1991 NASA/ASEE SUMMER FACULTY FELLOWSHIP PROGRAM

JOHN F. KENNEDY SPACE CENTER
UNIVERSITY OF CENTRAL FLORIDA

FRAME SHIFT/WARP COMPENSATION FOR THE ARID ROBOT SYSTEM

PREPARED BY:	Dr. Carl D. Latino
ACADEMIC RANK:	Associate Professor
UNIVERSITY AND DEPARTMENT:	Oklahoma State University Electrical & Computer Engineering
NASA/KSC	
DIVISION:	Mechanical Engineering
BRANCH:	Special Projects (ARID)
NASA COLLEAGUE:	Mr. V. Leon Davis
DATE:	August 9, 1991
CONTRACT NUMBER:	University of Central Florida NASA-NGT-60002 Supplement: 6

ACKNOWLEDGEMENTS

I would like to sincerely thank V. Leon Davis for the opportunity of working on the Automatic Radiator Inspection Device (ARID) project. I would like to add that his hospitality and friendship went beyond those of a supervisor and will long be remembered. I also want to acknowledge Chau Le, Todd Graham, and many other NASA and Boeing staff who have been very helpful and who helped make my stay here both productive and enjoyable.

Dr. Ramon Hosler and Kari Stiles need special mention for all their hard work which produced many well planned exciting and informative programs. I wish to applaud their genuine interest, efficency, friendliness and professionalism.

ABSTRACT

The inspection of an orbiter radiator panel is a time intensive tedious chore. The Automatic Radiator Inspection Device (ARID) is a system aimed at automating this procedure. The ARID must have the ability to aim a camera accurately at the desired inspection points which is in the order of 13000. The ideal inspection points are known. The panel may, however, be relocated due to inaccurate parking and warpage. A method of determining the mathematical description of a translated as well as warped surface by accurate measurement of only a few points on this surface is developed here.

The method developed uses a linear warp model whose effect is superimposed on the rigid body translation. Due to the angles involved, small angle approximations are possible which greatly reduces the computational complexity. Given an accurate linear warp model, all the desired translation and warp parameters can be obtained by knowledge of the ideal locations of four fiducial points and the corresponding measurements of these points on the actual radiator surface. The method uses three of the fiducials to define a plane and the fourth to define the warp. Given this information, it is possible to determine a transformation that will enable the ARID system to translate any desired inspection point on the ideal surface to its corresponding value on the actual surface.

SUMMARY

The orbiter radiators which are located on the inner surfaces of the payload doors need to be inspected for damage just before and just after each flight. This is a very time consuming, tedious and boring task which makes it an ideal application for a robot system. The Automatic Radiator Inspection Device (ARID) is a robot system being developed by NASA to address this problem. The device is a computer controlled four degree of freedom robot with a camera. It is designed to greatly reduce the inspection time and manpower needed. In addition it will improve the accuracy of data collection and processing.

To be effective, the robot must be capable of accurately positioning the camera relative to desired inspection points on the radiator panel. The system is to operate open loop, so it is essential that it knows precisely the positioning and orientation of the panel being inspected. Although the ideal position is known, the orbiter will not always be parked the same way and the door panel may not be open the same amount. Even though they are mechanically constrained, the doors may be warped. It was therefore essential to develop a method to deal with this problem.

The main objectives of this project were to evaluate methods under consideration and, if necessary, develop one capable of meeting the rigid constraints. To meet these constraints, it seemed necessary to develop an accurate warp model. Without a valid warp model it seems impractical to operate the system open loop. Warp was measured to be in the order of 2 inches over a 15 foot run. The warp needed to be mathematically descrivable. A first order warp approximation was assumed. Assuming that this model is true, it is possible to know the orientation and warpage of any panel by simply measuring four points whose ideal positions are known and their position independently verified (these points are called fiducials).

If a higher order warp modeling is needed, the procedure works, but requires that at least n+1 fiducial points be measured. A positive verification of the order needed for an acceptable model requires that more data be collected and evaluated. The data collected so far seems to indicate that a linear warp model will suffice to meet the requirements. If the warp is actuallu random the system may not be able to be operated in open loop.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

I. INTRODUCTION

The Automatic Radiator Inspection Device (ARID) is a robot system that will be used in the inspection of the orbiter radiator panels for mission damage and for pre flight inspection. The data collected is to be stored so that comparisons between present and prevous data can be made for damage assessment. For inspection purposes each panel is subdivided into a tiling of four inch square sections. The center of these squares defines an inspection point. These points are known in some xyz coordinate system, but there are no visual clues of these on the panel surfaces.

The robot will have within its memory the ideal xyz coordinates of all the desired inspection points but no means of feedback to verify these during normal operation. The ideal position of each inspection point is known to the robot but the actual xyz coordinate of the point on the actual surface must be determined. The robot will move a camera that is to take photographs of the entire radiator surface by taking a series of 4 in. by 4 in. pictures. The center of each of these pictures is an inspection point. The data obtained at these points may then be compared to data obtained from previous inspections. In this way any new damage wether it be from flight or handling can be determined.

