3 1296.000 3 1 3 18 1297.000 4 1 2 3 19 1298.000 4 1 2 3 1299.000 5 1 2 3 20 1300.000 5 1 2 3 1301.000 5 1 2 3 1302.000 5 1 2 3 21 1303.000 4 1 2 3 1304.000 3 1 1 3 1305.000 3 1 1 3 1306.000 2 1 2 1307.000 2 1 2 1308.000 2 1 2 22 1309.000 1 1 1 1310.000 1 1 1 1311.000 1 1 1 1312.000 1 1 1 1313.000 1 1 1314.000 1 1315.000 1316.000 1317.000
				FOR k := 1 TO 3 DO Intersect (.k.) := FP_sh (.k.) + t * Beam (.u, v, k.) ; dist := SQR(Intersect (.1.) - xt) + SQR(Intersect (.2.) - yt) + SQR(Intersect (.3.) - zt) ; IF (dist <= bound) THEN { Inside cube } BEGIN IF Map_to_Camera (Intersect, xs, ys) = TRUE THEN BEGIN WRITELN (cal_f, Intersect (.1.) :widf :3, Intersect (.2.) :wid :3, Intersect (.3.) :wid :3) ; WRITELN (cal_f, u :lenf, xs :len, ys :len) ; END END ;{for} v := v + step END ;{while v} u := u + step END ;{while u} END ;

LINE NUMBER B P C I Stmt

PHAN_CAL

SOURCE PROGRAM

```

1318.000
1319.000
1320.000
1321.000
1322.000
1323.000
1324.000
1325.000
1326.000
1327.000
1328.000
1329.000
1330.000
1331.000
1332.000
1333.000
1334.000
1335.000
1336.000
1337.000
1338.000
1339.000
1340.000
1341.000
1342.000
1343.000
1344.000
1345.000
1346.000
1347.000
1348.000
1349.000
1350.000
1351.000
1352.000
1353.000
1354.000
1355.000
1356.000
1357.000

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

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

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

*
*/
/
BEGIN
  { Open files }

  RESET ( INPUT, 'UNIT=SCARDS, INTERACTIVE' ) ;
  REWRITE ( Ca_l_f, 'UNIT=3' ) ;
  REWRITE ( Img_f, 'UNIT=4' ) ;
  REWRITE ( Ara_f, 'UNIT=2' ) ;

  Read_Parameters ;
  Read_Cal_Para ;
  Initialize ;
  Find_Tmat ;
  Find_Lmat ;
  Find_Ray_Eqs ;
  Find_Beam_Eqs ;
  Find_Beam_Planes ;
  Make_Image_Pattern ;
  Output_Pattern ;
  Find_Cal_Face ;
  Output_Cube_Image

END.

```

NO COMPILER DETECTED ERRORS

OPTIONS IN EFFECT: MARGIN\$ (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

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

MAIN 10-10-83 11:37:46 PAGE P001

```
C      Mapping Program : Eye.cal
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          DIRECTORY          # OF ROWS OF CONTOUR GRID
C      C          SROW               # OF COLUMNS OF CONTOUR GRID
C      C          SCOL               # OF ROWS IN ENTIRE RASTER SCREEN
C      C          CROW               # OF COLUMNS IN ENTIRE RASTER SCREEN
C      C          CCOL               # OF IMAGES NEEDED TO CODE SCENE
C      C          NIMAGE
C
C      C          LUCAL              LOGICAL UNIT FOR CALIBRATION INFO
C      C          LUDAT              LOGICAL UNIT FOR INPUT DATA
C      C          LUOUT              LOGICAL UNIT FOR FILE OUTPUT
C      C          LUTERM             LOGICAL UNIT FOR TERMINAL OUTPUT
C
C      PAT1    MATRIX OF RASTER IMAGE # 1
C      :
C      PAT7    MATRIX OF RASTER IMAGE # 7
C
C      C          LOGICAL DEVICES
C
C      C          LOGICAL UNIT #3    CALIBRATION INFO
C      C          LOGICAL UNIT #4    INPUT FILE
C      C          LOGICAL UNIT #5    OUTPUT FILE
C      C          LOGICAL UNIT #6    TERMINAL
C
C      C          Local variables
C      C          Transformation matrices
C
C      0001   C      REAL T(4,4), L(4,4)
C      C      Calibration cube data
C
```

MICHIGAN TERMINAL SYSTEM FURIKAN G(21.8)

```

0003      C      INTEGER CUBIMG(100,2), CUBCOL(100)          MAIN    10-10-83   11:37:46
          C      Begin
          C      Acquire and check the initial parameters.
          C      CALL INIT( CUBSUR, CUBIMG, CUBCOL )
          C      Calibrate the system.
          C      CALL TMAT( CUBSUR, CUBIMG, T )
          C      CALL LMAT( CUBSUR, CUBCOL, L )
          C      Solve for the unknown surface.
          C      CALL IMGIN
          C      CALL MAP( T, L )
          C      STOP
          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

```

PAGE P002

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

MAIN 10-10-83 11:37:46 PAGE P001

```

C
C--.
C      SUBROUTINE INIT( CUBSR, CUBIM, CUBCO )
C--.

C      include GLOBAL and 10 common
C      "GLOBAL", common (100, 120)

0002      C      INTEGER SROW, SCOL
C      Dimensions of the shutter grid

C      INTEGER CROW, CCOL
C      Dimensions of the camera grid

C      INTEGER NIMAGE, NUMCAL
C      Number of patterns needed
C      Number of calibration data

0004      C      COMMON/GLOBAL/
C              1      SROW, SCOL, CROW, CCOL, NIMAGE, NUMCAL
C      "10"  common (200, 210)
C      INTEGER LUCAL, LUDAT, LUOUT, LUTERM
C      Logical units for the calibration file, data, output,
C      and the terminal.

0005      C      COMMON/IO/
C              1      LUCAL, LUDAT, LUOUT, LUTERM
C
C      parameters -- calibration cube data
C      REAL CUBSR(100, 3)
C      INTEGER CUBIM(100, 2), CUBCO(100)

0008      C      Begin Initialization
C
C          C      LUCAL = 3
C          C      LUDAT = 4
C          C      LUOUT = 5
C          C      LUTERM = 6
C
C          C      CALL CALIN( CUBSR, CUBIM, CUBCO )
C
C          C      NIMAGE = INT(1.001 + ALOG(FLOAT(SCOL))/ALOG(2. ))
C
C          C      RETURN
C
0010      C
0011      C
0012      C
0013      C
C
0014      C      END
C      *OPTIONS IN EFFECT* 1D,EBCDIC, SOURCE,NOLIST, NODECK, LOAD, NOMAP
C      *OPTIONS IN EFFECT* 1NFCTN = INIT 57
C      *OPTIONS IN EFFECT* NAMF = INIT 100,000
C      *OPTIONS IN EFFECT* 101,000

```

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

MAIN! 10-10-83 11:37:47 PAGE P001

```
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-----0001
C SUBROUTINE CALIN (CUBS, CUBI, CUBC)
C-----0002
C INPUT ROUTINE FOR the CALIBRATION file
C INCLUDE Global and 10 commons
C "GLOBAL" common (100, 120)
C
C INTEGER SROW, SCOL
C Dimensions of the shutter grid
C
C INTEGER CROW, CCOL
C Dimensions of the camera grid
C
C INTEGER NIMAGE, NUMCAL
C Number of patterns needed
C Number of calibration data
C
C COMMON/GLOBAL/
C SROW, SCOL, CROW, CCOL, NIMAGE, NUMCAL
C
C "10" common (200, 210)
C INTEGER LUCAL, LUDAT, LUOUT, LUTERM
C Logical units for the calibration file, data, output,
C and the terminal.
C
C COMMON/IO/
C LUCAL, LUDAT, LUOUT, LUTERM
C
C
C Declare parameters
C REAL CUBS(100,3)
C INTEGER CUBI(100,2), CUBC(100)
C
```

STATISTICS NO DIAGNOSTICS GENERATED

```

0010      READ(LUCAL, 2000) SROW, SCOL, CROW, CCOL
0011      2000 FORMAT ( 216 / 216 )
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
          READ(LUCAL, 300, END = 2100) ( CUBS(I,J), J= 1; 3 )
0017      READ(LUCAL, 400) ( CUBC(I), CUBI(I,1), CUBI(I,2) )
0018      NUMCAL = NUMCAL + 1
0019      100 CONTINUE
0020      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( 316 )
C
0028      3000 FORMAT(99(3F10.2,/),3F10.2)
0029      3005 FORMAT(99(210./),2110)
0030      3010 FORMAT(100(110./))
0031      3015 FORMAT(1X, 'Number of calibration data is not adequate',
1           , NumCal = , 110 )
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

```

PAGE 001

11:37:47

	MAIN	10-10-83
0001	C C SUBROUTINE IMGIN	163.000 164.000 165.000 166.000 167.000 168.000 169.000 170.000 171.000 172.000 100.000 101.000 102.000 103.000 104.000 105.000 106.000 107.000 108.000 109.000 109.500 110.000 111.000 112.000 113.000 200.000 201.000 202.000 203.000 204.000 205.000 206.000 207.000 208.000 300.000 301.000 302.000 303.000 304.000 305.000 306.000 307.000 308.000 174.000 175.000 176.000 177.000 178.000 179.000 180.000 181.000 182.000 183.000 184.000
0002	C C COMMONS included here C "GLOBAL" common (100, 120)	- -
0003	C C INPUT ROUTINE FOR RASTER DATA	- -
0004	C C COMMONS included here C "GLOBAL" common (100, 120)	- -
0005	C C INPUT SROW, SCOL Dimensions of the shutter grid	- -
0006	C C INTEGER CROW, CCOL Dimensions of the camera grid	- -
0007	C C COMMON/GLOBAL/ SROW, SCOL, CROW, CCOL, NIMAGE, NUMCAL	- -
0008	C C "10" common (200, 210) INTEGER LUCAL, LUDAT, LUOUT, LUTERM Logical units for the calibration file, data, output, and the terminal.	- -
0009	C C COMMON/10/ LUCAL, LUDAT, LUOUT, LUTERM	- -
0010	C C "Patterns" common (300, 310) INTEGER PAT1(128, 128), PAT2(128, 128), PAT3(128, 128), PAT4(128, 128), PAT5(128, 128), PAT6(128, 128), PAT7(128, 128), PAT8(128, 128)	- -
0011	C CALL CLRtbl(PAT1, 128, 128)	- -
0012	C CALL CLRtbl(PAT2, 128, 128)	- -
0013	C CALL CLRtbl(PAT3, 128, 128)	- -
0014	C CALL CLRtbl(PAT4, 128, 128)	- -
0015	C CALL CLRtbl(PAT5, 128, 128)	- -
0016	C CALL CLRtbl(PAT6, 128, 128)	- -
0017	C CALL CLRtbl(PAT7, 128, 128)	- -
	C CALL CLRtbl(PAT8, 128, 128)	- -

```

C   TO SIGNAL END OF PATTERN
C
0018      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      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      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      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      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      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      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      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
0050      C100     WRITE( LUTERM, 3000 ) ( (PAT1(I,J), J=1,CCOL), I=1,CROW )
0051      C100     RETURN
0052      *OPTIONS IN EFFECT* ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
0053      *OPTIONS IN EFFECT* NAME = IMGIN   LINECNT = 57
0054      *STATISTICS* SOURCE STATEMENTS = 52,PROGRAM SIZE =
0055      *STATISTICS* NO DIAGNOSTICS GENERATED
0056      END
0057
0058
0059
0060
0061
0062
0063
0064
0065
0066
0067
0068
0069
0070
0071
0072
0073
0074
0075
0076
0077
0078
0079
0080
0081
0082
0083
0084
0085
0086
0087
0088
0089
0090
0091
0092
0093
0094
0095
0096
0097
0098
0099
0100
0101
0102
0103
0104
0105
0106
0107
0108
0109
0110
0111
0112
0113
0114
0115
0116
0117
0118
0119
0120
0121
0122
0123
0124
0125
0126
0127
0128
0129
0130
0131
0132
0133
0134
0135
0136
0137
0138
0139
0140
0141
0142
0143
0144
0145
0146
0147
0148
0149
0150
0151
0152
0153
0154
0155
0156
0157
0158
0159
0160
0161
0162
0163
0164
0165
0166
0167
0168
0169
0170
0171
0172
0173
0174
0175
0176
0177
0178
0179
0180
0181
0182
0183
0184
0185
0186
0187
0188
0189
0190
0191
0192
0193
0194
0195
0196
0197
0198
0199
0200
0201
0202
0203
0204
0205
0206
0207
0208
0209
0210
0211
0212
0213
0214
0215
0216
0217
0218
0219
0220
0221
0222
0223
0224
0225
0226
0227
0228
0229
0230
0231
0232
0233
0234
0235
0236
0237
0238
0239
0240
0241
0242
0243
0244
0245
0246
0247
0248
0249
0250
0251
0252
0253
0254
0255
0256
0257
0258
0259
0260
0261
0262
0263
0264
0265
0266
0267
0268
0269
0270
0271
0272
0273
0274
0275
0276
0277
0278
0279
0280
0281
0282
0283
0284
0285
0286
0287
0288
0289
0290
0291
0292
0293
0294
0295
0296
0297
0298
0299
0300
0301
0302
0303
0304
0305
0306
0307
0308
0309
0310
0311
0312
0313
0314
0315
0316
0317
0318
0319
0320
0321
0322
0323
0324
0325
0326
0327
0328
0329
0330
0331
0332
0333
0334
0335
0336
0337
0338
0339
0340
0341
0342
0343
0344
0345
0346
0347
0348
0349
0350
0351
0352
0353
0354
0355
0356
0357
0358
0359
0360
0361
0362
0363
0364
0365
0366
0367
0368
0369
0370
0371
0372
0373
0374
0375
0376
0377
0378
0379
0380
0381
0382
0383
0384
0385
0386
0387
0388
0389
0390
0391
0392
0393
0394
0395
0396
0397
0398
0399
0400
0401
0402
0403
0404
0405
0406
0407
0408
0409
0410
0411
0412
0413
0414
0415
0416
0417
0418
0419
0420
0421
0422
0423
0424
0425
0426
0427
0428
0429
0430
0431
0432
0433
0434
0435
0436
0437
0438
0439
0440
0441
0442
0443
0444
0445
0446
0447
0448
0449
0450
0451
0452
0453
0454
0455
0456
0457
0458
0459
0460
0461
0462
0463
0464
0465
0466
0467
0468
0469
0470
0471
0472
0473
0474
0475
0476
0477
0478
0479
0480
0481
0482
0483
0484
0485
0486
0487
0488
0489
0490
0491
0492
0493
0494
0495
0496
0497
0498
0499
0500
0501
0502
0503
0504
0505
0506
0507
0508
0509
0510
0511
0512
0513
0514
0515
0516
0517
0518
0519
0520
0521
0522
0523
0524
0525
0526
0527
0528
0529
0530
0531
0532
0533
0534
0535
0536
0537
0538
0539
0540
0541
0542
0543
0544
0545
0546
0547
0548
0549
0550
0551
0552
0553
0554
0555
0556
0557
0558
0559
0560
0561
0562
0563
0564
0565
0566
0567
0568
0569
0570
0571
0572
0573
0574
0575
0576
0577
0578
0579
0580
0581
0582
0583
0584
0585
0586
0587
0588
0589
0589
0590
0591
0592
0593
0594
0595
0596
0597
0598
0599
0600
0601
0602
0603
0604
0605
0606
0607
0608
0609
0610
0611
0612
0613
0614
0615
0616
0617
0618
0619
0620
0621
0622
0623
0624
0625
0626
0627
0628
0629
0630
0631
0632
0633
0634
0635
0636
0637
0638
0639
0640
0641
0642
0643
0644
0645
0646
0647
0648
0649
0650
0651
0652
0653
0654
0655
0656
0657
0658
0659
0660
0661
0662
0663
0664
0665
0666
0667
0668
0669
0669
0670
0671
0672
0673
0674
0675
0676
0677
0678
0679
0680
0681
0682
0683
0684
0685
0686
0687
0688
0689
0689
0690
0691
0692
0693
0694
0695
0696
0697
0698
0699
0700
0701
0702

```

OPTIONS IN EFFECT ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
 OPTIONS IN EFFECT NAME = IMGIN LINECNT = 57
 STATISTICS SOURCE STATEMENTS = 52,PROGRAM SIZE = 1702

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

MAIN 10-10-83 11:37:48 PAGE P001

C 235.000
C 236.000
C 237.000
C 239.000
C 241.000
C 242.000
C 243.000
C 244.000
C 245.000
C 246.000
C 200.000
C 201.000
C 202.000
C 203.000
C 204.000
C 205.000
C 206.000
C 207.000
C 208.000
C 248.000
C 249.000
C 250.000
C 251.000
C 252.000
C 253.000
C 254.000
C 255.000
C 256.000
C 257.000

0001 C SUBROUTINE SUROUT(SURPNT)
C OUTPUT ROUTINE FOR ESTIMATED LOCATION
C OF POINTS

0002 C IO Commons included here
C "10" Common (200, 210)
C INTEGER LUCAL, LUDAT, LUOUT, LUTERM
C Logical units for the calibration file, data, output,
C and the terminal.