In order to make a proper assessment, the position and orientation of each inspection point must be known accurately. If the orbiter is parked with a different attitude and the doors are open different amounts at each inspection accurate comparisons cannot be made without complex image processing capabilities. The difference in parking attitude, door opening angle and warpage in the doors needs to be known precisely so that the robot can locate the intended inspection points accurately in order to position the camera at the proper focal distance.

To this end a method was needed which would provide the necessary information to the ARID system so that it could make the proper adjustments to compensate for inaccurate parking. Rigid body translation, also known as frame shift is a well known way of handling such a problem. The ideal as well as actual positions of points whose coordinates are known and whose position can be separately verified (these will be referred to as the fiducial points)

must be provided to the ARID system. If the panel is rigid, three points whose ideal and their corresponding measured points are known, suffice to accurately determine all six values of rotation and tranlation. It was found, however, that the panels could not be assumed rigid. In an effort to solve this problem a variety of methods and models were being considered.

There was one that was already being developed by Boeing. Part of the task was to evaluate the merits and accuracy of the method under consideration as well as develop a better one if needed. It should be noted that there are only twelve fiducial points on each of the four aft panels and only eight on the four fore panels. Any solution to this problem while maintaining the open loop option could only rely on these and no other points on the panels.

II. THE ARID SYSTEM

2.1 THE RADIATOR PANELS

The orbiter payload bay is enclosed by eight doors arranged in four groups of two. The inner side of these doors have radiator panels used to cool the on board electronics. Each panel is a smooth surface with eight bolts for each of the forward doors and twelve bolts for each of the aft doors (See Figure 2-1.). These are the only clearly distinguishing marks on the otherwise smooth surface of the radiators. These points will be referred to as fiducials.

Each radiator is inspected by the robot in 4 in. by 4 in. squares. This translates into over 1600 inspection points per panel or almost 13000 total points. It is therefore clear that visual feedback would be quite difficult as well as computationally taxing on the system. The doors were designed to operate in space at zero g's, therefore in the one g environment on earth they are supported by a structure called the strongback. These supports make the door more rigid but may at the same time introduce some unexpected warp.

The radiator panels have some complex curves to conform with the orbiter shape. The two forward panels have the most difficult curves for ARID to negotiate, but even the aft six doors require the use of a multidegree of freedom robot for proper camera placement and orientation. The panels are hinged to the orbiter and at this edge are quite straight. At other points a noticeable warp was detected. For a 15 foot door panel, with three corners properly located, the fourth was found to differ in placement by as much as 2 inches.

Restrictions on focal distance of the camera state that the distance from the panel surface must be 24 inches with a plus or minus 0.125 inches. Although a 2 inch warp on a 15 foot run may be considered small, it is sixteen times too large for the ARID system to tolerate.

2.2 THE ARID ROBOT

The ARID robot, see Figure 2-2, is a four degree of freedom device. It has one prismatic and three revolute joints. this robot will be mounted in the Orbiter Processing Facility (OPF) on a track parallel to the longitudinal axis of the orbiter. Ideally, the robot will use its revolute joints to position and aim the camera at the panel,