0003 C COMMON/IO/
C LUCAL, LUDAT, LUOUT, LUTERM

0004 C PARAMETERS
C REAL SURPNT(3)

0005 C WRITE (LUOUT, 2000) SURPNT(1), SURPNT(2), SURPNT(3)

0006 C 2000 FORMAT(3F10.3)

0007 C 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          C
C          C
C          C          SUBROUTINE TMAT ( CUB, DIA, TM )
C          C          calibration of T matrix by least square method
C          C          Here T matrix is normalized by T44
C          C          NOTE T matrix is used only in (3-49) of Rogers in Eye.
C          C          But in Phantom, T matrix should not be normalized.
C          C
C          C          Include global and IO common here
C          C          "GLOBAL" common (100, 120)
C          C
C          C          INTEGER SROW, SCOL
C          C          Dimensions of the shutter grid
C          C          INTEGER CROW, CCOL
C          C          Dimensions of the camera grid
C          C          INTEGER NIMAGE, NUMCAL
C          C          Number of patterns needed
C          C          Number of calibration data
C
C          C          COMMON/GLOBAL/
C          C          SROW, SCOL, CROW, CCOL, NIMAGE, NUMCAL
C          C          1
C          C          "10" common (200, 210 )
C          C          INTEGER LUCAL, LUDAT, LUOUT, LUTERM
C          C          Logical units for the calibration file, data, output,
C          C          and the terminal.
C
C          C          COMMON/IO/
C          C          1 LUCAL, LUDAT, LUOUT, LUTERM
C
C          C          Solve BX = C
C          C
C          C          REAL CUB(100,3),TM(4,4)
C          C          INTEGER DIA(100,2), IWK(11),IER
C          C          REAL A(200,12),B(200,11),C(200),X(11),COMMU(4),WK(3000)
C
C          C          More least square parms
C          C          REAL EPS/4.76E-07/,QL,QLMAX/1.E07/
C          C          INTEGER ITS/10/, PRT1, PRT2
C
C          C          begin
C          C          CALL CLRITBL(A,200,12)
C          C          CALL CLRITBL(TM,4,4)
C
C          C          0013
C          C          0014

```

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

PAGE P002

10-10-83

TMAT

11:37:49

```

C      NDATA = 2 * NUMCAL          286.000
0015    J=0                         287.000
0016    DO 1005 I=1, NDATA, 2      288.000
0017    J=J+1                         289.000
0018    A(I,1) = CUB(J,1)           290.000
0019    A(I+1,2) = CUB(J,1)         291.000
0020    A(I,4) = CUB(J,2)           292.000
0021    A(I+1,5) = CUB(J,2)         293.000
0022    A(I,7) = CUB(J,3)           294.000
0023    A(I+1,8) = CUB(J,3)         295.000
0024    A(I,10) = 1.                296.000
0025    A(I+1,11) = 1.              297.000
0026    C      A(I,12) = - FLOAT( DTA(J,1) )   298.000
0027    A(I+1,12) = - FLOAT( DTA(J,2) )   299.000
0028    A(I,3) = - CUB(J,1) * FLOAT( DTA(J,1) ) 300.000
0029    A(I+1,3) = - CUB(J,1) * FLOAT( DTA(J,2) ) 301.000
0030    A(I,6) = - CUB(J,2) * FLOAT( DTA(J,1) ) 302.000
0031    A(I+1,6) = - CUB(J,2) * FLOAT( DTA(J,2) ) 303.000
0032    A(I,9) = - CUB(J,3) * FLOAT( DTA(J,1) ) 304.000
0033    A(I+1,9) = - CUB(J,3) * FLOAT( DTA(J,2) ) 305.000
0034    1005 CONTINUE                 306.000
0035    C      DO 1006 I=1,200          307.000
0036    DO 1007 J=1,11               308.000
0037    B(I,J) = A(I,J)             309.000
0038    1007  CONTINUE
0039    C(I) = - A(I,12)
0040    1006 CONTINUE
0041    C      This is a call to IMSL package
C%/%/%/% CALL LLBQF ( B,200,NDATA,11,C,200,1,O,COMMU,X,11,IWK,WK,IER )
C%/%/%/% IF ( IER .EQ. 0 ) GO TO 1010
C      This is a call to L2
C      PRT1 = 1
C      PRT2 = 0
C      CALL L2(NDATA,11,B,C,X,EPS,ITS,PRT1,PRT2,200,QL)
C      IF ( QL .LT. QLMAX ) GO TO 1010
C      WRITE(LUTERM,2001) QL
C      STOP
C      1010 CONTINUE
C      TM(1,1) = X(1)
C      TM(1,2) = X(2)
C      TM(1,4) = X(3)
C      TM(2,1) = X(4)
C      TM(2,2) = X(5)
0042
0043
0044
0045
0046
0047
0048
0049
0050
0051
0052
0053

```

```

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.
0061          C      WRITE(LUTERM,2000) ((TM(I,J),J=1,4),I=1,4)
0062          C      RETURN
0063          C      2000 FORMAT( /,1X,'T Matrix determined',/.4(4F14.7,/) )
0064          C      2001 FORMAT( 1X,'Abnormal termination --MESSAGE FROM SUBROUTINE TMAT',/,
0065          C      1           1X,'CONDITION NUMBER OF T MATRIX = ', E20.5 )
0066          C      END
0067          *OPTIONS IN EFFECT* ID.EBCDIC.SOURCE.NOLIST.NODECK.LOAD.NOMAP
0068          *OPTIONS IN EFFECT* NAME = TMAT      , LINECNT = 57
0069          *STATISTICS* SOURCE STATEMENTS = 65,PROGRAM SIZE = 333300
0070          *STATISTICS* NO DIAGNOSTICS GENERATED

```

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

MAIN 10-10-83 11:37:50 PAGE POO1

11 : 37 : 50

```

0015      DO 1005 I=1, NUMCAL
0016      A(I,1) = SURPTS(I,1)
0017      A(I,2) = - SURPTS(I,1) * FLOAT( COLPTS(I) )
0018      A(I,3) = SURPTS(I,2)
0019      A(I,4) = - SURPTS(I,2) * FLOAT( COLPTS(I) )
0020      A(I,5) = SURPTS(I,3)
0021      A(I,6) = - SURPTS(I,3) * FLOAT( COLPTS(I) )
0022      A(I,7) = 1.
0023      A(I,8) = - FLOAT( COLPTS(I) )
0024      1005 CONTINUE
C
0025      DO 1006 I=1,100
0026      DO 1007 J=1,7
0027      B(I,J) = A(I,J)
0028      CONTINUE
0029      C(I) = - A(I,8)
0030      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
0031      PRT1 = 1
0032      PRT2 = 0
0033      CALL L2(NUMCAL,7,B,C,X,EPS,ITS,PRT1,PRT2,100,QL)
C
0034      IF ( QL .LT. QLMAX ) GO TO 1010
C
0035      WRITE(LUTERM,2001) QL
0036      STOP
C
C
1010 CONTINUE
0037      LM(1,1) = X(1)
0038      LM(1,2) = X(1)
0039      LM(1,4) = X(2)
0040      LM(2,1) = X(3)
0041      LM(2,4) = X(4)
0042      LM(3,1) = X(5)
0043      LM(3,4) = X(6)
0044      LM(4,1) = X(7)
0045      LM(4,4) = 1.
0046      WRITE(LUTERM,2000) ((LM(I,J),J=1,4),I=1,4)
0047      RETURN
C
0048      2000 FORMAT('1X, 'L Matrix determined',/.4(4F14.7,/) )
C
0049      2001 FORMAT('1X, 'Abnormal termination --MESSAGE FROM SUBROUTINE LMAT',
1, ' /, 1X,'CONDITION NUMBER OF L MATRIX = ', E20.5 )
C
0050      END

```

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

```
*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
```

LMAT

10-10-83

PAGE E003

11:37:50

```

C 424.000
C 425.000
C 426.000
C 427.000
C 428.000
C 429.000
C ****
C MAP Module
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 squares equations
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 0002 Include all commons here
C "GLOBAL" common (100, 120)
C
C INTEGER SROW, SCOL
C Dimensions of the shutter grid
C
C 0003 INTEGER CROW, CCOL
C Dimensions of the camera grid
C
C 0004 INTEGER NIMAGE, NUMCAL
C Number of patterns needed
C Number of calibration data
C
C 0005 COMMON/GLOBAL/
C 1 SROW, SCOL, CROW, CCOL, NIMAGE, NUMCAL
C
C "IO" common (200, 210 )
C INTEGER LUCAL, LUDAT, LUOUT, LUTERM
C Logical units for the calibration file, data, output,
C and the terminal .
C
C

```

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

MAP 10-10-83 11:37:51 PAGE P002

```

0007      COMMON/10/
          1       LUDAT, LUOUT, LUTERM
          C
          C "Patterns" common (300,310)
          C   INTEGER PAT1(128,128), PAT2(128,128), PAT3(128,128),
          C     1           PAT4(128,128), PAT5(128,128), PAT6(128,128),
          C     2           PAT7(128,128), PAT8(128,128),
          C   Contain the image data.
          C
0008         COMMON/PATS/
          C   PAT1, PAT2, PAT3, PAT4, PAT5, PAT6,
          C     1           PAT7, PAT8
          C
0009         REAL TM(4,4), LM(4,4)
          C
0010        C Local variables:
          C
0011        C Matrices for least squares Ax=B
          C   REAL LSA(3,3), LSB(3), LSX(3)
          C
0012        C Known values in least squares equations
          C   REAL COL, XSTAR, YSTAR
          C   INTEGER U
          C
0013        C More least square parms
          C   REAL EPS/4.768E-07, QL, QLMAX/1.E07, WKAREA(200)
          C   INTEGER ITS/10/, PRT1, PRT2
          C
0014        C Begin
          C
0015        DO 20 J = 1, CCOL
0016        DO 10 I = 1, CROW
0017        IF (PAT1(I,J) .EQ. 0) GO TO 10
          C
0018        CALL CLRVC (LSB,3)
          C   CALL CLRVC (LSX,3)
          C   CALL CLRTBL (LSA,3,3)
          C
0019        CALL COLUMN(1,J,U)
          C
0020        XSTAR = FLOAT(I)
          C   YSTAR = FLOAT(J)
          C   COL = FLOAT(U)
          C
0021        CALL SETCOF(LSA,LSB,XSTAR,YSTAR,COL,TM,LM)
          C
0022        PRT1 = 0
          C   PRT2 = 0
          C   CALL L2(3,3,LSA,LSB,LSX,EPS,ITS,PRT1,PRT2,3,QL)
          C
0023        C%/%/%/%%
          C
0024
0025
0026
0027
0028
0029

```

```

0030          C      IF (QL .GT. QLMAX) GO TO 10
          C      CALL SUROUT( LSB )
          C      WRITE (LUOUT, 100) U, I, J, LSX(1), LSX(2), LSX(3)
          /
0031          C      10    CONTINUE
0032          C      20    CONTINUE
          C      100   FORMAT (1X, 3I6, 3F14.7 )
          C      RETURN
0033          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

```

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

MAIN

10-10-83

11:37:52

PAGE 0001

```

C      C      515.000
C      C      516.000
C      C      517.000
C      C      518.000
C      C      519.000
C      C      520.000
C      C      521.000
C      C      522.000
C      C      523.000
C      C      524.000
C      C      525.000
C      C      526.000
C      C      527.000
C      C      528.000
C      C      529.000
C      C      530.000
C      C      531.000
C      C      532.000
C      C      533.000
C      C      534.000
C      C      535.000
C      C      536.000
C      C      537.000
C      C      538.000
C      C      539.000
C      C      540.000
C      C      541.000
C      C      542.000
C      C      543.000
C      C      544.000
C      C      545.000
C      C      546.000

0001    C      SUBROUTINE COLUMN (RW,CL,U)
C      C      Column finds the shutter column (u) for the given point by looking
C      C      at the images the point appeared in. It is assumed that pattern two
C      C      illuminates the right half of the shutter, etc..
C      C      Input : RW - camera row for point.
C      C      CL - camera column for point.
C      C      Output: U - shutter column number for point
C      C

C      C      Include GLOBAL and PATTERN commons here
C      C      "GLOBAL" common (100,120)
C      C      INTEGER SROW, SCOL
C      C      Dimensions of the shutter grid
C      C      INTEGER CROW, CCOL
C      C      Dimensions of the camera grid
C      C      INTEGER NIMAGE, NUMCAL
C      C      Number of patterns needed
C      C      Number of calibration data
C      C      COMMON/GLOBAL/
C      C      1      SROW, SCOL, CROW, CCOL, NIMAGE, NUMCAL
C      C      "Patterns" common (300,310)
C      C      INTEGER PAT1(128,128), PAT2(128,128), PAT3(128,128),
C      C      1      PAT4(128,128), PAT5(128,128), PAT6(128,128),
C      C      2      PAT7(128,128), PAT8(128,128)
C      C      Contain the image data.
C      C      COMMON/PATS/
C      C      1      PAT1, PAT2, PAT3, PAT4, PAT5, PAT6,
C      C      2      PAT7, PAT8
C      C      INTEGER RW,CL,U
C      C      begin
C      C      U      = PAT2(RW,CL) * ( 2 ** (NIMAGE-2) )
C      C      1      + PAT3(RW,CL) * ( 2 ** (NIMAGE-3) )
C      C      2      + PAT4(RW,CL) * ( 2 ** (NIMAGE-4) )
C      C      3      + PAT5(RW,CL) * ( 2 ** (NIMAGE-5) )
C      C      4      + PAT6(RW,CL) * ( 2 ** (NIMAGE-6) )
C      C      5      + PAT7(RW,CL) * ( 2 ** (NIMAGE-7) )
C      C      6      + PAT8(RW,CL) * ( 2 ** (NIMAGE-8) )
C      C      7      +
C      C      RETURN

```

548.000

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

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

MAIN 10-10-83 11:37:52 PAGE P001

```

C      549.000
C      550.000
C      551.000
C      552.000
C      553.000
C      554.000
C      555.000
C      556.000
C      557.000
C      558.000
C      559.000
C      560.000
C      561.000
C      562.000
C      563.000
C      564.000
C      565.000
C      566.000
C      567.000
C      568.000
C      569.000
C      570.000
C      571.000
C      572.000
C      573.000
C      574.000
C      575.000
C      576.000
C      577.000
C      578.000
C      579.000
C      580.000
C      581.000
C      582.000
C      583.000
C      584.000
C      585.000
C      586.000
C      587.000
C      588.000
C      589.000
C      590.000
C      591.000
C      592.000
C      593.000

0001    C
C-----+
C      SUBROUTINE SETCOF(A, B, XS, VS, 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-T14*x)*x + (T21-T24*x*)y + (T31-T34*x*)z + (T41-T44*x*) = 0
C   (T12-T14*y)*x + (T22-T24*y*)y + (T32-T34*y*)z + (T42-T44*y*) = 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        VS - 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      REAL XS, VS, U, T(4,4), L(4,4), A(3,3), B(3)
C
C Begin:
C
C      A(1,1) = T(1,1) - T(1,4) * XS
C      A(1,2) = T(2,1) - T(2,4) * XS
C      A(1,3) = T(3,1) - T(3,4) * XS
C      B(1)   = -( T(4,1) - T(4,4) * XS )
C
C      A(2,1) = T(1,2) - T(1,4) * VS
C      A(2,2) = T(2,2) - T(2,4) * VS
C      A(2,3) = T(3,2) - T(3,4) * VS
C      B(2)   = -( T(4,2) - T(4,4) * VS )
C
C      A(3,1) = L(1,1) - L(1,4) * U
C      A(3,2) = L(2,1) - L(2,4) * U
C      A(3,3) = L(3,1) - L(3,4) * U
C      B(3)   = -( L(4,1) - L(4,4) * U )
C
C      RETURN
C
C
C      END
C
*OPTIONS IN EFFECT* ID,FBCCDIC,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 Mapping Program : Eye.Tlmat
C
C Version : This is based on the program of Dr. Martin Altschuler
C Calibration of T matrix and L matrix are not
C performed.
C Instead, T and L matrix are calculated by geometric
C consideration.

C
C DIRECTORY          # OF ROWS OF CONTOUR GRID
C SROW              # OF COLUMNS OF CONTOUR GRID
C SCOL              # OF ROWS IN ENTIRE RASTER SCREEN
C CROW              # OF COLUMNS IN ENTIRE RASTER SCREEN
C CCOL              # OF IMAGES NEEDED TO CODE SCENE
C NIMAGE
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 LOGICAL DEVICES
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 Main program
C
C Local variables
C
C Transformation matrices
C
C REAL T(4,4), L(4,4)
C
C Begin
C
C 0001

```

		MAIN	10-10-83	11:39:12	PAGE P002
0002	C C	Acquire and check the initial parameters.			
	CALL INIT				
0003	C C	Calibrate the system.			
	CALL TMAT(T)				
0004	C C	CALL LMAT(L)			
0005	C C	Solve for the unknown surface.			
	CALL IMGIN				
0006	C C	CALL MAP(T, L)			
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          69.000  
C          70.000  
C          71.000  
C          72.000  
C          73.000  
C          74.000  
C          75.000  
C          76.000  
C          77.000  
C          1.000  
C          2.000  
C          3.000  
C          3.200  
C          3.400  
C          3.600  
C          3.800  
C          4.000  
C          5.000  
C          6.000  
C          6.300  
C          6.600  
C          6.700  
C          6.800  
C          7.000  
C          8.000  
C          9.000  
C          10.000  
C          11.000  
C          12.000  
C          13.000  
C          14.000  
C          14.500  
C          15.000  
C          16.000  
C          17.000  
C          18.000  
C          19.000  
C          20.000  
C          21.000  
C          22.000  
C          23.000  
C          24.000  
C          79.000  
C          80.000  
C          81.000  
C          82.000  
C          83.000  
C          84.000  
C          85.000  
C          86.000  
C          87.000  
C          88.000  
C          89.000  
C          90.000  
C          91.000  
  
0001      C          SUBROUTINE INIT  
          C          Include GLOBAL and 10 common  
          C          C          "GLOBAL" common ( 1, 15 )  
          C          REAL SDIST, STHETA, SPHI  
          C          REAL SSEP, SWIDTH, SHIGH  
          C          C          INTEGER SROW, SCOL  
          C          Dimensions of the shutter grid  
          C          REAL CDIST, CTHETA, CPHI  
          C          REAL CSEP, CWIDTH, CHIGH  
          C          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          "10" COMMON ( 16, 25 )  
          C          INTEGER LUCAL, LUDAT, LUOUT, LUTERM  
          C          Logical units for the calibration file, data, output,  
          C          and the terminal.  
          C          COMMON/IO/  
          1          LUCAL, LUDAT, LUOUT, LUTERM  
          C          C          Begin initialization  
          C          LUCAL = 3  
          C          LUDAT = 4  
          C          LUOUT = 5  
          C          LUTERM = 6  
          C          C          read global variables  
          C          READ (LUCAL, 2000) SDIST, STHETA, SPHI, SSEP, SWIDTH, SHIGH,  
          1          SCOL, SROW
```

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

```

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

```

```

C          107.000
C          108.000
C          109.000
C          110.000
C          111.000
C          112.000
C          113.000
C          114.000
C          115.000
C          116.000
C          117.000
C          1.000
C          2.000
C          3.000
C          3.200
C          3.400
C          3.600
C          3.800
C          4.000
C          5.000
C          6.000
C          6.300
C          6.600
C          6.700
C          6.800
C          7.000
C          8.000
C          9.000
C          10.000
C          11.000
C          12.000
C          13.000
C          14.000
C          14.500
C          15.000
C          16.000
C          17.000
C          18.000
C          19.000
C          20.000
C          21.000
C          22.000
C          23.000
C          24.000
C          26.000
C          27.000
C          28.000
C          29.000
C          30.000
C          31.000
C          32.000
C          33.000
C          34.000
C          35.000
C          36.000

C          SUBROUTINE IMGIN
C          -----
C          INPUT ROUTINE FOR RASTER DATA
C          COMMONS included here
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/
C          1      SDIST, STHETA, SPHI, SSEP, SWIDTH, SHIGH, SRW, SCOL,
C          2      CDIST, CTHETA, CPHI, CSEP, CWIDTH, CHIGH, CRW, CCOL,
C          3      NIMAGE
C          C          "10" COMMON (16,25)
C          INTEGER LUCAL, LUDAT, LUOUT, LUTERM
C          Logical units for the calibration file, data, output,
C          and the terminal.
C          COMMON/IO/
C          1      LUCAL, LUDAT, LUOUT, LUTERM
C          C          "Patterns" common (26,37)
C          INTEGER PAT1(128,128), PAT2(128,128), PAT3(128,128),
C          1      PAT4(128,128), PAT5(128,128), PAT6(128,128),
C          2      PAT7(128,128), PAT8(128,128)
C          C          Contain the image data.
C          COMMON/PATS/
C          1      PAT1, PAT2, PAT3, PAT4, PAT5, PAT6,
C          2      PAT7, PAT8

```

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

IMGIN

PAGE 0002

11:39:13

10-10-83

```

C          C      CALL CLRtbl( PAT1, 128, 128 )
C          C      CALL CLRtbl( PAT2, 128, 128 )
C          C      CALL CLRtbl( PAT3, 128, 128 )
C          C      CALL CLRtbl( PAT4, 128, 128 )
C          C      CALL CLRtbl( PAT5, 128, 128 )
C          C      CALL CLRtbl( PAT6, 128, 128 )
C          C      CALL CLRtbl( PAT7, 128, 128 )
C          C      CALL CLRtbl( PAT8, 128, 128 )

C          C      LOOP THRU DATA UNTIL YOU GET A (-1,-1) ENTRY
C          C      TO SIGNAL END OF PATTERN

C          1      READ (LUDAT, 2000, END=100) I,J
C          IF ((I.EQ.-1) .AND. (J.EQ.-1)) GO TO 2
C          PAT1(I,J) = 1
C          GO TO 1

C          2      READ (LUDAT, 2000, END=100) I,J
C          IF ((I.EQ.-1) .AND. (J.EQ.-1)) GO TO 3
C          PAT2(I,J) = 1
C          GO TO 2

C          3      READ (LUDAT, 2000, END=100) I,J
C          IF ((I.EQ.-1) .AND. (J.EQ.-1)) GO TO 4
C          PAT3(I,J) = 1
C          GO TO 3

C          4      READ (LUDAT, 2000, END=100) I,J
C          IF ((I.EQ.-1) .AND. (J.EQ.-1)) GO TO 5
C          PAT4(I,J) = 1
C          GO TO 4

C          5      READ (LUDAT, 2000, END=100) I,J
C          IF ((I.EQ.-1) .AND. (J.EQ.-1)) GO TO 6
C          PAT5(I,J) = 1
C          GO TO 5

C          6      READ (LUDAT, 2000, END=100) I,J
C          IF ((I.EQ.-1) .AND. (J.EQ.-1)) GO TO 7
C          PAT6(I,J) = 1
C          GO TO 6

C          7      READ (LUDAT, 2000, END=100) I,J
C          IF ((I.EQ.-1) .AND. (J.EQ.-1)) GO TO 8
C          PAT7(I,J) = 1
C          GO TO 7

C          8      READ (LUDAT, 2000, END=100) I,J
C          IF ((I.EQ.-1) .AND. (J.EQ.-1)) GO TO 100
C          PAT8(I,J) = 1
C          GO TO 8

C          119.000
C          120.000
C          121.000
C          122.000
C          123.000
C          124.000
C          125.000
C          126.000
C          127.000
C          128.000
C          129.000
C          130.000
C          131.000
C          132.000
C          133.000
C          134.000
C          135.000
C          136.000
C          137.000
C          138.000
C          139.000
C          140.000
C          141.000
C          142.000
C          143.000
C          144.000
C          145.000
C          146.000
C          147.000
C          148.000
C          149.000
C          150.000
C          151.000
C          152.000
C          153.000
C          154.000
C          155.000
C          156.000
C          157.000
C          158.000
C          159.000
C          160.000
C          161.000
C          162.000
C          163.000
C          164.000
C          165.000
C          166.000
C          167.000
C          168.000
C          169.000
C          170.000
C          171.000

```

```
C      100 RETURN
      2000 FORMAT( 216 )
      3000 FORMAT( 1614 )
C
END
*OPTIONS IN EFFECT* ID,EBCDIC,SOURCE,NOLIST,NOECK,LOAD,NOMAP
*OPTIONS IN EFFECT* NAME = IMGIN   , LINECNT = 57
*STATISTICS* SOURCE STATEMENTS = 57,PROGRAM SIZE = 1710
*STATISTICS* NO DIAGNOSTICS GENERATED
```

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

MAIN 10-10-83 11:39:13 PAGE P001

C 180.000
C 181.000
C 182.000
C 184.000
C 186.000
C 187.000
C 188.000
C 189.000
C 190.000
C 191.000
C 192.000
C 193.000
C 194.000
C 195.000
C 196.000
C 197.000
C 198.000
C 199.000
C 200.000
C 201.000
C 202.000

0001 C SUBROUTINE SUROUT(SURPNT)
C OUTPUT ROUTINE FOR ESTIMATED LOCATION
C OF POINTS

0002 C Commons included here
C "10" COMMON (16,25)
C INTEGER LUCAL, LUDAT, LUOUT, LUTERM
C Logical units for the calibration file, data, output,
C and the terminal.

0003 C COMMON/IO/
1 C LUCAL, LUDAT, LUOUT, LUTERM

0004 C PARAMETERS
C REAL SURPNT(3)
C

0005 C WRITE (LUOUT,2000) SURPNT(1), SURPNT(2), SURPNT(3)
0006 C 2000 FORMAT(3F10.3)

0007 C RETURN
C END

0008 *OPTIONS IN EFFECT* ID,EBCDIC,SOURCE,NOLIST,NOECK,LOAD,NOMAP
OPTIONS IN EFFECT NAME = SUROUT LINECNT = 57
STATISTICS SOURCE STATEMENTS = 8,PROGRAM SIZE = 332
STATISTICS NO DIAGNOSTICS GENERATED

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

MAIN 10-10-83 11:39:14 PAGE P001

```

C          229.000
C          230.000
C          231.000
C          232.000
C          233.000
C          234.000
C          235.000
C          236.000
C          237.000
C          238.000
C          1.000
C          2.000
C          3.000
C          3.200
C          3.400
C          3.600
C          3.800
C          4.000
C          5.000
C          6.000
C          6.300
C          6.600
C          6.700
C          6.800
C          7.000
C          8.000
C          9.000
C          10.000
C          11.000
C          12.000
C          13.000
C          14.000
C          14.500
C          15.000
C          16.000
C          17.000
C          18.000
C          19.000
C          20.000
C          21.000
C          22.000
C          23.000
C          24.000
C          240.000
C          241.000
C          242.000
C          243.000
C          244.000
C          245.000
C          246.000
C          247.000
C          248.000
C          249.000
C          250.000

0001      C          SUBROUTINE LMAT ( LM )
C          C          read in L matrix from phantom
C          C          -----
C          C          Include GLOBAL and 10 Common here
C          C          "GLOBAL" common ( 1, 15 )
C          C          REAL SDIST, STHETA, SPHI
C          C          REAL SSEP, SWIDTH, SHIGH
C          C          INTEGER SROW, SCOL
C          C          Dimensions of the shutter grid
C          C          REAL CDIST, CTTHETA, CPHI
C          C          REAL CSEP, CWIDTH, CHIGH
C          C          INTEGER CROW, CCOL
C          C          Dimensions of the camera grid
C          C          INTEGER NIMAGE
C          C          Number of patterns needed.
C          C          COMMON/GLOBAL/
C          1          SDIST, STHETA, SPHI, SSEP, SWIDTH, SHIGH, SROW, SCOL,
C          2          CDIST, CTTHETA, CPHI, CSEP, CWIDTH, CHIGH, CROW, CCOL,
C          3          NIMAGE
C          C          "10" COMMON ( 16, 25 )
C          C          INTEGER LUCAL, LUDAT, LUOUT, LUTERM
C          C          Logical units for the calibration file, data, output,
C          C          and the terminal.
C          C          COMMON/IO/
C          C          LUCAL, LUDAT, LUOUT, LUTERM
C          C          REAL LM(4,4)
C          C          CALL CLRTBL(LM, 4, 4)
C          C          READ (LUCAL, 2000) ( ( LM(I, J), J = 1, 4 ), I = 1, 4 )
C          C          WRITE (LUTERM, 3000) ( ( 1, J, LM(I, J), J = 1, 4 ), I = 1, 4 )
C          C          2000 FORMAT( F15.7, 3F14.7 )
C          C3000 FORMAT( 4( 214, F14.7 ) )

```

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

252.000
253.000

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

PAGE PO01

11:39:14

MAIN 10-10-83

```

C          254.000
C          255.000
C          256.000
C          257.000
C          258.000
C          259.000
C          260.000
C          261.000
C          262.000
C          263.000
C          264.000
C          265.000
C          266.000
C          267.000
C          269.000
C          270.000
C          271.000
C          272.000
C          273.000
C          274.000
C          275.000
C          276.000
C          277.000
C          278.000
C          279.000
C          280.000
C          281.000
C          282.000
C          283.000
C          284.000
C          285.000
C          286.000
C          287.000
C          1.000
C          2.000
C          3.000
C          3.200
C          3.400
C          3.600
C          3.800
C          4.000
C          5.000
C          6.000
C          6.300
C          6.600
C          6.700
C          6.800
C          7.000
C          8.000
C          9.000
C          10.000
C          11.000
C          12.000

C-----0001-----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          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 squares equations
C
C-----0002-----C
C          Parameters:
C          TM - Transformation matrix for camera geometry
C          LM - Equation of the plane for each column of the shutter
C
C-----0003-----C
C          Include all commons here
C          "GLOBAL" common (1,15)
C
C-----0004-----C
C          REAL SDIST, STHETA, SPHI
C          REAL SSEP, SWIDTH, SHIGH
C          INTEGER SROW, SCOL
C          Dimensions of the shutter grid
C
C-----0005-----C
C          REAL CDIST, CTHETA, CPHI
C          REAL CSEP, CWIDTH, CHIGH
C
C-----0006-----C
C          INTEGER CROW, CCOL
C          Dimensions of the camera grid
C
C-----0007-----C
C          INTEGER NIMAGE
C          Number of patterns needed.
C
C-----0008-----C
C          COMMON/GNRA1 /

```

```

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

```

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

10-10-83

PAGE P003

11:39:14

```

C          CALL SETCOF(ILSA,LSB,XSTAR,YSTAR,COL,TM,LW)      MAP
0030      C          322.000
C          CALL L2(3,3,LSA,LSB,LSX,EPS,ITS,PRT1,PRT2,3,QL)    323.000
0031      PRT1 = 0   324.000
0032      PRT2 = 0   325.000
0033      C          326.000
C          CALL LEQT2F(ILSA,1,3,3,LSB,O,WKAREA,QL)           327.000
C          CALL SUROUT(LSB)                                 328.000
C          WRITE(LUDUT,100) U, I, J, LSX(1), LSX(2), LSX(3)  329.000
0034      Q          330.000
C          IF(QL .GT. QLMAX) GO TO 10                      331.000
C          CALL SUROUT(LSB)                                 332.000
C          C%/%/%/%/                                         333.000
C          WRITE(LUDUT,100) U, I, J, LSX(1), LSX(2), LSX(3)  334.000
0035      C          335.000
C          10        CONTINUE                                336.000
0036      0037      20        CONTINUE                                337.000
C          100       FORMAT(1X,3I6,3F14.7)                   338.000
C          RETURN                                              339.000
0038      C          END                                     340.000
0039      C          *OPTIONS IN EFFECT* ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
0040      *OPTIONS IN EFFECT* NAME = MAP                   57
*STATISTICS* SOURCE STATEMENTS = 40,PROGRAM SIZE = 1832
*STATISTICS* NO DIAGNOSTICS GENERATED

```

```

C      C      SUBROUTINE COLUMN (RW,CL,U)
C
C      C      Column finds the shutter column (u) for the given point by looking
C      C      at the images the point appeared in. It is assumed that pattern two
C      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      C      "GLOBAL" common ( 1,15 )
C
C      C      REAL SDIST, STHETA, SPHI
C
C      C      REAL SSEP, SWIDTH, SHIGH
C
C      C      INTEGER SROW, SCOL
C              Dimensions of the shutter grid
C
C      C      REAL CDIST, CTHETA, CPHI
C
C      C      REAL CSEP, CWIDTH, CHIGH
C
C      C      INTEGER CROW, CCOL
C              Dimensions of the camera grid
C
C      C      INTEGER NIMAGE
C              Number of patterns needed.
C
C      C      COMMON/GLOBAL/
C              SDIST, STHETA, SPHI, SSEP, SWIDTH, SHIGH, SROW, SCOL,
C              CDIST, CTHETA, CPHI, CSEP, CWIDTH, CHIGH, CROW, CCOL,
C              NIMAGE
C
C      C      "Patterns" common ( 26,37 )
C
C      C      INTEGER PAT1(128,128), PAT2(128,128), PAT3(128,128),
C              1          PAT4(128,128), PAT5(128,128), PAT6(128,128),
C              2          PAT7(128,128), PAT8(128,128)
C
C              C      Contain the image data.
C
C      C      COMMON/PATS/
C              PAT1, PAT2, PAT3, PAT4, PAT5, PAT6,
C              1          PAT7, PAT8
C
C      C      INTEGER RW,CL,U
C
C      C      begin
C
C      C      346.000
C      C      347.000
C      C      348.000
C      C      349.000
C      C      350.000
C
C      C      351.000
C      C      352.000
C      C      353.000
C      C      354.000
C      C      355.000
C
C      C      356.000
C      C      357.000
C      C      358.000
C      C      359.000
C
C      C      360.000
C      C      361.000
C      C      1.000
C      C      2.000
C      C      3.000
C      C      3.200
C      C      3.400
C      C      3.600
C      C      3.800
C      C      4.000
C
C      C      5.000
C      C      6.000
C      C      6.300
C      C      6.600
C      C      6.700
C      C      6.800
C      C      7.000
C      C      8.000
C      C      9.000
C      C      10.000
C      C      11.000
C      C      12.000
C      C      13.000
C      C      14.000
C      C      14.500
C      C      15.000
C      C      15.500
C      C      16.000
C      C      16.500
C      C      17.000
C      C      17.500
C      C      18.000
C      C      18.500
C      C      19.000
C      C      19.500
C      C      20.000
C      C      21.000
C      C      22.000
C      C      23.000
C      C      24.000
C      C      25.000
C      C      26.000
C      C      27.000
C      C      28.000
C      C      29.000
C      C      30.000
C      C      31.000
C      C      32.000
C      C      33.000
C      C      34.000
C      C      35.000
C      C      36.000
C      C      364.000
C      C      365.000
C      C      366.000
C      C      367.000