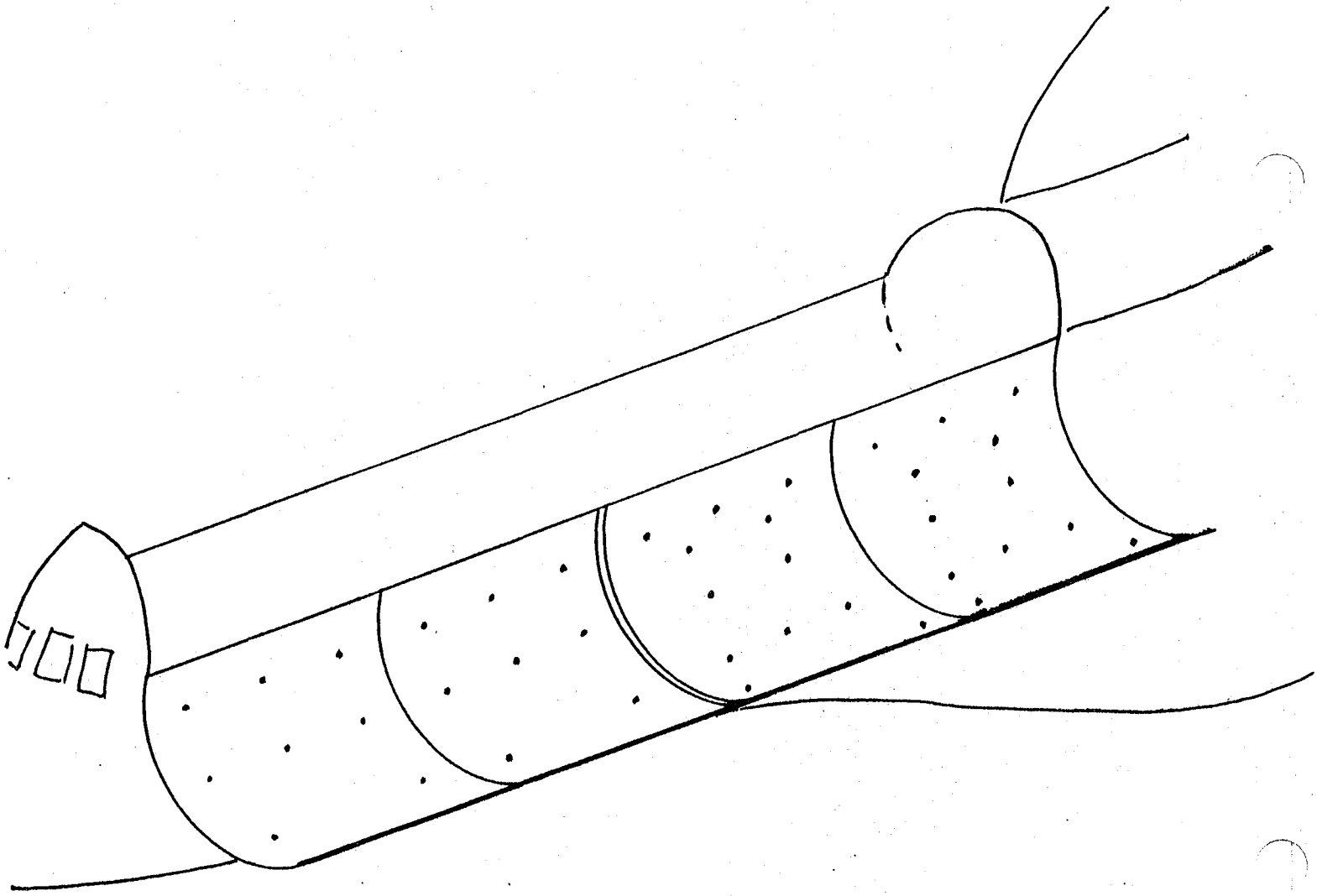

Figure 2-1 Fiducial Points on the Radiator Panels

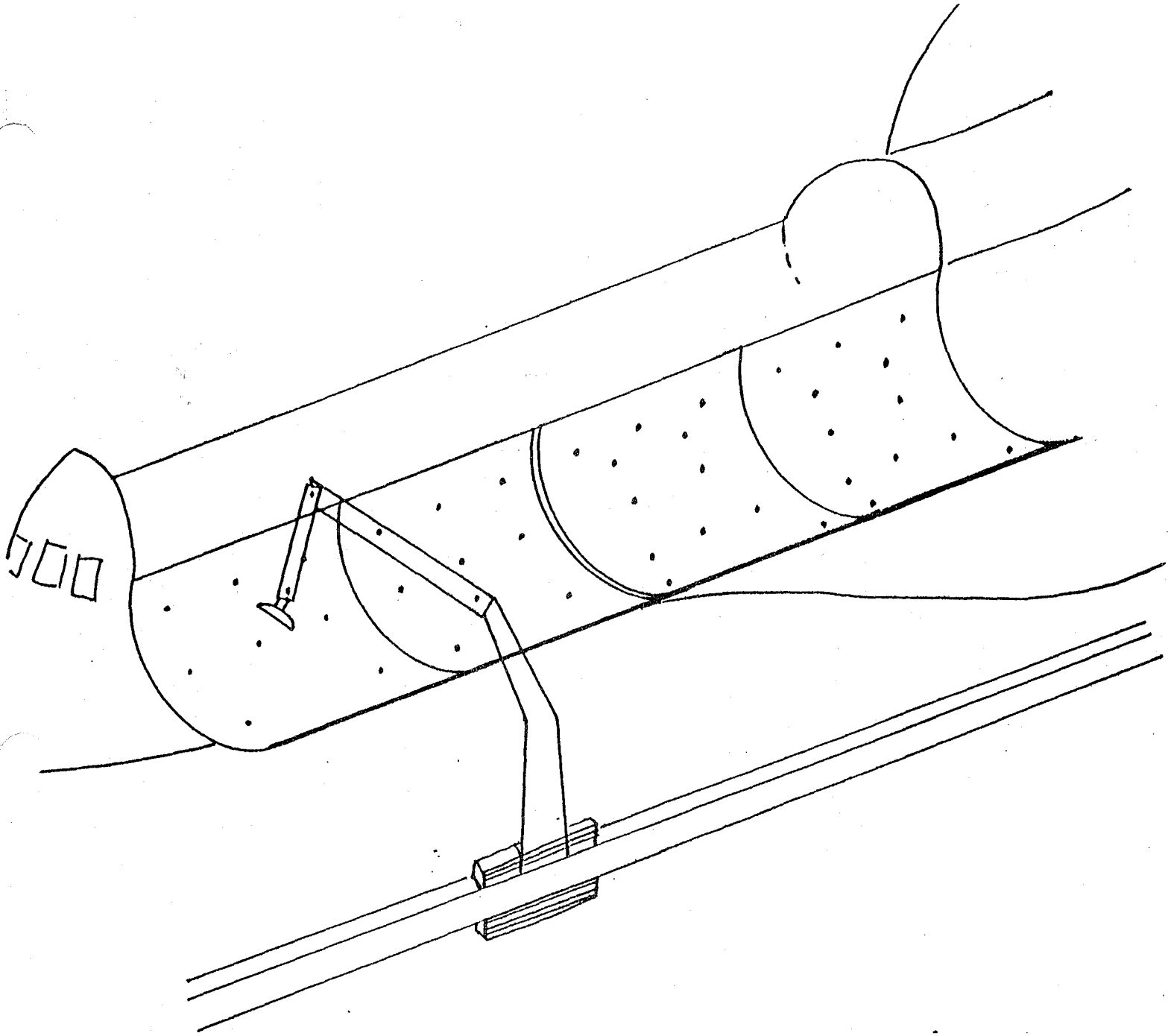

Figure 2-2 The ARID Robot With Radiator Panels

perpendicular to the surface, near one end of the track. It will then use the prismatic joint to move in a path parallel to a long series of inspection points and photograph them. At the end of one such inspection, at the end of the track, the revolute joints adjust to reposition the camera to the next inspection row. The robot then reverses didrection and repeats the procedure in the opposite direction.