```

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

PAGE P002

11:39:16

10-10-83

PAGE P002

```

0013      C      U
          = PAT2(RW,CL) * ( 2 ** (NIMAGE-2))
          + PAT3(RW,CL) * ( 2 ** (NIMAGE-3))
          + PAT4(RW,CL) * ( 2 ** (NIMAGE-4))
          + PAT5(RW,CL) * ( 2 ** (NIMAGE-5))
          + PAT6(RW,CL) * ( 2 ** (NIMAGE-6))
          + PAT7(RW,CL) * ( 2 ** (NIMAGE-7))
          + PAT8(RW,CL) * ( 2 ** (NIMAGE-8))
          + 1

0014      C
          END
          *OPTIONS IN EFFECT*  ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
          *OPTIONS IN EFFECT*  NAME = COLUMN      LINECNT = 57
          SOURCE STATEMENTS = 15,PROGRAM SIZE = 986
          *STATISTICS*
          *STATISTICS* NO DIAGNOSTICS GENERATED

```

368.000

369.000

370.000

371.000

372.000

373.000

374.000

375.000

376.000

377.000

378.000

379.000

```

C-----+
C          C SUBROUTINE SETCOF(A, B, XS, YS, U, T, L)
C          C Setcoef sets the coefficients on the following three
C          C equations which are used to solve for the point (x,y,z):
C          C
C          C (T11-T14*x) + (T21-T24*x)*z + (T31-T34*x)*z = 0
C          C (T12-T14*y) + (T22-T24*y)*z + (T32-T34*y)*z = 0
C          C (L11-L14*u) + (L21-L24*u)*y + (L31-L34*u)*z + (L41-L44*u) = 0
C          C
C          C Parameters:
C          C
C          C Input: XS - x* in the equations
C          C         YS - y* in the equations
C          C         U - column of the shutter
C          C         T - Transformation matrix for camera geometry
C          C         L - Equation of the plane for each shutter column
C          C
C          C Output: A, B: Least square matrices
C          C
C          C-----+
C          C          REAL XS, YS, U, T(4,4), L(4,4), A(3,3), B(3)
C          C
C          C          Begin.:
C          C          A(1,1) = T(1,1) - T(1,4) * XS
C          C          A(1,2) = T(2,1) - T(2,4) * XS
C          C          A(1,3) = T(3,1) - T(3,4) * XS
C          C          B(1) = - ( T(4,1) - T(4,4) * XS )
C          C
C          C          A(2,1) = T(1,2) - T(1,4) * YS
C          C          A(2,2) = T(2,2) - T(2,4) * YS
C          C          A(2,3) = T(3,2) - T(3,4) * YS
C          C          B(2) = - ( T(4,2) - T(4,4) * YS )
C          C
C          C          A(3,1) = L(1,1) - L(1,4) * U
C          C          A(3,2) = L(2,1) - L(2,4) * U
C          C          A(3,3) = L(3,1) - L(3,4) * U
C          C          B(3) = - ( L(4,1) - L(4,4) * U )
C          C
C          C          RETURN
C          C-----+
C          C          END
C          *OPTIONS IN EFFECT* ID,EBCDIC,SOURCE,NOLIST,NOECK,LOAD,NOMAP
C          *OPTIONS IN EFFECT* NAME = SETCOF . LINECNT = 57
C          *STATISTICS* SOURCE STATEMENTS = 16,PROGRAM SIZE = 614
C          *STATISTICS* NO DIAGNOSTICS GENERATED

```

NO STATEMENTS FLAGGED IN THE ABOVE COMPILENTIONS.

8.3. Eye.Tlmat Program

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

MAIN 10-10-83 17:05:51 PAGE P001

```
C      Mapping Program : Eye.TLMat
```

```
C Version : This is based on the program of Dr. Martin Altshuler C
C Calibration of T matrix and L matrix are not
C performed.
```

```
C Instead, T and L matrix are calculated by geometric
C consideration.
```

```
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 IMAGES 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 PAT7          MATRIX OF RASTER IMAGE # 7
C
C LOGICAL DEVICES
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 Main program
```

```
C Local variables
```

```
C Transformation matrices
```

```
C REAL T(4,4). L(4,4)
```

```

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

```

MICHIGAN TERMINAL SYSTEM FOR IRAN G(21-8)

MAIN 10-10-83

PAGE P001

17:05·51

```

0017      READ ( LUCAL, 2000 ) CDIST, CTHETA, CPHI, CSEP, CWIDTH, CHIGH,
1
0018      2000 FORMAT ( F9.4, 5F8.4, 216 )
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
END
0024      *OPTIONS IN EFFECT* ID,EBCDIC,SOURCE,NOLIST,NOECK,LOAD,NOMAP
*OPTIONS IN EFFECT* NAME = INIT   . LINECNT = 57
*STATISTICS* SOURCE STATEMENTS = 24,PROGRAM SIZE = 692
*STATISTICS* NO DIAGNOSTICS GENERATED

```

```

C          107.000
C          108.000
C          109.000
C          110.000
C          111.000
C          112.000
C          113.000
C          114.000
C          115.000
C          116.000
C          117.000
C          1.000
C          2.000
C          3.000
C          3.200
C          3.400
C          3.600
C          3.800
C          4.000
C          5.000
C          6.000
C          6.300
C          6.600
C          6.700
C          6.800
C          7.000
C          8.000
C          9.000
C          10.000
C          11.000
C          12.000
C          13.000
C          14.000
C          14.500
C          15.000
C          16.000
C          17.000
C          18.000
C          19.000
C          20.000
C          21.000
C          22.000
C          23.000
C          24.000
C          26.000
C          27.000
C          28.000
C          29.000
C          30.000
C          31.000
C          32.000
C          33.000
C          34.000
C          35.000

C          10-10-83

0001      C          SUBROUTINE IMGIN

0002      C          INPUT ROUTINE FOR RASTER DATA

0003      C          COMMONS included here

0004      C          "GLOBAL" COMMON (1,15)

0005      C          REAL SDIST, STHETA, SPHI
0006      C          REAL SSEP, SWIDTH, SHIGH
0007      C          INTEGER SROW, SCOL
0008      C          Dimensions of the shutter grid
0009      C          REAL CDIST, CTTHETA, CPHI
0010      C          REAL CSEP, CWIDTH, CHIGH
0011      C          INTEGER CROW, CCOL
0012      C          Dimensions of the camera grid
0013      C          INTEGER NIMAGE
0014      C          Number of patterns needed.

0015      C          COMMON/GLOBAL/
0016      C          SDIST, STHETA, SPHI, SSEP, SWIDTH, SHIGH, SROW, SCOL,
0017      C          CDIST, CTTHETA, CPHI, CSEP, CWIDTH, CHIGH, CROW, CCOL,
0018      C          NIMAGE

0019      C          "10" COMMON (16,25)

0020      C          INTEGER LUCAL, LUDAT, LUOUT, LUTERM
0021      C          Logical units for the calibration file, data, output,
0022      C          and the terminal.

0023      C          COMMON/IO/
0024      C          1          LUCAL, LUDAT, LUOUT, LUTERM
0025      C          "Patterns" common (26,37)

0026      C          INTEGER PAT1(128,128), PAT2(128,128), PAT3(128,128),
0027      C          1          PAT4(128,128), PAT5(128,128), PAT6(128,128),
0028      C          2          PAT7(128,128), PAT8(128,128),
0029      C          Contain the image data.

0030      C          COMMON/PATS/
0031      C          1          PAT1, PAT2, PAT3, PAT4, PAT5, PAT6.

```

```

C   C CALL CLRtbl( PAT1, 128, 128 )
C   C CALL CLRtbl( PAT2, 128, 128 )
C   C CALL CLRtbl( PAT3, 128, 128 )
C   C CALL CLRtbl( PAT4, 128, 128 )
C   C CALL CLRtbl( PAT5, 128, 128 )
C   C CALL CLRtbl( PAT6, 128, 128 )
C   C CALL CLRtbl( PAT7, 128, 128 )
C   C CALL CLRtbl( PAT8, 128, 128 )

C   C LOOP THRU DATA UNTIL YOU GET A (-1,-1) ENTRY
C   C TO SIGNAL END OF PATTERN

C   1 READ (LUDAT, 2000, END=100) I,J
C     IF ((I.EQ.-1) .AND. (J.EQ.-1)) GO TO 2
C       PAT1(I,J) = 1
C     GO TO 1

C   2 READ (LUDAT, 2000, END=100) I,J
C     IF ((I.EQ.-1) .AND. (J.EQ.-1)) GO TO 3
C       PAT2(I,J) = 1
C     GO TO 2

C   3 READ (LUDAT, 2000, END=100) I,J
C     IF ((I.EQ.-1) .AND. (J.EQ.-1)) GO TO 4
C       PAT3(I,J) = 1
C     GO TO 3

C   4 READ (LUDAT, 2000, END=100) I,J
C     IF ((I.EQ.-1) .AND. (J.EQ.-1)) GO TO 5
C       PAT4(I,J) = 1
C     GO TO 4

C   5 READ (LUDAT, 2000, END=100) I,J
C     IF ((I.EQ.-1) .AND. (J.EQ.-1)) GO TO 6
C       PAT5(I,J) = 1
C     GO TO 5

C   6 READ (LUDAT, 2000, END=100) I,J
C     IF ((I.EQ.-1) .AND. (J.EQ.-1)) GO TO 7
C       PAT6(I,J) = 1
C     GO TO 6

C   7 READ (LUDAT, 2000, END=100) I,J
C     IF ((I.EQ.-1) .AND. (J.EQ.-1)) GO TO 8
C       PAT7(I,J) = 1
C     GO TO 7

C   8 READ (LUDAT, 2000, END=100) I,J
C     IF ((I.EQ.-1) .AND. (J.EQ.-1)) GO TO 100
C       PAT8(I,J) = 1
C     GO TO 8

C   C WRITE( LUTERM, 3000 ) (( PAT1(I,J), J=1,CCOL ), I=1,CROW )

C014      119.000
C015      120.000
C016      121.000
C017      122.000
C018      123.000
C019      124.000
C020      125.000
C021      126.000
C022      127.000
C023      128.000
C024      129.000
C025      130.000
C026      131.000
C027      132.000
C028      133.000
C029      134.000
C030      135.000
C031      136.000
C032      137.000
C033      138.000
C034      139.000
C035      140.000
C036      141.000
C037      142.000
C038      143.000
C039      144.000
C040      145.000
C041      146.000
C042      147.000
C043      148.000
C044      149.000
C045      150.000
C046      151.000
C047      152.000
C048      153.000
C049      154.000
C050      155.000
C051      156.000
C052      157.000
C053      158.000
C054      159.000
C055      160.000
C056      161.000
C057      162.000
C058      163.000
C059      164.000
C060      165.000
C061      166.000
C062      167.000
C063      168.000
C064      169.000
C065      170.000
C066      171.000
C067      172.000
C068      173.000

```

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

	IMGIN	10-10-83	17:05:52	PAGE P003
C				174.000
0054	100 RETURN			175.000
0055	2000 FORMAT(216)			176.000
0056	3000 FORMAT(1614)			177.000
0057	C END			178.000
	OPTIONS IN EFFECT ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP			179.000
	OPTIONS IN EFFECT NAME = IMGIN LINECNT = 57			
	STATISTICS SOURCE STATEMENTS = 57,PROGRAM SIZE = 1710			
	STATISTICS NO DIAGNOSTICS GENERATED			

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

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)		MAIN	10-10-83	17:05:53	PAGE 0001
C	C	C	203.000	204.000	205.000
C	C	C	206.000		

```

0001      C      SUBROUTINE TMAT ( TM )
0002      C      read in T matrix from phantom
0003      C      include 10 common here
0004      C      "10" COMMON ( 16,25 )
0005      C      INTEGER LUCAL, LUOUT, LUTERM
0006      C      Logical units for the calibration file, data, output,
0007      C      and the terminal.
0008      C      COMMON/10/
0009      C      LUCAL, LUOUT, LUOUT, LUTERM
0010      C      REAL TM(4, 4)
0011      C      CALL CLRtbl(TM, 4, 4)
0012      C      READ (LUCAL,2000) (( TM(I, J), J = 1, 4), I = 1, 4)
0013      C      WRITE (LUTERM,3000) (( 1, J, TM(I, J), J = 1, 4), I = 1, 4)
0014      C      2000 FORMAT( F15.7, 3F14.7 )
0015      C      C3000 FORMAT( 4( 214, F14.7 ) )
0016      C      RETURN
0017      END
0018      *OPTIONS IN EFFECT*  ID, EBCDIC, SOURCE, NOLIST, NODECK, LOAD, NOMAP
0019      *OPTIONS IN EFFECT*  NAME = TMAT   LINECNT = 57
0020      *STATISTICS*   SOURCE STATEMENTS = 9, PROGRAM SIZE =
0021      *STATISTICS*   NO DIAGNOSTICS GENERATED

```

```

C
C
C
C      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      C   REAL SDIST, STHETA, SPHI
C      C   REAL SSEP, SWIDTH, SHIGH
C
C      C   INTEGER SROW, SCOL
C      C   Dimensions of the shutter grid
C
C      C   REAL CDIST, CTTHETA, CPHI
C      C   REAL CSEP, CWIDTH, CHIGH
C
C      C   INTEGER CROW, CCOL
C      C   Dimensions of the camera grid
C
C      C   INTEGER NJIMAGE
C      C   Dimensions of the camera grid
C
C      C   Number of patterns needed.
C
C      C   COMMON/GLOBAL/
C      C   1      SDIST, STHETA, SPHI, SSEP, SWIDTH, SHIGH, SROW, SCOL,
C      C   2      CDIST, CTTHETA, CPHI, CSEP, CWIDTH, CHIGH, CROW, CCOL,
C      C   3      NJIMAGE
C
C      C   "IO" COMMON ( 16, 25 )
C
C      C   INTEGER LUCAL, LUOUT, LUOUT, LUTERM
C      C   Logical units for the calibration file, data, output,
C      C   and the terminal.
C
C      C   COMMON/IO/
C      C   1      LUCAL, LUOUT, LUOUT, LUTERM
C
C      C   REAL LM(4,4)
C      C   CALL CLRtbl(LM, 4, 4)
C
C      C   READ (LUCAL, 2000) ( ( LM(I, J), J = 1, 4 ), I = 1, 4 )
C      C   WRITE (LUTERM, 3000) ( ( I, J, LM(I, J), J = 1, 4 ), I = 1, 4 )
C
C      C   2000  FORMAT( F15.7, 3F14.7 )
C      C3000  FORMAT( 4( 214, F14.7 ) )
C
C
C      229.000
C      230.000
C      231.000
C      232.000
C      233.000
C      234.000
C      235.000
C      236.000
C      237.000
C      238.000
C      1.000
C      2.000
C      3.000
C      3.200
C      3.400
C      3.600
C      3.800
C      4.000
C      5.000
C      6.000
C      6.300
C      6.600
C      6.700
C      6.800
C      7.000
C      8.000
C      9.000
C      10.000
C      11.000
C      12.000
C      13.000
C      14.000
C      14.500
C      15.000
C      16.000
C      17.000
C      18.000
C      19.000
C      20.000
C      21.000
C      22.000
C      23.000
C      24.000
C      240.000
C      241.000
C      242.000
C      243.000
C      244.000
C      245.000
C      246.000
C      247.000
C      248.000
C      249.000
C      250.000
C      251.000

```

	MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)	LMAT	10-10-83	17:05:54	PAGE P002
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      254.000
C      255.000
C      256.000
C      257.000
C      258.000
C      259.000
C      260.000
C      261.000
C      262.000
C      263.000
C      264.000
C      265.000
C      266.000
C      267.000
C      268.000
C      269.000
C      270.000
C      271.000
C      272.000
C      273.000
C      274.000
C      275.000
C      276.000
C      277.000
C      278.000
C      279.000
C      280.000
C      281.000
C      282.000
C      283.000
C      284.000
C      285.000
C      286.000
C      287.000
C      1.000
C      2.000
C      3.000
C      3.200
C      3.400
C      3.600
C      3.800
C      4.000
C      5.000
C      6.000
C      6.300
C      6.600
C      6.700
C      6.800
C      7.000
C      8.000
C      9.000
C      10.000
C      11.000
C      12.000
C      13.000
C      14.000

C      MAP Module
C      This module does the mapping of the unknown surface.
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 squares equations
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      Include all commons here
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/
C      1 SDIST, STHETA, SPHI, SSEP, SWIDTH, SHIGH, SROW, SCOL,

```

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8) MAP 10-10-83 17:05:54 PAGE P002

```

      2      CDIST, CTHETA, CPHI, CSEP, CWIDTH, CHIGH, CROW, CCOL, 14.500
      3      NIMAGE 15.000
      C      "10" COMMON (16.25) 16.000
      C      17.000
      C      INTEGER LUCAL, LUDAT, LUOUT, LUTERM 18.000
      C      Logical units for the calibration file, data, output, 19.000
      C      and the terminal. 20.000
      C      COMMON/10/ 21.000
      1      LUCAL, LUDAT, LUOUT, LUTERM 22.000
      C      23.000
      C      "Patterns" common (26.37) 24.000
      C      25.000
      C      INTEGER PAT1(128,128), PAT2(128,128), PAT3(128,128), 26.000
      1      PAT4(128,128), PAT5(128,128), PAT6(128,128), 27.000
      2      PAT7(128,128), PAT8(128,128) 28.000
      C      Contain the image data. 29.000
      C      COMMON/PATS/ 30.000
      1      PAT1, PAT2, PAT3, PAT4, PAT5, PAT6, 31.000
      2      PAT7, PAT8 32.000
      C      REAL TM(4,4), LM(4,4) 33.000
      0013     C      Local variables: 34.000
      C      Matrices for least squares AX=B 35.000
      0014     C      REAL LSA(3,3), LSB(3), LSX(3) 36.000
      C      Known values in least squares equations 37.000
      C      REAL COL, XSTAR, VSTAR 38.000
      C      INTEGER U 39.000
      C      More least square parms 40.000
      0015     C      REAL EPS/4.768E-07/, QL, QLMAX/1.E07/, WKAREA(200) 41.000
      C      INTEGER ITS/10/, PRT1, PRT2 42.000
      0016     C      Begin... 43.000
      0017     C      DO 20 J = 1, CCOL 44.000
      0018     C      DO 10 I = 1, CROW 45.000
      0019     C      IF (PAT1(I,J) .EQ. 0) GO TO 10 46.000
      0020     C      CALL CLRVC (LSB,3) 47.000
      0021     C      CALL CLRVC (LSX,3) 48.000
      0022     C      CALL CLRVC (LSA,3,3) 49.000
      0023     C      CALL COLUMN(I,J,U) 50.000
      0024     C      XSTAR = FLOAT(I) 51.000
      0025     C      VSTAR = FINAT(I) 52.000
      0026     C      ... 53.000
      0027     C      ... 54.000
      0028     C      ...
  
```

```

C          CALL SETCOF(LSA,LSB,XSTAR,YSTAR,COL,TM,LN)
322.000
323.000
324.000
325.000
326.000
327.000
328.000
329.000
330.000
331.000
332.000
333.000
334.000
335.000
336.000
337.000
338.000
339.000
340.000
342.000
343.000
344.000
345.000

0030      C
0031      C          PRT1 = 0
0032      C          PRT2 = 0
0033      C          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)
C
0034      C          IF (QL .GT. QLMAX) GO TO 10
C
C/%%%%%%   CALL SUROUT( LSB )
C
0035      C          WRITE (LUOUT, 100) U, I, J, LSX(1), LSX(2), LSX(3)
C
0036      C          10  CONTINUE
0037      C          20  CONTINUE
0038      C          100 FORMAT (1X, 3I6, 3F14.7 )
C
0039      C          RETURN
END
*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

```

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

MAIN 10-10-83 17:05:55 PAGE P001

```

C          346.000
C          347.000
C          348.000
C          349.000
C          350.000
C          351.000
C          352.000
C          353.000
C          354.000
C          355.000
C          356.000
C          357.000
C          358.000
C          359.000
C          360.000
C          361.000
C          1.000
C          2.000
C          3.000
C          3.200
C          3.400
C          3.600
C          3.800
C          4.000
C          5.000
C          6.000
C          6.300
C          6.600
C          6.700
C          6.800
C          7.000
C          8.000
C          9.000
C          10.000
C          11.000
C          12.000
C          13.000
C          14.000
C          14.500
C          15.000
C          26.000
C          27.000
C          28.000
C          29.000
C          30.000
C          31.000
C          32.000
C          33.000
C          34.000
C          35.000
C          36.000
C          364.000
C          365.000
C          366.000

C          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          Include GLOBAL and PATTERN commons here
C
C          "GLOBAL" common ( 1.15 )
C          REAL SDIST, STHETA, SPHI
C          REAL SSEP, SWIDTH, SHIGH
C
C          INTEGER SROW, SCOL
C          Dimensions of the shutter grid
C
C          REAL CDIST, CTHETA, CPHI
C
C          REAL CSEP, CWIDTH, CHIGH
C
C          INTEGER CROW, CCOL
C          Dimensions of the camera grid
C
C          INTEGER NIMAGE
C          Number of patterns needed.
C
C          COMMON/GLOBAL/
C          SDIST, STHETA, SPHI, SSEP, SWIDTH, SHIGH, SROW, SCOL,
C          CDIST, CTHETA, CPHI, CSEP, CWIDTH, CHIGH, CROW, CCOL,
C          NIMAGE
C
C          "Patterns" common ( 26, 37 )
C
C          INTEGER PAT1(128,128), PAT2(128,128), PAT3(128,128),
C          1      PAT4(128,128), PAT5(128,128), PAT6(128,128),
C          2      PAT7(128,128), PAT8(128,128)
C
C          Contain the image data.
C
C          COMMON/PATS/
C          1      PAT1, PAT2, PAT3, PAT4, PAT5, PAT6,
C          2      PAT7, PAT8
C
C          INTEGER RW, CL,U

```

```

0013      C          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      C          RETURN
           C          END
0015      *OPTIONS IN EFFECT* ID,BCDIC,SOURCE,NOLIST,NOECK,LOAD,NOMAP
           *OPTIONS IN EFFECT* NAME = COLUMN    LINECNT = 57
           *STATISTICS* SOURCE STATEMENTS = 15,PROGRAM SIZE = 986
           *STATISTICS* NO DIAGNOSTICS GENERATED

```

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

PAGE P001

MAIN

17:05:56

PAGE P001

MAIN

17:05:56

```

C          380.000
C          381.000
C          382.000
C          383.000
C          384.000
C          385.000
C          386.000
C          387.000
C          388.000
C          389.000
C          390.000
C          391.000
C          392.000
C          393.000
C          394.000
C          395.000
C          396.000
C          397.000
C          398.000
C          399.000
C          400.000
C          401.000
C          402.000
C          403.000
C          404.000
C          405.000
C          406.000
C          407.000
C          408.000
C          409.000
C          410.000
C          411.000
C          412.000
C          413.000
C          414.000
C          415.000
C          416.000
C          417.000
C          418.000
C          419.000
C          420.000
C          421.000
C          422.000
C          423.000
C          424.000

0001      C          SUBROUTINE SETCOF(A, B, XS, VS, U, T, L)
C          Setcof sets the coefficients on the following three
C          equations which are used to solve for the point (x,y,z):
C          (T11-T14*x)*x + (T21-T24*x)*y + (T31-T34*x)*z + (T41-T44*x) * 0
C          (T12-T14*y)*x + (T22-T24*y)*y + (T32-T34*y)*z + (T42-T44*y) * 0
C          (L11-L14*u)*x + (L21-L24*u)*y + (L31-L34*u)*z + (L41-L44*u) * 0

0002      C          Parameters:
C          Input:   XS - x* in the equations
C          VS - 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          Output: A, B: Least square matrices
C          REAL XS, VS, U, T(4,4), L(4,4), A(3,3), B(3)

0003      C          Begin...
C          A(1,1) = T(1,1) - T(1,4) * XS
C          A(1,2) = T(2,1) - T(2,4) * XS
C          A(1,3) = T(3,1) - T(3,4) * XS
C          B(1) = - ( T(4,1) - T(4,4) * XS )
C          A(2,1) = T(1,2) - T(1,4) * VS
C          A(2,2) = T(2,2) - T(2,4) * VS
C          A(2,3) = T(3,2) - T(3,4) * VS
C          B(2) = - ( T(4,2) - T(4,4) * VS )
C          A(3,1) = L(1,1) - L(1,4) * U
C          A(3,2) = L(2,1) - L(2,4) * U
C          A(3,3) = L(3,1) - L(3,4) * U
C          B(3) = - ( L(4,1) - L(4,4) * U )

0007      C          RETURN
0008      END
0009      *OPTIONS IN EFFECT* ID, EBCDIC, SOURCE, NOLIST, NODECK, LOAD, NOMAP
0010      *OPTIONS IN EFFECT* NAME = SETCOF    LINECNT = 57
0011      *STATISTICS* SOURCE STATEMENTS = 16, PROGRAM SIZE = 614
0012      *STATISTICS* NO DIAGNOSTICS GENERATED
0013
0014
0015
0016

```

NO STATEMENTS FLAGGED IN THE ABOVE COMPILENTIONS.

MICHIGAN TERMINAL SYSTEM FORTRAN G(21:8)

MAIN PAGE P001

16:41:50

10-10-83

```

C      1.000      3.000      6.000      7.000      7.500
C      2.000      4.000      6.000      7.000      7.700
C      3.000      5.000      7.000      8.000      8.500
C      4.000      6.000      8.000      9.000      9.500
C      5.000      7.000      9.000      10.000     11.000
C      6.000      8.000      10.000     11.500     12.000
C      7.000      9.000      11.000     13.000     14.000
C      8.000      10.000     12.000     15.000     16.000
C      9.000      11.000     13.000     17.000     18.000
C      10.000     12.000     14.000     19.000     20.000
C      11.000     13.000     15.000     21.000     22.000
C      12.000     14.000     16.000     23.000     25.000
C      13.000     15.000     17.000     26.000     26.500
C      14.000     16.000     18.000     27.000     28.000
C      15.000     17.000     19.000     29.000     30.000
C      16.000     18.000     20.000     31.000     32.000
C      17.000     19.000     21.000     33.000     34.000
C      18.000     20.000     22.000     35.000     36.000
C      19.000     21.000     23.000     36.500     37.000
C      20.000     22.000     24.000     38.000     39.000
C      21.000     23.000     25.000     40.000     41.000
C      22.000     24.000     26.000     42.000     42.500
C      23.000     25.000     27.000     43.000     44.000
C      24.000     26.000     28.000     44.000     45.000
C      25.000     27.000     29.000     45.000     46.000
C      26.000     28.000     30.000     46.000     47.000
C      27.000     29.000     31.000     48.000     49.000

C THIS SUBROUTINE SOLVES THE LINEAR EQUATIONS AX = B.
C
C A -- M X N matrix
C EPS -- threshold for the singular values.
C       Those S.V.'s (in ARRAY Q) smaller than EPS * QMAX
C       are set to zero.
C EPS1 -- precision required 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 REFERENCE: G.H. GOLUB & C. REINSCH
C NUMER. MATH. 14, 403-420, (1970).
C FORTRAN BY ALAN CLINE, NCAR, BOULDER, CO., (1970).

C SUBROUTINES CALLED: SVD S1

C
C COMMON/BILKA/LDTA,LCAL,LGC1,LUNOUT,INFL,IM1,IM2,LCD
C COMMON/SUB/U(200,200),V(11,11),Q(11),DX(11),R(200)
C REAL A(M1,N),B(M),X(N)
C DOUBLE PRECISION S
C DATA EPS1/1.E-05/
C
C begin
C LUNOUT ALONE INITIALIZED HERE TO MINIMIZE INTERFACE
C WITH REST OF PROGRAM AND MINIMIZE CHANGES TO CANNED PROGRAM.
C
C LUNOUT=6
C CALL SVD(A,M,N,1,1,MI1)
C
C QL=Q(1)
C QH=Q(1)
C DO 2 J=2,N
C       QH=AMAX1(Q(J),QH)
C       QL=AMIN1(Q(J),QL)
C 2 CONTINUE
C
C QL=QH/QL
C
C DO 12 FO=1 WRITE(LUNOUT,104) QL,(Q(I)),I=1,N
C
C 0001      SUBROUTINE L2(M,N,A,B,X,EPS,I1,I2,I3,M1,QL)
C
C 0002      0003      0004      0005      0006
C
C 0007      0008      0009      0010      0011      0012      0013      0014      0015      0016
C
C 0011      0012      0013      0014      0015      0016
C
C 0016      0017      0018      0019      0020      0021      0022      0023      0024      0025
C
C 0025      0026      0027      0028      0029      0030      0031      0032      0033      0034
C
C 0034      0035      0036      0037      0038      0039      0040      0041      0042      0043
C
C 0043      0044      0045      0046      0047      0048      0049      0050      0051      0052
C
C 0052      0053      0054      0055      0056      0057      0058      0059      0060      0061
C
C 0061      0062      0063      0064      0065      0066      0067      0068      0069      0070
C
C 0070      0071      0072      0073      0074      0075      0076      0077      0078      0079
C
C 0079      0080      0081      0082      0083      0084      0085      0086      0087      0088
C
C 0088      0089      0090      0091      0092      0093      0094      0095      0096      0097
C
C 0097      0098      0099      0100      0101      0102      0103      0104      0105      0106
C
C 0106      0107      0108      0109      0110      0111      0112      0113      0114      0115
C
C 0115      0116      0117      0118      0119      0120      0121      0122      0123      0124
C
C 0124      0125      0126      0127      0128      0129      0130      0131      0132      0133
C
C 0133      0134      0135      0136      0137      0138      0139      0140      0141      0142
C
C 0142      0143      0144      0145      0146      0147      0148      0149      0150      0151
C
C 0151      0152      0153      0154      0155      0156      0157      0158      0159      0160
C
C 0160      0161      0162      0163      0164      0165      0166      0167      0168      0169
C
C 0169      0170      0171      0172      0173      0174      0175      0176      0177      0178
C
C 0178      0179      0180      0181      0182      0183      0184      0185      0186      0187
C
C 0187      0188      0189      0190      0191      0192      0193      0194      0195      0196
C
C 0196      0197      0198      0199      0200      0201      0202      0203      0204      0205
C
C 0205      0206      0207      0208      0209      0210      0211      0212      0213      0214
C
C 0214      0215      0216      0217      0218      0219      0220      0221      0222      0223
C
C 0223      0224      0225      0226      0227      0228      0229      0230      0231      0232
C
C 0232      0233      0234      0235      0236      0237      0238      0239      0240      0241
C
C 0241      0242      0243      0244      0245      0246      0247      0248      0249      0250
C
C 0250      0251      0252      0253      0254      0255      0256      0257      0258      0259
C
C 0259      0260      0261      0262      0263      0264      0265      0266      0267      0268
C
C 0268      0269      0270      0271      0272      0273      0274      0275      0276      0277
C
C 0277      0278      0279      0280      0281      0282      0283      0284      0285      0286
C
C 0286      0287      0288      0289      0290      0291      0292      0293      0294      0295
C
C 0295      0296      0297      0298      0299      0300      0301      0302      0303      0304
C
C 0304      0305      0306      0307      0308      0309      0310      0311      0312      0313
C
C 0313      0314      0315      0316      0317      0318      0319      0320      0321      0322
C
C 0322      0323      0324      0325      0326      0327      0328      0329      0330      0331
C
C 0331      0332      0333      0334      0335      0336      0337      0338      0339      0340
C
C 0340      0341      0342      0343      0344      0345      0346      0347      0348      0349
C
C 0349      0350      0351      0352      0353      0354      0355      0356      0357      0358
C
C 0358      0359      0360      0361      0362      0363      0364      0365      0366      0367
C
C 0367      0368      0369      0370      0371      0372      0373      0374      0375      0376
C
C 0376      0377      0378      0379      0380      0381      0382      0383      0384      0385
C
C 0385      0386      0387      0388      0389      0390      0391      0392      0393      0394
C
C 0394      0395      0396      0397      0398      0399      0400      0401      0402      0403
C
C 0403      0404      0405      0406      0407      0408      0409      0410      0411      0412
C
C 0412      0413      0414      0415      0416      0417      0418      0419      0420      0421
C
C 0421      0422      0423      0424      0425      0426      0427      0428      0429      0430
C
C 0430      0431      0432      0433      0434      0435      0436      0437      0438      0439
C
C 0439      0440      0441      0442      0443      0444      0445      0446      0447      0448
C
C 0448      0449      0450      0451      0452      0453      0454      0455      0456      0457
C
C 0457      0458      0459      0460      0461      0462      0463      0464      0465      0466
C
C 0466      0467      0468      0469      0470      0471      0472      0473      0474      0475
C
C 0475      0476      0477      0478      0479      0480      0481      0482      0483      0484
C
C 0484      0485      0486      0487      0488      0489      0490      0491      0492      0493
C
C 0493      0494      0495      0496      0497      0498      0499      0500      0501      0502
C
C 0502      0503      0504      0505      0506      0507      0508      0509      0510      0511
C
C 0511      0512      0513      0514      0515      0516      0517      0518      0519      0520
C
C 0520      0521      0522      0523      0524      0525      0526      0527      0528      0529
C
C 0529      0530      0531      0532      0533      0534      0535      0536      0537      0538
C
C 0538      0539      0540      0541      0542      0543      0544      0545      0546      0547
C
C 0547      0548      0549      0550      0551      0552      0553      0554      0555      0556
C
C 0556      0557      0558      0559      0560      0561      0562      0563      0564      0565
C
C 0565      0566      0567      0568      0569      0570      0571      0572      0573      0574
C
C 0574      0575      0576      0577      0578      0579      0580      0581      0582      0583
C
C 0583      0584      0585      0586      0587      0588      0589      0590      0591      0592
C
C 0592      0593      0594      0595      0596      0597      0598      0599      0600      0601
C
C 0601      0602      0603      0604      0605      0606      0607      0608      0609      0610
C
C 0610      0611      0612      0613      0614      0615      0616      0617      0618      0619
C
C 0619      0620      0621      0622      0623      0624      0625      0626      0627      0628
C
C 0628      0629      0630      0631      0632      0633      0634      0635      0636      0637
C
C 0637      0638      0639      0640      0641      0642      0643      0644      0645      0646
C
C 0646      0647      0648      0649      0650      0651      0652      0653      0654      0655
C
C 0655      0656      0657      0658      0659      0660      0661      0662      0663      0664
C
C 0664      0665      0666      0667      0668      0669      0670      0671      0672      0673
C
C 0673      0674      0675      0676      0677      0678      0679      0680      0681      0682
C
C 0682      0683      0684      0685      0686      0687      0688      0689      0690      0691
C
C 0691      0692      0693      0694      0695      0696      0697      0698      0699      0700
C
C 0700      0701      0702      0703      0704      0705      0706      0707      0708      0709
C
C 0709      0710      0711      0712      0713      0714      0715      0716      0717      0718
C
C 0718      0719      0720      0721      0722      0723      0724      0725      0726      0727
C
C 0727      0728      0729      0730      0731      0732      0733      0734      0735      0736
C
C 0736      0737      0738      0739      0740      0741      0742      0743      0744      0745
C
C 0745      0746      0747      0748      0749      0750      0751      0752      0753      0754
C
C 0754      0755      0756      0757      0758      0759      0760      0761      0762      0763
C
C 0763      0764      0765      0766      0767      0768      0769      0770      0771      0772
C
C 0772      0773      0774      0775      0776      0777      0778      0779      0780      0781
C
C 0781      0782      0783      0784      0785      0786      0787      0788      0789      0790
C
C 0790      0791      0792      0793      0794      0795      0796      0797      0798      0799
C
C 0799      0800      0801      0802      0803      0804      0805      0806      0807      0808
C
C 0808      0809      0810      0811      0812      0813      0814      0815      0816      0817
C
C 0817      0818      0819      0820      0821      0822      0823      0824      0825      0826
C
C 0826      0827      0828      0829      0830      0831      0832      0833      0834      0835
C
C 0835      0836      0837      0838      0839      0840      0841      0842      0843      0844
C
C 0844      0845      0846      0847      0848      0849      0850      0851      0852      0853
C
C 0853      0854      0855      0856      0857      0858      0859      0860      0861      0862
C
C 0862      0863      0864      0865      0866      0867      0868      0869      0870      0871
C
C 0871      0872      0873      0874      0875      0876      0877      0878      0879      0880
C
C 0880      0881      0882      0883      0884      0885      0886      0887      0888      0889
C
C 0889      0890      0891      0892      0893      0894      0895      0896      0897      0898
C
C 0898      0899      0900      0901      0902      0903      0904      0905      0906      0907
C
C 0907      0908      0909      0910      0911      0912      0913      0914      0915      0916
C
C 0916      0917      0918      0919      0920      0921      0922      0923      0924      0925
C
C 0925      0926      0927      0928      0929      0930      0931      0932      0933      0934
C
C 0934      0935      0936      0937      0938      0939      0940      0941      0942      0943
C
C 0943      0944      0945      0946      0947      0948      0949      0950      0951      0952
C
C 0952      0953      0954      0955      0956      0957      0958      0959      0960      0961
C
C 0961      0962      0963      0964      0965      0966      0967      0968      0969      0970
C
C 0970      0971      0972      0973      0974      0975      0976      0977      0978      0979
C
C 0979      0980      0981      0982      0983      0984      0985      0986      0987      0988
C
C 0988      0989      0990      0991      0992      0993      0994      0995      0996      0997
C
C 0997      0998      0999      09999

```