It is in fact not possible to do this since the orbiter may be parked in a slightly different attitude every time. To make things even worse, the panels may be warped. The robot must be able to compensate for these and guide the camera accurately therefore must articulate the revolute joints as well as the prismatic one. Compensation can be accomplished if the panel shape can be effectively reported to the robot without the need to identify every inspection point. It is therefore necessary to be able to mathematically define the position of the fiducial points on the panel under test as well as their ideal locations.

With this information the robot would have to be able to generate a transfer function. This function would enable the robot to find the actual location of each inspection point given its ideal location. Since there are only a limited number of fiducial points, any method which is to be viable for an open loop solution, must do it reliably with these. If the order of the model exceeds the minimum number of fiducials needed, another method or a closed loop solution must be found.

III. THE FRAME SHIFT/WARP COMPENSATION PROBLEM

3.1 METHOD TO BE EVALUATED

Boeing was working on the frame shift/warp compensation problem and identified a method which showed promise. The method utilized two coordinate systems to identify the inspection points in redundant fashion. The method is described in greater detail in Appendix A.

IThe method employs the use of two coordinate systems each having a complete set of inspection points defined relative to its coordinate axes. It then generates two vectors, from the robot origin to the origins of the two coordinate axes. The relative orientations of the coordinate systems and the robot origin can and must be determined. The vector used for inspection is composed of the average of the two different vectors, from the robot origin, used to identify the inspection point.

In an unwarped system, the average vector will point to the exact inspection point since both vectors identify the point exactly. In a warped system it is hoped that the average, or weighted average of the vectors, gives a close approximation of the exact inspection point. The method requires the selection of four fiducial points (points whose coordinate positions can be verified) located in a rectangular layout on the radiator surface. Two dimetrically opposed points will be used as the origins of the two coordinate systems.

Each of the coordinate systems is defined by a vector from the origin to one of the adjacent points and another vector, also beginning at the origin, perpendicular to the first and pointing in the direction of the other adjacent point. The third coordinate is defined by the cross product of the two vectors. Each of these two coordinate systems must relate to the robot origin. Handling this step, especially in view of the fact that it must be done twice for every inspection point, will require a considerable amount of computing.

Every inspection point in the unwarped surface is known in terms of the two coordinate systems. It is therefore necessary to store two different values for each inspection point. The two different

vectors needed to define each inspection point must be known to the robot. When the surface is warped, two new vectors from the robot origin to the origins of the two coordinate systems must be known. The inspection point location is approximated by using the average of the two different vectors defining the point through the two coordinate systems. The procedure was being evaluated by considering it as it applied to the actual radiator panel. This approach proved too cumbersome to give any useful results.

To evaluate and determine if the method is viable a simple method that gave absolute results was needed. As part of the project we were asked to develop a way of verifying or rejecting the method. The method was evaluated for a simplified surface (see Appendix A) and was found to have errors as large as half the maximum induced warp. That is to say that if one corner of a linearly warped panel is displaced from ideal by some value W it is possible to have errors as great as W/2. Since it is essential to keep absolute error to less than 0.125 inches, W could not be allowed to exceed 0.25 inches.

Empirical measurements show that W is in fact in the order of inches. There is no reason to assume that the method is more applicable to curved surfaces, it was, therefore, concluded that a more accurate method was needed.

3.2 CURVE FITTING

If two non coincident points are on the xy plane, a unique line $y=mx + b$ can be drawn through them. For three points in the xy plane, a continuous function $y = ax^2 + bx + c$ can be found to exactly go through these points (assuming every x has only one y value). It is well know that in general, an n^{th} order function can be uniquely found for any n+1 points.

It therefore may seem reasonable to sample all the fiducials and use the displacements of each from its ideal position to find the warp of the radiator surface. It also may seem reasonable to sample more points in order to get a "better model". This, however, is not so. The sampling of many points is useful in verifying a model. A model may be developed from the statistics obtained from the sampling of a very large number of points. When only a small number of points is available, it is essential to know what kind of warping the panel may undergo. If warp is truly linear, sampling more than two points

and computing their displacement from ideal should result in a linear progression of error values.

If the ideal xyz locations of three colinear points is known and the actual positions are measured there may be some error in all the readings. A best fit line may be drawn using these points to help minimize this error. If frame shift was already taken into account, and measurements are accurate, the only discrepancy is due to warp. Warp can be measured by aligning one of the ideal end points with the corresponding measured point. All the other points must then be translated by the same vector. This results in an error vector of zero length for one point. Finding the error vector for the other points should result in a monotonically increasing error function as the measurements are taken farther from the first point.