```

0017      QH=QH*EPS
0018      DO 3 J=1,N
0019          IF(Q(J).LT.QH)Q(J)=0.
0020      CONTINUE
0021      C     IF (I2.EQ.1) WRITE(LLNDUT,104) QL,(Q(I),I=1,N)
0022      C     CALL S1(X,B,M,N)
0023      C     IF(I3.NE.0)WRITE(LLNDUT,100)(X(J),J=1,N)
0024      C     IF(I1.LE.0)RETURN
0025      C     iterative improvement
0026      C     XNORM=0.
0027      C     DO 5 J=1,N
0028          C     XNORM=AMAX1(XNORM,ABS(X(J)))
0029      C     IF(XNORM.EQ.0.)RETURN
0030      C     DO 10 K=1,I1
0031      C     DO 7 I=1,M
0032          C     S=O.
0033      C     DO 6 J=1,N
0034          C     S=S+DBLE(A(I,J))*DBLE(X(J))
0035      C     CONTINUE
0036      C     R(I) = DBLE( B(I) ) - S
0037      C     CONTINUE
0038      C     CALL S1(DX,R,M,N)
0039      C     DXNORM=0.
0040      C     DO 8 J=1,N
0041          C     T=X(J)
0042          C     X(J)=X(J)+DX(J)
0043          C     DXNORM=AMAX1(DXNORM,ABS(X(J)-T))
0044      C     CONTINUE
0045      C     IF (I3.NE.0) WRITE(LLNDUT,100)(X(J),J=1,N)
0046          C     IF ( NOT. ( DXNORM .LE. EPS1*XNDRM ) ) GO TO 11
0047          C     IF (I3.NE.0) WRITE(LLNDUT,102) K
0048      C     RETURN
0049      C     11 CONTINUE
0050      C     10 CONTINUE
0051      C     WRITE(LLNDUT,101)I1
0052      C     RETURN
0053      C     100 FORMAT(1HO,(5E12.3))
0054      C     101 FORMAT(' CONVERGENCE IN ITERATIVE IMPROVEMENT DID NOT OCCUR IN ',1
1

```

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

0055 102 FORMAT(13,' STEPS OF ITERATIVE REFINEMENT WERE TAKEN. ')
0056 103 FORMAT(' ITERATION',13,' L2 NORM OF RESIDUAL = ',E12.3,/.
 1 (5E12.3))
0057 104 FORMAT(1/, ' CONDITION OF MATRIX = ',E12.3,/.
 1 SINGULAR VALUES ',./,(5E12.3))
 C
0058 END
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

L2

PAGE PO03

10-10-83

16:41:56

97.000
98.000
99.000
100.000
101.000
102.000
103.000

```

C
C
C      C SUBROUTINE SVD(A,M,N,IU,IV,M1)
C
C      C COMPUTATION OF THE SINGULAR VALUES AND COMPLETE ORTHOGONAL
C      C DECOMPOSITION OF A RECTANGULAR MATRIX A.
C
C      C      A = U * DIAG(G) * V(TRANSPOSE)
C      C      U(TRANSPOSE) * U = V(TRANSPOSE) * V = I
C
C      C      IF IU=1, THEN U IS COMPUTED.
C      C      IF IV=1, THEN V IS COMPUTED.
C
C      C      dimensions A(M,N),U(M,N),V(N,N),G(N)
C
C      C      REAL A(M1,N)
C      C      COMMON/SUB/U(200,200),V(11,11),Q(11),E(200)
C      C      DATA TOL/1.102E-30/
C
C      C      begin
C      C      EPS=4.768E-07
C
C      C      DO 1 I=1,M
C      C          DO 100 J=1,N
C      C              U(I,J)=A(I,J)
C
C      1001    CONTINUE
C      1      CONTINUE
C
C      C      G=0.
C      C      X=0.
C
C      C      DO 11 I=1,N
C      C          E(I)=G
C      C          S=0.
C      C          L=I+1
C
C      111     DO 2 J=I,M
C      C          S=S+U(J,I)*U(J,I)
C
C      2      CONTINUE
C
C      C      BEGIN HOUSEHOLDER'S REDUCTION TO BIDIAGONAL FORM.
C
C      C      G=0.
C      C      IF(.NOT.(S.GE.TOL))GO TO 5
C      C          F=U(I,I)
C      C          G=SQRT(S)
C      C          IF(F.GE.0.)G=-G
C      C          H=F+G-S
C      C          U(I,I)=F-G
C
C      0017
C      0018
C      0019
C
C      0020
C      0021
C      0022
C      0023
C      0024
C      0025
C      0026

```

```

      0027      IF(.NOT.(L.LE.N))GO TO 1005    10-10-93
      0028      DO 4 J=L,N
      0029          S=0.
      0030          DO 3 K=I,M
      0031              S=S+U(K,I)*U(K,J)
      0032          CONTINUE
      0033          F=S/H
      0034          DO 1004 K=I,M
      0035              U(K,J)=U(K,J)+F*U(K,I)
      0036          CONTINUE
      0037          CONTINUE
      0038          CONTINUE
      0039          C
      0040          Q(I)=G
      0041          G=0.
      0042          IF(.NOT.(L.LE.N))GO TO 10
      0043          S=O.
      0044          DO 6 J=L,N
      0045          S=S+U(I,J)*U(I,J)
      0046          CONTINUE
      0047          IF(.NOT.(S.GE.TOL))GO TO 1010
      0048          F=U(I,I+1)
      0049          G=SQR(S)
      0050          IF(F.GE.O.)G=-G
      0051          H=F*G-S
      0052          U(I,I+1)=F-G
      0053          DO 7 J=L,N
      0054              E(J)=U(I,J)/H
      0055          CONTINUE
      0056          DO 9 J=L,M
      0057              S=O.
      0058              DO 8 K=L,N
      0059                  S=S+U(J,K)*U(I,K)
      0060          CONTINUE
      0061          DO 1009 K=L,N
      0062              U(J,K)=U(J,K)+S+E(K)
      0063          CONTINUE
      0064          CONTINUE
      0065          DO 1009
      0066              10 CONTINUE
      0067              Y=ABS(Q(I))*ABS(E(I))
      0068              IF(Y.GT.X)X=Y
      0069              11 CONTINUE
      C      END HOUSEHOLDER'S REDUCTION
      C      BEGIN COMPUTATION OF VECTOR V
      C      IF(.NOT.(IV.EQ.1))GO TO 18
      0070          DO 17 I=1,N
      0071              I=N+I-I
      0072          IF(.NOT.(L.LE.N))GO TO 165
      0073          IF(.NOT.(G.NE.O.))GO TO 15
      0074
  
```

```

0076      DO 12 J=L,N
0077      V(J,I)-U(I,J)/H
0078      CONTINUE
12      C
          DO 14 J=L,N
          S=0.
          DO 13 K=L,N
          S=S+U(I,K)*V(K,J)
          CONTINUE
13      C
          DO 1014 K=L,N
          V(K,J)=V(K,J)+S*V(K,I)
          CONTINUE
1014    C
          CONTINUE
14      C
          CONTINUE
15      C
          DO 16 J=L,N
          V(I,J)=0.
          V(J,I)=0.
          CONTINUE
16      C
          CONTINUE
          DO 165
          CONTINUE
165     C
          V(I,I)=1.
          G=E(I)
          L=I
          CONTINUE
165     C
          END COMPUTATION OF V
          C
          BEGIN COMPUTATION OF VECTOR U
          C
          IF( .NOT. (IU.EQ.1) )GO TO 26
          DO 25 I=1,N
          I=N+1-I
          L=I+1
          G=Q(I)
          IF( .NOT. (L.LE.N) )GO TO 195
          DO 19 J=L,N
          U(I,J)=0.
          CONTINUE
19      C
          CONTINUE
195    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)
          CONTINUE
          F=S/H
          DO 1021 K=I,M
          U(K,J)=U(K,J)+F*U(K,I)
20      C

```

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

10-10-83

16:41:57

PAGE 004

```

      0120      1021      CONTINUE          249.000
      0121      21       CONTINUE          250.000
      0122      215      CONTINUE          251.000
      C
      DO 22 J=1,M
      U(J,1)=U(J,1)/G
      CONTINUE
      22
      C
      GO TO 1025
      ELSE
      CONTINUE
      23
      C
      DO 24 J=1,M
      U(J,1)=0.
      CONTINUE
      24
      C
      0128      1025      CONTINUE          254.500
      0129      25       U(I,I)=U(I,I)+1.
      0130      26       CONTINUE          255.000
      C
      END COMPUTATION OF U          256.000
      C
      BEGIN DIAGONALIZATION OF THE BIDIAGONAL FORM G USING QR ALGORITHM 257.000
      C
      EPS=EPS*X          258.000
      DO 41 K1=1,N          259.000
      K=N+1-K1
      BEGIN BIG LOOP          260.000
      CONTINUE
      41
      C
      DO 28 L1=1,K          261.000
      L=K+1-L1
      IF(.NOT.(ABS(E(L)).GT.EPS))GO TO 32
      IF(.NOT.(ABS(Q(L-1)).GT.EPS))GO TO 29
      CONTINUE
      28
      C
      0139      29       CONTINUE          262.000
      0140      27       C=O.
      0141      28       S=1.
      0142      29       L1=L-1
      0143      28       DO 31 I=L,K
      0144      29       F=S+E(I)
      0145      28       E(I)=C*E(I)
      0146      29       IF(ABS(F).LE.EPS)GO TO 32
      0147      28       G=Q(I)
      0148      29       H=SQR(F+F+G*G)
      0149      28       Q(I)=H
      0150      29       C=G/H
      0151      28       S=F/H
      0152      29       IF(.NOT.(IU.EQ.1))GO TO 1031
      0153      28       DO 30 J=1,M
      0154      29       Y=IU,I,J
      0155      28       ..
      0156      29       ..
      0157      28       ..
      0158      29       ..
      0159      28       ..
      ..
  
```

```

0161      U(J,L1)=Y*C+Z*S
0162      U(J,1)=-Y*S+Z*C
0163      CONTINUE
0164      CONTINUE
0165      CONTINUE
0166      CONTINUE
0167      C      Z=Q(K)
0168      C      IF(L.EQ.K)GO TO 39
0169      C      ACCELERATION SCHEME FOR CONVERGENCE BY SHIFTING TO
0170      C      DISPLACED EIGENVALUES
0171      C      X=Q(L)
0172      C      Y=Q(K-1)
0173      C      G=E(K)
0174      C      H=E(K)
0175      C      F=((Y-Z)*(Y+Z)+(G-H)*(G+H))/(2.*H*Y)
0176      C      G=SQR(F+F+1.)
0177      C      IF(.NOT.(F.LT.0.))GO TO 33
0178      C      F=((X-Z)*(X+Z)+H*(Y/(F-G)-H))/X
0179      C      GO TO 34
0180      C      CONTINUE
0181      C      F=((X-Z)*(X+Z)+H*(Y/(F+G)-H))/X
0182      C      CONTINUE
0183      C      C=1.
0184      C      S=1.
0185      C      L2=L+1
0186      C      DO 38 I=L2,K
0187      C      G=E(I)
0188      C      Y=Q(I)
0189      C      H=S*G
0190      C      G=C*G
0191      C      Z=SQR(F+F+H*H)
0192      C      E(I-1)=Z
0193      C      C=F/Z
0194      C      S=H/Z
0195      C      F=X*C+G*S
0196      C      G=-X*S+G*C
0197      C      H=Y*S
0198      C      Y=Y*C
0199      C      IF(.NOT.(IV.EQ.1))GO TO 36
0200      C      DO 35 J=1,N
0201      C      X=V(J,I-1)
0202      C      Z=V(J,I)
0203      C      V(J,I-1)=X*C+Z*S
0204      C      V(J,I)=-X*S+Z*C
0205      C      CONTINUE
0206      C      Z=SQR(F+F+H*H)
0207      C      Q(I-1)=Z
0208      C      C=F/Z
0209      C      S=H/Z

```

MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

PAGE PO06

16:41:57

10-10-83

SVD

```

0209      F=C+G+S*Y          344.000
0210      X=-S*G+C*Y       345.000
0211      IF(.NOT.(IU.EQ.1))GO TO 1038 346.000
0212      DO 37 J=1,M        347.000
0213      Y=U(J,I)           348.000
0214      Z=U(J,I-1)         349.000
0215      U(J,I-1)=Y*C+Z*S 350.000
0216      U(J,I)=-Y*S+Z*C 351.000
0217      CONTINUE           352.000
0218      1038               353.000
0219      CONTINUE           354.000
0220      C                  354.500
0221      E(L)=O.            355.000
0222      E(K)=F.            356.000
0223      Q(K)=X             357.000
0224      GO TO 27            358.000
0225      C                  358..500
0226      END BIG LOOP        359.000
0227      C                  359.500
0228      C                  360.000
0229      C                  360.500
0230      C                  361.000
0231      C                  361.500
0232      C                  362.000
0233      END                 363.000
*OPTIONS IN EFFECT* ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP
*OPTIONS IN EFFECT* NAME = SVD   LINECNT = 57
*STATISTICS* SOURCE STATEMENTS = 233,PROGRAM SIZE =
*STATISTICS* NO DIAGNOSTICS GENERATED
      5466

```


MICHIGAN TERMINAL SYSTEM FORTRAN G(21.8)

	MAIN	10-10-83	16:41:59	PAGE P001
C			413.000	
C-			414.000	
C			415.000	
C	SUBROUTINE CLRtbl(MATRIX,NROW,NCOL)		416.000	
C			417.000	
C			418.000	
C			418.500	
C			419.000	
C			420.000	
C	THIS SUBROUTINE CLEARS A 2-D TABLE (CALLED 'MATRIX') OF		421.000	
C	DIMENSION NROW X NCOL.		422.000	
C			423.000	
C	REAL MATRIX(NROW,NCOL)		424.000	
C			425.000	
C	begin		426.000	
C			426.500	
C	IF (.NOT.((NROW.LT.1).OR.(NCOL.LT.1)))GO TO 1001		427.000	
0003			428.000	
0004	STOP		428.500	
C			429.000	
0005	1001 CONTINUE		430.000	
0006	DO 1002 J=1,NCOL		431.000	
0007	DO 1003 I=1,NROW		432.000	
0008	MATRIX(I,J)=0.		433.000	
0009	1003 CONTINUE		434.000	
0010	1002 CONTINUE		434.500	
C			435.000	
0011	RETURN		436.000	
C			437.000	
0012	END			
*	OPTIONS IN EFFECT* ID,EBCDIC,SOURCE,NOLIST,NODECK,LOAD,NOMAP			
*	OPTIONS IN EFFECT* NAME = CLRtbl , LINECNT = 67			
*	STATISTICS* SOURCE STATEMENTS = 12,PROGRAM SIZE = 544			
*	STATISTICS* NO DIAGNOSTICS GENERATED			

```

C   438.000
C   439.000
C   440.000
C   442.000
C   443.000
C   447.000
C   448.000
C   448.500
C   449.000
C   450.000
C   451.000
C   452.000
C   453.000
C   454.000
C   455.000
C   456.000
C   456.500
C   457.000
C   458.000
C   459.000
C   459.500
C   460.000
C   461.000
C   462.000
C   462.500
C   463.000
C   464.000
C   465.000
C
C   SUBROUTINE CLRVC(VECTOR,N)
C
C   THIS SUBROUTINE CLEARS AN ARRAY (CALLED 'VECTOR') OF
C   DIMENSION N.
C
C   REAL VECTOR(N)
C
C   begin
C
C   IF( .NOT. (N.LT.1) )GO TO 1001
C   STOP
C   1001 CONTINUE
C
C   DO 1002 I=1,N
C       VECTOR(I)=0.
C   1002 CONTINUE
C
C   RETURN
C
C   END
C
C   *OPTIONS IN EFFECT* ID,EBCDIC,SOURCE,NOLIST,NOECK,LOAD,NOMAP
C   *OPTIONS IN EFFECT* NAME = CLRVC    LINECNT = 57
C   *STATISTICS* SOURCE STATEMENTS = 10,PROGRAM SIZE = 430
C   *STATISTICS* NO DIAGNOSTICS GENERATED

```

NO STATEMENTS FLAGGED IN THE ABOVE COMPILENTS.

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

1.000
2.000
3.000
4.000
5.000
6.000
11.000
12.000
13.000
14.000
15.000
16.000
17.000
18.000
19.000
20.000
21.000
22.000
23.000
24.000
25.000
26.000
27.000
28.000
29.000
30.000
31.000
32.000
33.000
34.000
35.000
36.000
37.000
38.000
39.000
39.200
39.400
39.600
39.640
39.680
39.720
40.000
41.000
42.000
43.000
44.000
45.000
46.000
46.500
47.000
48.000
49.000
50.000

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_dov : REAL ;
      BEGIN
      { Open files }
      RESET ( Pha_f, 'UNIT=2' );
      RESET ( Eye_f, 'UNIT=5' );
      { Initialize }
      FOR U := 1 TO u_max DO
      FOR V := 1 TO v_max DO
      Data (U, V, 1) := Xs;
      Data (U, V, 2) := Ys;
      Data (U, V, 3) := X1;
      Data (U, V, 4) := Y1;
      Data (U, V, 5) := Z1
      END;
      WHILE NOT EOF ( Pha_f ) DO
      BEGIN
      READLN ( Pha_f, U, V, Xs, Ys, X1, Y1, Z1 );
      Data (U, V, 1) := Xs;
      Data (U, V, 2) := Ys;
      Data (U, V, 3) := X1;
      Data (U, V, 4) := Y1;
      Data (U, V, 5) := Z1
      END;

```

<MAIN>

PHAN_CAL

STMT

SOURCE PROGRAM

```
V-----1-----2-----3-----4-----5-----6-----7-----8-----9-----10-----11-----12-----13-----14-----15-----16-----17-----18-----19-----20-----21-----22-----23-----24-----25-----26-----27-----28-----29-----30-----31-----32-----33-----34-----35-----36-----37-----38-----39-----40-----41-----42-----43
```

51.000 { read calculated points from eye file }

52.000

53.000

54.000 WHILE NOT EOF (Eye_f) DO

55.000 BEGIN

56.000 READLN (Eye_f, U, xs, ys, x1, y1, z1);

56.100 { find v }

56.200 V := 0 ;

56.300 REPEAT V := V + 1 UNTIL (V > v_max) | ((Data(U, V, 1.) = xs) & (Data(.U, V, 2.) = ys)) ;

56.400

56.500

56.600

56.800 IF V <= v_max THEN BEGIN

56.920 Data (.U, V, 6.) := X1 ;

57.000 Data (.U, V, 7.) := Y1 ;

58.000 Data (.U, V, 8.) := Z1 ;

59.000 END;

59.500 ELSE WRITELN ('*** Not Found ***')

59.700 END;

59.800 END;

60.000 { find average and standard deviation of distances between }

61.000 { intersection points from phantom file and calculated points }

62.000 { from eye file }

63.000

64.000

65.000

65.100 WRITELN ('----- intersection -----');

65.300 X1 = Y1 = Z1

65.600 WRITELN ('----- computed point -----');

65.800

66.000 Num_data := 0 ;

67.000 Sum := 0 ;

68.000 Sum_sqr := 0 ;

69.000

70.000 FOR U := 1 TO u_max DO

71.000 FOR V := 1 TO v_max DO

72.000 IF Data (.U, V, 1.) > 0 THEN BEGIN

73.000

74.000

75.000 X1 := Data (.U, V, 3.) ; X2 := Data (.U, V, 6.) ;

76.000 Y1 := Data (.U, V, 4.) ; Y2 := Data (.U, V, 7.) ;

77.000 Z1 := Data (.U, V, 5.) ; Z2 := Data (.U, V, 8.) ;

78.000

79.000 Dist_sqr := SQR (X1 - X2) + SQR (Y1 - Y2) + SQR (Z1 - Z2) ;

80.000 Sum_sqr := Sum_sqr + Dist_sqr ;

81.000

82.000 Dist := SQR (Dist_sqr) ;

83.000 Sum := Sum + Dist ;

84.000 WRITELN (X1 :10:3, Y1 :10:3, Z1 :10:3, X2 :12:3, Y2 :10:3, Z2 :10:3, Dist :12:3) ;

85.000

86.000 Num_data := Num_data + 1

87.000

PASCAL/VIS RELEASE 2.1 PHAN_CAL :<MAIN>
 LINE NUMBER B P C I STMT # SOURCE PROGRAM
 V-----1-----2-----3-----4-----5-----6-----7-----8-----9-----1V

```

88.000 1   1 2
89.000 1   1 2
90.000 1   1 2
91.000 1   1 2
92.000 1   1 2
93.000 1   1 2
93.500 1   1 2
94.000 1   1 2
94.500 1   1 2
95.000 1   1 2
96.000 1   1 2

```

END ;

Mean := Sum / Num_data ; StanDev := SQRT (Sum_sqr / Num_data - Mean * Mean) ;

writeln ; writeln (' --- Average Distance : ', mean : 7 : 3) ;

writeln (' --- Standard Deviation : ', stanDev : 7 : 3)

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

LINE NUMBER B P C I Stmt #

PHAN_CAL

S O U R C E P R O G R A M

```
1.000 1
2.000 2
3.000 3
4.000 4
5.000 5
6.000 6
11.000 7
12.000 8
13.000 9
14.000 10
15.000 11
16.000 12
17.000 13
18.000 14
19.000 15
20.000 16
21.000 17
22.000 18
23.000 19
24.000 20
25.000 21
26.000 22
27.000 23
28.000 24
29.000 25
30.000 26
31.000 27
32.000 28
33.000 29
34.000 30
35.000 31
36.000 32
37.000 33
38.000 34
39.000 35
39.200 36
39.400 37
39.600 38
39.640 39
39.680 40
39.720 41
40.000 42
41.000 43
42.000 44
43.000 45
44.000 46
45.000 47
46.000 48
46.500 49
47.000 50
48.000 51
49.000 52
50.000 53
51.000 54
52.000 55
53.000 56
54.000 57
55.000 58
56.000 59
57.000 60
58.000 61
59.000 62
60.000 63
61.000 64
62.000 65
63.000 66
64.000 67
65.000 68
66.000 69
67.000 70
68.000 71
69.000 72
70.000 73
71.000 74
72.000 75
73.000 76
74.000 77
75.000 78
76.000 79
77.000 80
78.000 81
79.000 82
80.000 83
81.000 84
82.000 85
83.000 86
84.000 87
85.000 88
86.000 89
87.000 90
88.000 91
89.000 92
90.000 93
91.000 94
92.000 95
93.000 96
94.000 97
95.000 98
96.000 99
97.000 100
98.000 101
99.000 102
100.000 103
101.000 104
102.000 105
103.000 106
104.000 107
105.000 108
106.000 109
107.000 110
108.000 111
109.000 112
110.000 113
111.000 114
112.000 115
113.000 116
114.000 117
115.000 118
116.000 119
117.000 120
118.000 121
119.000 122
120.000 123
121.000 124
122.000 125
123.000 126
124.000 127
125.000 128
126.000 129
127.000 130
128.000 131
129.000 132
130.000 133
131.000 134
132.000 135
133.000 136
134.000 137
135.000 138
136.000 139
137.000 140
138.000 141
139.000 142
140.000 143
141.000 144
142.000 145
143.000 146
144.000 147
145.000 148
146.000 149
146.500 150
147.000 151
148.000 152
149.000 153
150.000 154
151.000 155
152.000 156
153.000 157
154.000 158
155.000 159
156.000 160
157.000 161
158.000 162
159.000 163
160.000 164
161.000 165
162.000 166
163.000 167
164.000 168
165.000 169
166.000 170
167.000 171
168.000 172
169.000 173
170.000 174
171.000 175
172.000 176
173.000 177
174.000 178
175.000 179
176.000 180
177.000 181
178.000 182
179.000 183
180.000 184
181.000 185
182.000 186
183.000 187
184.000 188
185.000 189
186.000 190
187.000 191
188.000 192
189.000 193
190.000 194
191.000 195
192.000 196
193.000 197
194.000 198
195.000 199
196.000 200
197.000 201
198.000 202
199.000 203
200.000 204
201.000 205
202.000 206
203.000 207
204.000 208
205.000 209
206.000 210
207.000 211
208.000 212
209.000 213
210.000 214
211.000 215
212.000 216
213.000 217
214.000 218
215.000 219
216.000 220
217.000 221
218.000 222
219.000 223
220.000 224
221.000 225
222.000 226
223.000 227
224.000 228
225.000 229
226.000 230
227.000 231
228.000 232
229.000 233
230.000 234
231.000 235
232.000 236
233.000 237
234.000 238
235.000 239
236.000 240
237.000 241
238.000 242
239.000 243
240.000 244
241.000 245
242.000 246
243.000 247
244.000 248
245.000 249
246.000 250
247.000 251
248.000 252
249.000 253
250.000 254
251.000 255
252.000 256
253.000 257
254.000 258
255.000 259
256.000 260
257.000 261
258.000 262
259.000 263
260.000 264
261.000 265
262.000 266
263.000 267
264.000 268
265.000 269
266.000 270
267.000 271
268.000 272
269.000 273
270.000 274
271.000 275
272.000 276
273.000 277
274.000 278
275.000 279
276.000 280
277.000 281
278.000 282
279.000 283
280.000 284
281.000 285
282.000 286
283.000 287
284.000 288
285.000 289
286.000 290
287.000 291
288.000 292
289.000 293
290.000 294
291.000 295
292.000 296
293.000 297
294.000 298
295.000 299
296.000 300
297.000 301
298.000 302
299.000 303
300.000 304
301.000 305
302.000 306
303.000 307
304.000 308
305.000 309
306.000 310
307.000 311
308.000 312
309.000 313
310.000 314
311.000 315
312.000 316
313.000 317
314.000 318
315.000 319
316.000 320
317.000 321
318.000 322
319.000 323
320.000 324
321.000 325
322.000 326
323.000 327
324.000 328
325.000 329
326.000 330
327.000 331
328.000 332
329.000 333
330.000 334
331.000 335
332.000 336
333.000 337
334.000 338
335.000 339
336.000 340
337.000 341
338.000 342
339.000 343
340.000 344
341.000 345
342.000 346
343.000 347
344.000 348
345.000 349
346.000 350
347.000 351
348.000 352
349.000 353
350.000 354
351.000 355
352.000 356
353.000 357
354.000 358
355.000 359
356.000 360
357.000 361
358.000 362
359.000 363
360.000 364
361.000 365
362.000 366
363.000 367
364.000 368
365.000 369
366.000 370
367.000 371
368.000 372
369.000 373
370.000 374
371.000 375
372.000 376
373.000 377
374.000 378
375.000 379
376.000 380
377.000 381
378.000 382
379.000 383
380.000 384
381.000 385
382.000 386
383.000 387
384.000 388
385.000 389
386.000 390
387.000 391
388.000 392
389.000 393
390.000 394
391.000 395
392.000 396
393.000 397
394.000 398
395.000 399
396.000 400
397.000 401
398.000 402
399.000 403
400.000 404
401.000 405
402.000 406
403.000 407
404.000 408
405.000 409
406.000 410
407.000 411
408.000 412
409.000 413
410.000 414
411.000 415
412.000 416
413.000 417
414.000 418
415.000 419
416.000 420
417.000 421
418.000 422
419.000 423
420.000 424
421.000 425
422.000 426
423.000 427
424.000 428
425.000 429
426.000 430
427.000 431
428.000 432
429.000 433
430.000 434
431.000 435
432.000 436
433.000 437
434.000 438
435.000 439
436.000 440
437.000 441
438.000 442
439.000 443
440.000 444
441.000 445
442.000 446
443.000 447
444.000 448
445.000 449
446.000 450
447.000 451
448.000 452
449.000 453
450.000 454
451.000 455
452.000 456
453.000 457
454.000 458
455.000 459
456.000 460
457.000 461
458.000 462
459.000 463
460.000 464
461.000 465
462.000 466
463.000 467
464.000 468
465.000 469
466.000 470
467.000 471
468.000 472
469.000 473
470.000 474
471.000 475
472.000 476
473.000 477
474.000 478
475.000 479
476.000 480
477.000 481
478.000 482
479.000 483
480.000 484
481.000 485
482.000 486
483.000 487
484.000 488
485.000 489
486.000 490
487.000 491
488.000 492
489.000 493
490.000 494
491.000 495
492.000 496
493.000 497
494.000 498
495.000 499
496.000 500
497.000 501
498.000 502
499.000 503
500.000 504
501.000 505
502.000 506
503.000 507
504.000 508
505.000 509
506.000 510
507.000 511
508.000 512
509.000 513
510.000 514
511.000 515
512.000 516
513.000 517
514.000 518
515.000 519
516.000 520
517.000 521
518.000 522
519.000 523
520.000 524
521.000 525
522.000 526
523.000 527
524.000 528
525.000 529
526.000 530
527.000 531
528.000 532
529.000 533
530.000 534
531.000 535
532.000 536
533.000 537
534.000 538
535.000 539
536.000 540
537.000 541
538.000 542
539.000 543
540.000 544
541.000 545
542.000 546
543.000 547
544.000 548
545.000 549
546.000 550
547.000 551
548.000 552
549.000 553
550.000 554
551.000 555
552.000 556
553.000 557
554.000 558
555.000 559
556.000 560
557.000 561
558.000 562
559.000 563
560.000 564
561.000 565
562.000 566
563.000 567
564.000 568
565.000 569
566.000 570
567.000 571
568.000 572
569.000 573
570.000 574
571.000 575
572.000 576
573.000 577
574.000 578
575.000 579
576.000 580
577.000 581
578.000 582
579.000 583
580.000 584
581.000 585
582.000 586
583.000 587
584.000 588
585.000 589
586.000 590
587.000 591
588.000 592
589.000 593
590.000 594
591.000 595
592.000 596
593.000 597
594.000 598
595.000 599
596.000 600
597.000 601
598.000 602
599.000 603
600.000 604
601.000 605
602.000 606
603.000 607
604.000 608
605.000 609
606.000 610
607.000 611
608.000 612
609.000 613
610.000 614
611.000 615
612.000 616
613.000 617
614.000 618
615.000 619
616.000 620
617.000 621
618.000 622
619.000 623
620.000 624
621.000 625
622.000 626
623.000 627
624.000 628
625.000 629
626.000 630
627.000 631
628.000 632
629.000 633
630.000 634
631.000 635
632.000 636
633.000 637
634.000 638
635.000 639
636.000 640
637.000 641
638.000 642
639.000 643
640.000 644
641.000 645
642.000 646
643.000 647
644.000 648
645.000 649
646.000 650
647.000 651
648.000 652
649.000 653
650.000 654
651.000 655
652.000 656
653.000 657
654.000 658
655.000 659
656.000 660
657.000 661
658.000 662
659.000 663
660.000 664
661.000 665
662.000 666
663.000 667
664.000 668
665.000 669
666.000 670
667.000 671
668.000 672
669.000 673
670.000 674
671.000 675
672.000 676
673.000 677
674.000 678
675.000 679
676.000 680
677.000 681
678.000 682
679.000 683
680.000 684
681.000 685
682.000 686
683.000 687
684.000 688
685.000 689
686.000 690
687.000 691
688.000 692
689.000 693
690.000 694
691.000 695
692.000 696
693.000 697
694.000 698
695.000 699
696.000 700
697.000 701
698.000 702
699.000 703
700.000 704
701.000 705
702.000 706
703.000 707
704.000 708
705.000 709
706.000 710
707.000 711
708.000 712
709.000 713
710.000 714
711.000 715
712.000 716
713.000 717
714.000 718
715.000 719
716.000 720
717.000 721
718.000 722
719.000 723
720.000 724
721.000 725
722.000 726
723.000 727
724.000 728
725.000 729
726.000 730
727.000 731
728.000 732
729.000 733
730.000 734
731.000 735
732.000 736
733.000 737
734.000 738
735.000 739
736.000 740
737.000 741
738.000 742
739.000 743
740.000 744
741.000 745
742.000 746
743.000 747
744.000 748
745.000 749
746.000 750
747.000 751
748.000 752
749.000 753
750.000 754
751.000 755
752.000 756
753.000 757
754.000 758
755.000 759
756.000 760
757.000 761
758.000 762
759.000 763
760.000 764
761.000 765
762.000 766
763.000 767
764.000 768
765.000 769
766.000 770
767.000 771
768.000 772
769.000 773
770.000 774
771.000 775
772.000 776
773.000 777
774.000 778
775.000 779
776.000 780
777.000 781
778.000 782
779.000 783
780.000 784
781.000 785
782.000 786
783.000 787
784.000 788
785.000 789
786.000 790
787.000 791
788.000 792
789.000 793
790.000 794
791.000 795
792.000 796
793.000 797
794.000 798
795.000 799
796.000 800
797.000 801
798.000 802
799.000 803
800.000 804
801.000 805
802.000 806
803.000 807
804.000 808
805.000 809
806.000 810
807.000 811
808.000 812
809.000 813
810.000 814
811.000 815
812.000 816
813.000 817
814.000 818
815.000 819
816.000 820
817.000 821
818.000 822
819.000 823
820.000 824
821.000 825
822.000 826
823.000 827
824.000 828
825.000 829
826.000 830
827.000 831
828.000 832
829.000 833
830.000 834
831.000 835
832.000 836
833.000 837
834.000 838
835.000 839
836.000 840
837.000 841
838.000 842
839.000 843
840.000 844
841.000 845
842.000 846
843.000 847
844.000 848
845.000 849
846.000 850
847.000 851
848.000 852
849.000 853
850.000 854
851.000 855
852.000 856
853.000 857
854.000 858
855.000 859
856.000 860
857.000 861
858.000 862
859.000 863
860.000 864
861.000 865
862.000 866
863.000 867
864.000 868
865.000 869
866.000 870
867.000 871
868.000 872
869.000 873
870.000 874
871.000 875
872.000 876
873.000 877
874.000 878
875.000 879
876.000 880
877.000 881
878.000 882
879.000 883
880.000 884
881.000 885
882.000 886
883.000 887
884.000 888
885.000 889
886.000 890
887.000 891
888.000 892
889.000 893
890.000 894
891.000 895
892.000 896
893.000 897
894.000 898
895.000 899
896.000 900
897.000 901
898.000 902
899.000 903
900.000 904
901.000 905
902.000 906
903.000 907
904.000 908
905.000 909
906.000 910
907.000 911
908.000 912
909.000 913
910.000 914
911.000 915
912.000 916
913.000 917
914.000 918
915.000 919
916.000 920
917.000 921
918.000 922
919.000 923
920.000 924
921.000 925
922.000 926
923.000 927
924.000 92
```

```

      :<MAIN>          SOURCE PROGRAM
LINE NUMBER B P C I   STMT #          V-----1-----2-----3-----4-----5-----6-----7-----8-----9-----+-----+
51.000
52.000
53.000
54.000 WHILE NOT EOF ( Eye_f ) DO
55.000 BEGIN
56.000   READLN ( Eye_f, U, xs, ys, x1, y1, z1 ) ;
56.100
56.200   { find v }

56.300   1
56.400   1
56.500   2
56.600   1
56.800   1
56.860   1
56.920   1
57.000   2
58.000   2
59.000   2
59.500   1
59.700   1
59.800   1
60.000
61.000
62.000
63.000
64.000
65.000
65.100
65.300
65.600
65.800
66.000
67.000
68.000
69.000
70.000
71.000
72.000
73.000
74.000
75.000
76.000
77.000
78.000
79.000
80.000
81.000
82.000
83.000
84.000
85.000
86.000
87.000
{ read calculated points from eye file }

13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43

{ read calculated points from eye file }

{ find average and standard deviation of distances between
{ intersection points from phantom file and calculated points
{ from eye file

FOR U := 1 TO v_max DO
  FOR V := 1 TO v_max DO
    IF Data( .U, V, 1.) = xs & ( Data( .U, V, 2.) = ys ) THEN
      BEGIN
        Sum := 0;
        Sum_sqr := 0;
        FOR U := 1 TO v_max DO
          FOR V := 1 TO v_max DO
            IF Data( .U, V, 1.) > 0 THEN
              BEGIN
                X1 := Data( .U, V, 3. );
                Y1 := Data( .U, V, 4. );
                Z1 := Data( .U, V, 5. );
                X2 := Data( .U, V, 6. );
                Y2 := Data( .U, V, 7. );
                Z2 := Data( .U, V, 8. );
                Dist_sqr := SQR( X1 - X2 ) + SQR( Y1 - Y2 ) + SQR( Z1 - Z2 );
                Sum_sqr := Sum_sqr + Dist_sqr;
              END;
            Dist := SQR( Dist_sqr );
            Sum := Sum + Dist;
          END;
        WRITELN( X1 :10:3, Y1 :10:3, Z1 :10:3, X2 :12:3, Y2 :10:3, Z2 :10:3, Dist :12:3 );
      END;
    END;
  END;
END;