If the points are equally spaced, the error vector should have approximate lengths l, 2*l, 3*l etc. If the plotting of the vector length versus distance is not a straight line it will be necessary to decide how to proceed. If a linear model is valid, a straight line best fit should be used. Trying to use all the points and fitting a curve to them will result in the problem of oversampling. Oversampling is the condition where a curve is assumed higher order than it actually is. The order of the curve should not be based solely on the samples taken. If a line is expected, when points are read with a certain degree of error, then fitting a straight line in conjunction with these points is the way to proceed. Fitting a second order curve to fit these points will not accurately depict what is actually occurring even if the curve seems to fit better.

It must be stressed that the model should dictate the order of the polynomial and not the number of points being considered. If more points are collected the negative effects of noisy readings will be filtered out when some method of curve fitting is applied. It should therefore be stressed that if the method used bases all the calculations on only four fiducials, these must be read as accurately as possible. Additional points may be read to reduce the effects of noisy readings but the order of the curve must be kept fixed. In order to have confidence in the model selected, data must be taken on orbiter panels (or accurate mock-ups) and based on these data, select a suitable model.

Some examples are illustrated in Figure 3-1. The first case shows two points. There is only one way of drawing a line through these.

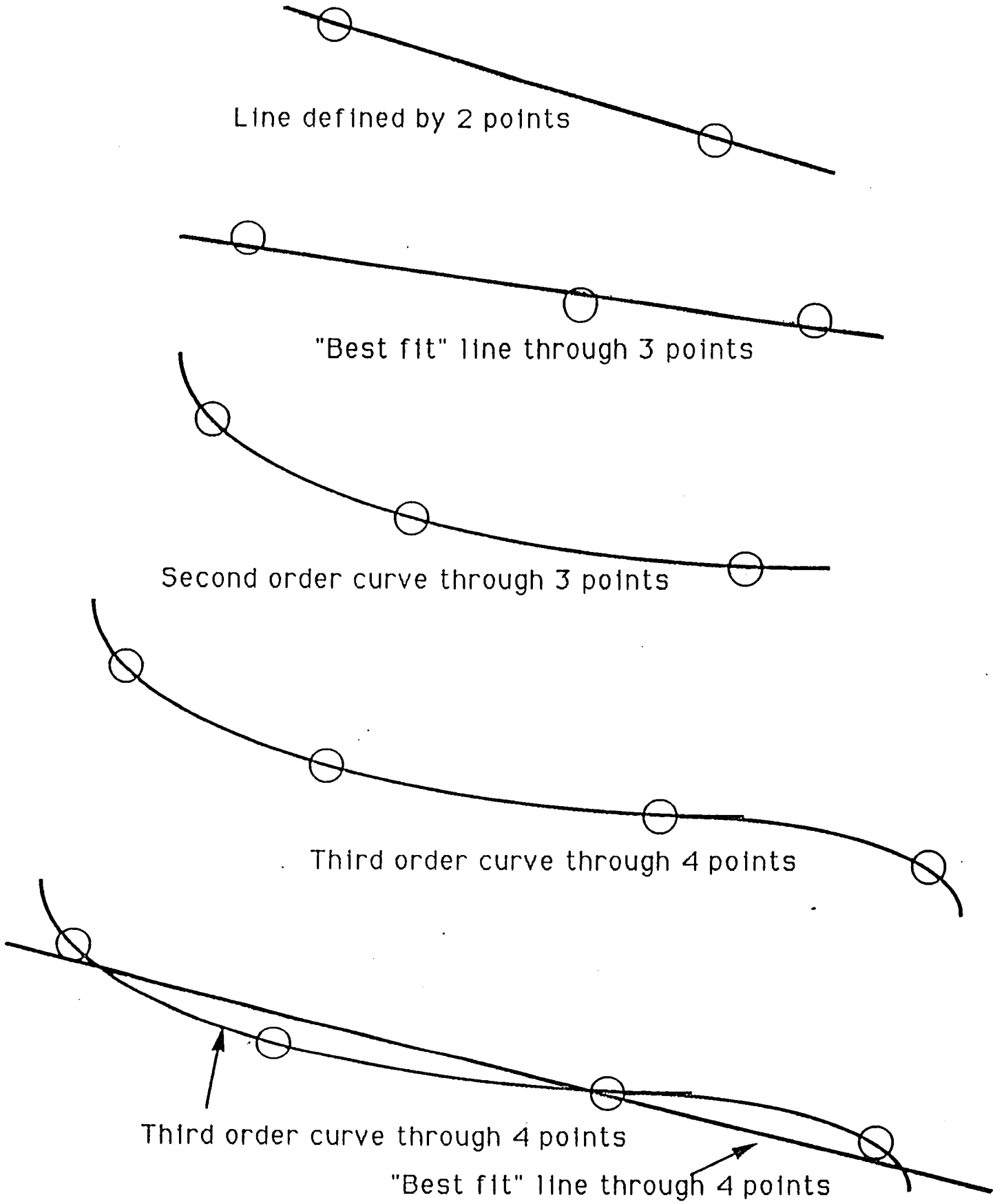

Figure 3-1 Curve Fitting Examples

the second and third show three points each. If it is known that the actual curve is a line, a best fit must be tried. If, however, the points are known to be exact, the curve must be of at least second order. The bottom two illustrate how with four points, a third order curve can be determined, for second or first order, some curve fitting is necessary.