```

LINE NUMBER	B	P	C	I	STMT #	PHAN_CAL :<MAIN>	SOURCE PROGRAM
88.000	1	1	2				
89.000		1	2				
90.000		1	2				
91.000					44	Mean := Sum / Num_data ;	
92.000					45	StanDev := SQRT (Sum_sqr / Num_data - Mean * Mean) ;	
93.000						END :	
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						END.	
96.000							

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

```

# sn exec.e
#   $ .00,   $13.971
#amrty -5
#Done.
#   $.00,   $13.971
#$run lamp:eye,obj+1amp:sup,obj+2+naas:ims1 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,   $14.121

```


2	4	12	4	12	4	12	4	12	4	12	4	12	4	12	4	12	4	12	
3	5	1	Geomet	1	Geomet	1	Geomet	1	Geomet	1	Geomet	1	Geomet	1	Geomet	1	Geomet	1	Geomet
4	6	6	1	Imat	6	2	Geomet	6	4	Geomet	6	2	Imat	6	4	Geomet	6	5	Geomet
5	7	6	2	Geomet	7	7	Geomet	7	5	Geomet	7	3	Geomet	7	5	Geomet	7	6	Geomet
6	8	6	3	Imat	8	6	4	Geomet	8	6	5	Geomet	8	6	7	Geomet	8	7	Geomet
7	9	6	6	not observed	9	6	not observed	9	6	not observed	9	6	not observed	9	6	not observed	9	6	not observed
8	10	6	7	GEO	10	6	TMAT	7	7	TMAT	7	7	TMAT	7	7	TMAT	7	7	TMAT
9	11	6	8	not observed	11	6	not observed	8	6	not observed	8	6	not observed	8	6	not observed	8	6	not observed
10	12	6	9	GEO	12	6	TMAT	9	7	TMAT	9	7	TMAT	9	7	TMAT	9	7	TMAT
11	13	6	10	not observed	13	6	not observed	10	6	not observed	10	6	not observed	10	6	not observed	10	6	not observed
12	14	6	11	GEO	14	6	TMAT	11	7	TMAT	11	7	TMAT	11	7	TMAT	11	7	TMAT
13	15	6	12	not observed	15	6	not observed	12	6	not observed	12	6	not observed	12	6	not observed	12	6	not observed
14	16	6	13	GEO	16	6	TMAT	13	7	TMAT	13	7	TMAT	13	7	TMAT	13	7	TMAT
15	17	6	14	not observed	17	6	not observed	14	6	not observed	14	6	not observed	14	6	not observed	14	6	not observed
16	18	6	15	GEO	18	6	TMAT	15	7	TMAT	15	7	TMAT	15	7	TMAT	15	7	TMAT
17	19	6	16	not observed	19	6	not observed	16	6	not observed	16	6	not observed	16	6	not observed	16	6	not observed
18	20	6	17	GEO	20	6	TMAT	17	7	TMAT	17	7	TMAT	17	7	TMAT	17	7	TMAT
19	21	6	18	not observed	21	6	not observed	18	6	not observed	18	6	not observed	18	6	not observed	18	6	not observed
20	22	6	19	GEO	22	6	TMAT	19	7	TMAT	19	7	TMAT	19	7	TMAT	19	7	TMAT
21	23	6	20	not observed	23	6	not observed	20	6	not observed	20	6	not observed	20	6	not observed	20	6	not observed
22	24	6	21	GEO	24	6	TMAT	21	7	TMAT	21	7	TMAT	21	7	TMAT	21	7	TMAT
23	25	6	22	not observed	25	6	not observed	22	6	not observed	22	6	not observed	22	6	not observed	22	6	not observed
24	26	6	23	GEO	26	6	TMAT	23	7	TMAT	23	7	TMAT	23	7	TMAT	23	7	TMAT
25	27	6	24	not observed	27	6	not observed	24	6	not observed	24	6	not observed	24	6	not observed	24	6	not observed
26	28	6	25	GEO	28	6	TMAT	25	7	TMAT	25	7	TMAT	25	7	TMAT	25	7	TMAT
27	29	6	26	not observed	29	6	not observed	26	6	not observed	26	6	not observed	26	6	not observed	26	6	not observed
28	30	6	27	GEO	30	6	TMAT	27	7	TMAT	27	7	TMAT	27	7	TMAT	27	7	TMAT
29	31	6	28	not observed	31	6	not observed	28	6	not observed	28	6	not observed	28	6	not observed	28	6	not observed
30	32	6	29	GEO	32	6	TMAT	29	7	TMAT	29	7	TMAT	29	7	TMAT	29	7	TMAT
31	33	6	30	not observed	33	6	not observed	30	6	not observed	30	6	not observed	30	6	not observed	30	6	not observed
32	34	6	31	GEO	34	6	TMAT	31	7	TMAT	31	7	TMAT	31	7	TMAT	31	7	TMAT
33	35	6	32	not observed	35	6	not observed	32	6	not observed	32	6	not observed	32	6	not observed	32	6	not observed
34	36	6	33	GEO	36	6	TMAT	33	7	TMAT	33	7	TMAT	33	7	TMAT	33	7	TMAT
35	37	6	34	not observed	37	6	not observed	34	6	not observed	34	6	not observed	34	6	not observed	34	6	not observed
36	38	6	35	GEO	38	6	TMAT	35	7	TMAT	35	7	TMAT	35	7	TMAT	35	7	TMAT
37	39	6	36	not observed	39	6	not observed	36	6	not observed	36	6	not observed	36	6	not observed	36	6	not observed
38	40	6	37	GEO	40	6	TMAT	37	7	TMAT	37	7	TMAT	37	7	TMAT	37	7	TMAT
39	41	6	38	not observed	41	6	not observed	38	6	not observed	38	6	not observed	38	6	not observed	38	6	not observed
40	42	6	39	GEO	42	6	TMAT	39	7	TMAT	39	7	TMAT	39	7	TMAT	39	7	TMAT
41	43	6	40	not observed	43	6	not observed	40	6	not observed	40	6	not observed	40	6	not observed	40	6	not observed
42	44	6	41	GEO	44	6	TMAT	41	7	TMAT	41	7	TMAT	41	7	TMAT	41	7	TMAT
43	45	6	42	not observed	45	6	not observed	42	6	not observed	42	6	not observed	42	6	not observed	42	6	not observed
44	46	6	43	GEO	46	6	TMAT	43	7	TMAT	43	7	TMAT	43	7	TMAT	43	7	TMAT
45	47	6	44	not observed	47	6	not observed	44	6	not observed	44	6	not observed	44	6	not observed	44	6	not observed
46	48	6	45	GEO	48	6	TMAT	45	7	TMAT	45	7	TMAT	45	7	TMAT	45	7	TMAT
47	49	6	46	not observed	49	6	not observed	46	6	not observed	46	6	not observed	46	6	not observed	46	6	not observed
48	50	6	47	GEO	50	6	TMAT	47	7	TMAT	47	7	TMAT	47	7	TMAT	47	7	TMAT
49	51	6	48	not observed	51	6	not observed	48	6	not observed	48	6	not observed	48	6	not observed	48	6	not observed
50	52	6	49	GEO	52	6	TMAT	49	7	TMAT	49	7	TMAT	49	7	TMAT	49	7	TMAT
51	53	6	50	not observed	53	6	not observed	50	6	not observed	50	6	not observed	50	6	not observed	50	6	not observed
52	54	6	51	GEO	54	6	TMAT	51	7	TMAT	51	7	TMAT	51	7	TMAT	51	7	TMAT
53	55	6	52	not observed	55	6	not observed	52	6	not observed	52	6	not observed	52	6	not observed	52	6	not observed
54	56	6	53	GEO	56	6	TMAT	53	7	TMAT	53	7	TMAT	53	7	TMAT	53	7	TMAT
55	57	6	54	not observed	57	6	not observed	54	6	not observed	54	6	not observed	54	6	not observed	54	6	not observed
56	58	6	55	GEO	58	6	TMAT	55	7	TMAT	55	7	TMAT	55	7	TMAT	55	7	TMAT
57	59	6	56	not observed	59	6	not observed	56	6	not observed	56	6	not observed	56	6	not observed	56	6	not observed
58	60	6	57	GEO	60	6	TMAT	57	7	TMAT	57	7	TMAT	57	7	TMAT	57	7	TMAT
59	61	6	58	not observed	61	6	not observed	58	6	not observed	58	6	not observed	58	6	not observed	58	6	not observed
60	62	6	59	GEO	62	6	TMAT	59	7	TMAT	59	7	TMAT	59	7	TMAT	59	7	TMAT
61	63	6	60	not observed	63	6	not observed	60	6	not observed	60	6	not observed	60	6	not observed	60	6	not observed
62	64	6	61	GEO	64	6	TMAT	61	7	TMAT	61	7	TMAT	61	7	TMAT	61	7	TMAT
63	65	6	62	not observed	65	6	not observed	62	6	not observed	62	6	not observed	62	6	not observed	62	6	not observed
64	66	6	63	GEO	66	6	TMAT	63	7	TMAT	63	7	TMAT	63	7	TMAT	63	7	TMAT
65	67	6	64	not observed	67	6	not observed	64	6	not observed	64	6	not observed	64	6	not observed	64	6	not observed
66	68	6	65	GEO	68	6	TMAT	65	7	TMAT	65	7	TMAT	65	7	TMAT	65	7	TMAT
67	69	6	66	not observed	69	6	not observed	66	6	not observed	66	6	not observed	66	6	not observed	66	6	not observed
68	70	6	67	GEO	70	6	TMAT	67	7	TMAT	67	7	TMAT	67	7	TMAT	67	7	TMAT
69	71	6	68	not observed	71	6	not observed	68	6	not observed	68	6	not observed	68	6	not observed	68	6	not observed
70	72	6	69	GEO	72	6	TMAT	69	7	TMAT	69	7	TMAT	69	7	TMAT	69	7	TMAT
71	73	6	70	not observed	73	6	not observed	70	6	not observed	70	6	not observed	70	6	not observed	70	6	not observed
72	74	6	71	GEO	74	6	TMAT	71	7	TMAT	71	7	TMAT	71	7	TMAT	71	7	TMAT
73	75	6	72	not observed	75	6	not observed	72	6	not observed	72	6	not observed	72	6	not observed	72	6	not observed
74	76	6	73	GEO	76	6	TMAT	73	7	TMAT	73	7	TMAT	73	7	TMAT	73	7	TMAT
75	77	6	74	not observed	77	6	not observed	74	6	not observed	74	6	not observed	74	6	not observed	74	6	not observed
76	78	6	75	GEO	78	6	TMAT	75	7	TMAT	75	7	TMAT	75	7	TMAT	75	7	TMAT
77	79	6	76	not observed	79	6	not observed	76	6	not observed	76	6	not observed	76	6	not observed	76	6	not observed
78	80	6	77	GEO	80	6	TMAT	77	7	TMAT	77	7	TMAT	77	7	TMAT	77	7	TMAT
79	81	6	78	not observed	81	6	not observed	78	6	not observed	78	6	not observed	78	6	not observed	78	6	not observed
80	82	6	79	GEO	82	6	TMAT	79	7	TMAT	79	7	TMAT	79	7	TMAT	79	7	TMAT
81	83	6	80	not observed	83	6	not observed	80	6	not observed	80	6	not observed	80	6	not observed	80	6	not observed
82	84	6	81</td																

length of the side of a cube -- Length

8 8.0000

Step in u and v -- Step

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

----- T matrix -----

-34.8575208 -15.1447174

3.6250092 -26.2314068

-8.2500031 19.4638063

330.0000000 330.0000000

0.0 0.0

0.0 0.0

0.0 0.0

----- L matrix -----

-6.7556749 -7.6546563

5.3372140 -4.419468

-3.1819810 2.4748723

90.0000000 90.0000000

VMAT *** not observed ***

GEO *** not observed ***

TMAT *** not observed ***

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 14

```

$Log Output: LAMP, 01:32:11 Thu Aug 04/83
#50 exec.f
# $ .00, $13.79f
#empty -2
#Done.
# $ .00, $13.79f
#empty -3
#Done.
# $ .00, $13.80f
#empty -4
#Done.
# $ .00, $13.80f
#r lamp.phan.obj+paschalvslib 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

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

University of Michigan Terminal System (Model UL-223)

卷之三

The image shows a decorative border composed of a repeating pattern of three symbols: 'S', 'M', and 'T'. The 'S' symbol is a simple 'S' shape. The 'M' symbol is a stylized 'M' shape. The 'T' symbol is a stylized 'T' shape. These symbols are arranged in a grid-like pattern, creating a decorative border around the central area.

```

#50 exec a
" $ 01 ; $14.13T
" lamp analy.obj+pasca\valib 2=-2 5=-5 t=3
#Execution begins
----- intersection -----    computed point -----    distance
x1      y1      z1          x2      y2      z2

```

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	-5.117	-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	-1.641	-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	0.618

```
-0.670    8.723    -1.000    -0.970    8.707    -1.171    0.316
-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, $14.17 f
:mts
# $ .13, $14.32 f
#copy -log2 'print*
```