3.3 METHOD DEVELOPED

Before and after each mission, the orbiter will be parked and inspected. The orbiter may be rotated and translated about three axes, the doors might be open at different angles and be warped as well. The rotation, translation and warp are relatively small in terms of radiator dimensions, but unacceptably large for accurate robotic inspection especially when the system is to operate in open loop. Several methods were considered for solving the warping problem including a variety of curve fitting approaches. All methods that were tried had a variety of shortcomings therefore it was decided to develop a totally new method.

The ARID system knows the ideal xyz coordinates of each inspection point. The actual inspection point is at some location x'y'z'. To inspect this point, it must be given information about the orbiter orientation and panel warp. It is desired to have an effective, accurate way of performing the proper conversion from an ideal inspection point to the actual orbiter surface. The radiators are effectively straight and rigid along the hinge line. If the edge farthest from the hinges is also found to be straight, the linear warp model seems reasonable.

If warp can be modeled, then it is possible to obtain the exact mathematical equations that govern the effects of translation, rotation and warp. By assuming that superposition holds and that warp is linear, it is possible to consider warp and translation separately then combine their effects. Translation includes rotation about the three axes. The problem is thus greatly simplified by separately considering the rigid body rotation, translation then the effects of warp. Without warp, knowing the ideal as well as displaced coordinates of three fiducials, displacements and angular rotations can be determined. This is true since it is simply the displacement of a rigid body in space.

By using only three points to determine a surface, noise in the readings could cause problems. To avoid this, it may be desirable to measure various points and use a curve fitting tecnique to make a best fit. Answers to these questions are, however, not within the scope of this report. It will be assumed that the fiducials are precisely located, are read accurately, and the linear warp model accurately defines the surface in question. The measured angles of

rotation were found to be less than one degree around any axis. Based on empirical data, warp of the doors is of the order of 2 inches for a linear distance of 15 feet. This is a small angle and permits the use of several approximations which greatly reduces necessary computations.

As an example, a system of four points will be used to illustrate the linear warp model method, see Figure 3-2. The points are considered to be at the corners of a rectangle, and reside on the xz plane. These simplifications make the method easier to understand and do not invalidate the fact that it works with some restrictions in a non coplanar point system.

Computer programs were written to evaluate the effectiveness of the procedure. A computer program was written in C++ to take a point in an xyz coordinate system, rotate and translate it. This and other programs were used as tools to make qualitative and quantitative measurements on the effects of small angle approximations on the error. It was found that first order approximations were accurate to much less than 1% error for values being considered. This allowed for many simplification to be used while keeping the results very accurate.

Theoretically, if the shape of the orbiter door is known, an accurate model for warpage is known and information about four of the ideal and actual fiducial points is known, the rotation translation and warpage can be obtained. For simplicity of explanation of the procedure, warpage has been assumed linear. Higher order warp can be handled in similar fashion, but will not be discussed here since the need has not been shown. The method will, however, be equally valid for more complex warp models provided that they are accurate representations of reality, there are a sufficent number of fiducials and superposition holds. This assumption is valid if the warp can be linearly superimposed on a surface of any shape.

This is not exactly correct, however, it is very close and well within our acceptable tolerances. Linear warp will be considered throughout this report unless it is otherwise stated.

EXAMPLE: Refer to Figure 3-2. Consider a rectangle in the xz plane. If the corner farthest from the origin is lifted in the z direction by a small amount, and if linearity is asssumed, the vertical displacement of any point on the surface may be calculated. The

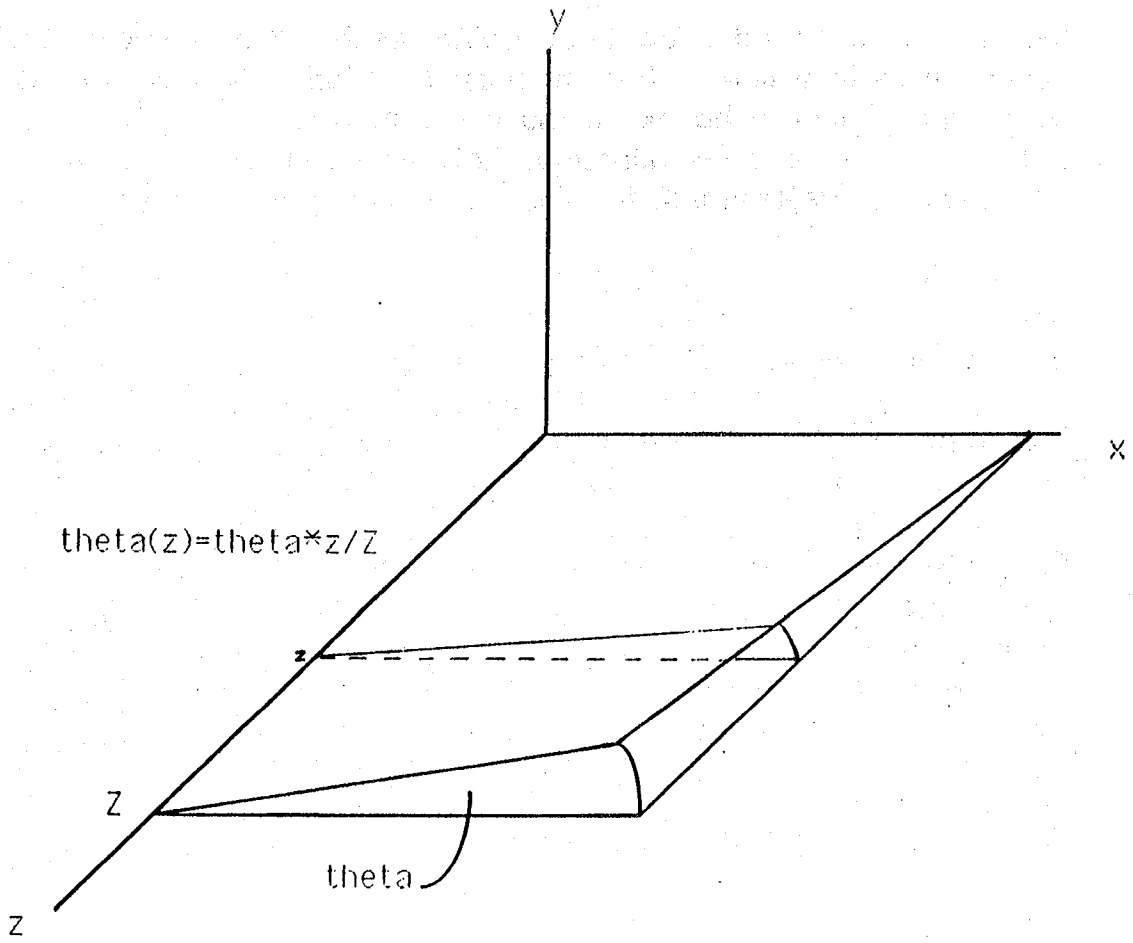

Figure 3-2 Planar Surface With Linear Warp

angle of any line formed by the intersection of the surface and a plane parallel to the xy plane is also easily determined. Assuming the four points in the xyz coordinate system are: (0,0,0), (X,0,0), (X,Y,0), and (0,Y,0). For this example the points are all coplanar.

Warping may be modeled by the lifting of one of the corners in the positive z direction, say delta z. Appllying it at the (X,Y,0) corner, moves the point to (X,Y,delta z). Actually if a corner is lifted the effected sides actually rotate, however, since the rotation is so slight, typically less than one degree, it may be modeled as a linear translation perpendicular to the xz plane. Given the linear warp assumption, the vertical displacement at any point (x,y) is:

$Z(x,y) = (x/X)*(y/Y)*(\text{delta } z)$

Therefore given four points allows the calculation of the vertical displacement at any point

3.4 LINEAR WARP

The assumptions are that with this warp, all planes parallel to the xz planes that intersect the warped surface are straight lines. The same is true for planes parallel to the yz plane. To define the warped surface it suffices to define the plane and the displacement at one corner. For the previous example three points are on the xz plane. The fourth corner is lifted by some value delta z.hat is if three points define the plane, finding how high above the plane the fourth point is defines the warp.

Given the same linear warp assumptions on any curved surface the same principle may be applied. Three of the fiducial points are used for defining a plane, then the height above (or below) the plane is established for the unwarped surface. With warp, the same plane is defined and the new displacement is obtained. The difference in the two numbers in conjunction with distance between the "warped" point and the fiducial is the warp.

The intersection of a plane perpendicular to the defined plane and parallel to one of the vectors (see Fig. 3-3) shows that the intersection of the surface is an undistorted curve that is rotated. Since warp is considred linear, the "angle of warp" is approximated by the inverse tangent of the warp divided by the linear dimension (call this theta). With this assumption, the angle of rotation of the

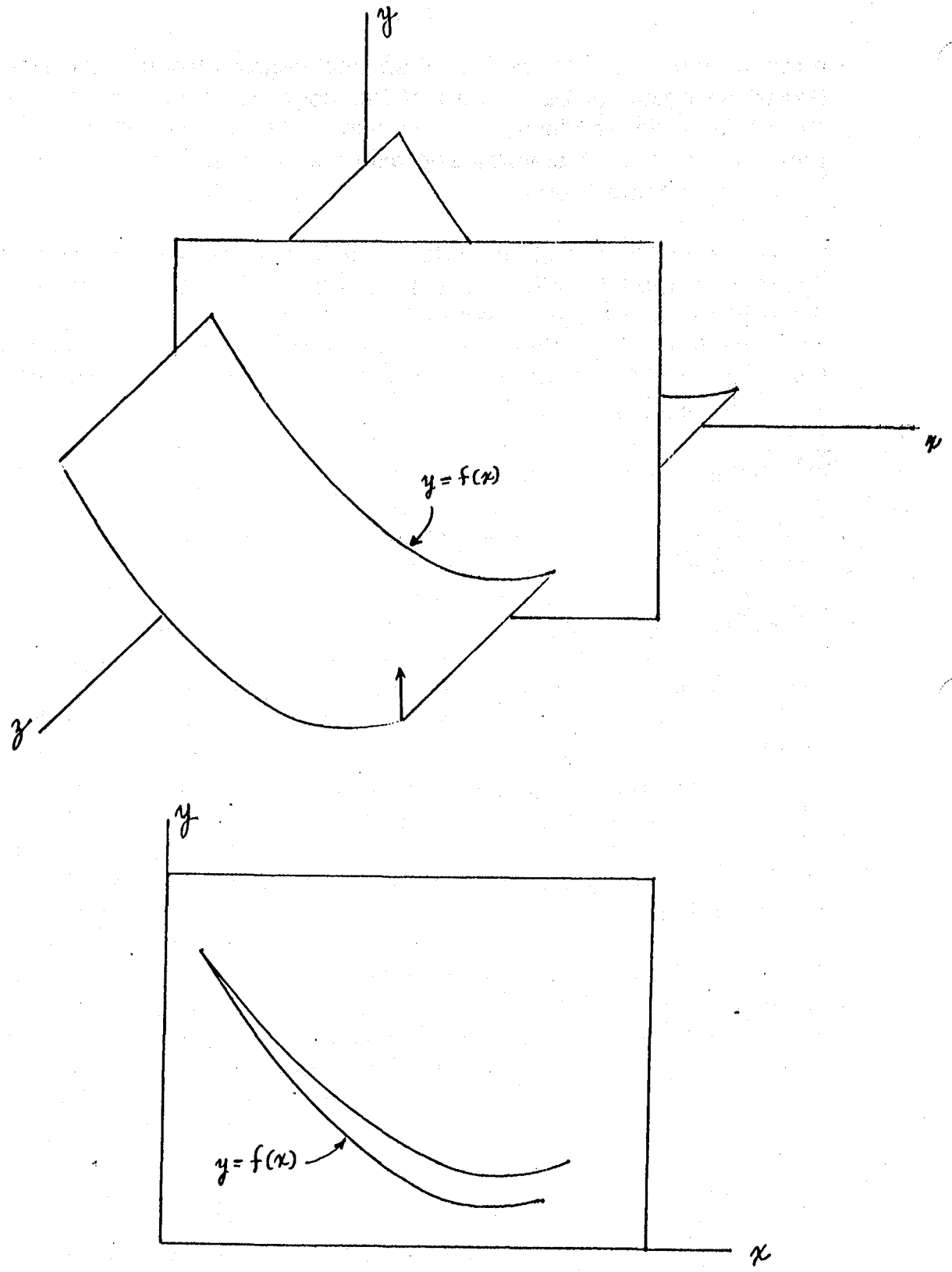

Figure 3-3 Warping With No Distortion

curve in question is theta at one extreme, 0 at the other and theta*x/X in between.

It should be stressed that as long as linear warp is assumed any shape surface can be handled this way with minimal amout of math necessary for the location of the inspection points. If linear warp cannot be assumed, a model must be selected that accurately defines the warping. This could be a second, third or some higher order system. the priciple should still hold.

3.5 COORDINATE TRANSFORMATION

When considering different coordinates to identify a given point in space it is often essential to convert between the two systems. Although most texts dealing with introductory robotics topics cover this in detail, a brief description is included here for convenience.

Given a point described in a rectangular coordinate system, the point may be displaced along the three axes or rotated about them. The order that these operations are performed matters, as well as to know relative to which system the operations are being performed. Given a general point P in space xyz, translating this point by some vector v then rotating the new point by some angles thetax, thetay and thetaz about the thee coordinate axes in turn, becomes a new point P'. To return to the original point, P' must first be rotated by minus thetaz, minus thetay, minus thetax, then translated by minus v in that order. For very small angles and an approximate solution, the order of angular rotations is unimportant.

To translate any arbitrary point Pi = (x, y, z) by an arbitrary vector v = ai + bj + ck results in a new point Pi' = (x+a, y+b, z+c).
Let $Vi = [x\ y\ z]^T$ be the vectorial representation of point Pi
Rotating Pi about the x axis by angle a is accomplished by multiplying Vi by Rot(x,a)
Where

$$Rot(x,a) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \text{Cos } a & \text{-Sin } a \\ 0 & \text{Sin } a & \text{Cos } a \end{bmatrix}$$

Similarly

$$Rot(y,a) = \begin{bmatrix} \text{Cos } a & 0 & \text{Sin } a \\ 0 & 1 & 0 \\ \text{-Sin } a & 0 & \text{Cos } a \end{bmatrix}$$

$$\text{Rot}(z,a) = \begin{bmatrix} \text{Cos a} & \text{-Sin a} & 0 \\ \text{Sin a} & \text{Cos a} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

By using these matrices it is possible to move any arbitrary point Pi, represented by vector Vi, by two distinct rotations so that the vector will point in any desired direction. Figure 3-4 depicts an arbitrary vector in space with convention of positive angle rotations. Similarly three rotations about the three axes can align any arbitary plane into one parallel to any desired orientation. Figure 3-5 illustrates how a rectangle in the xy plane can be rotated into any arbitrary orientation. Reversing the steps can take a rectangle, with one corner at the origin, and rotate it into any desired plane.

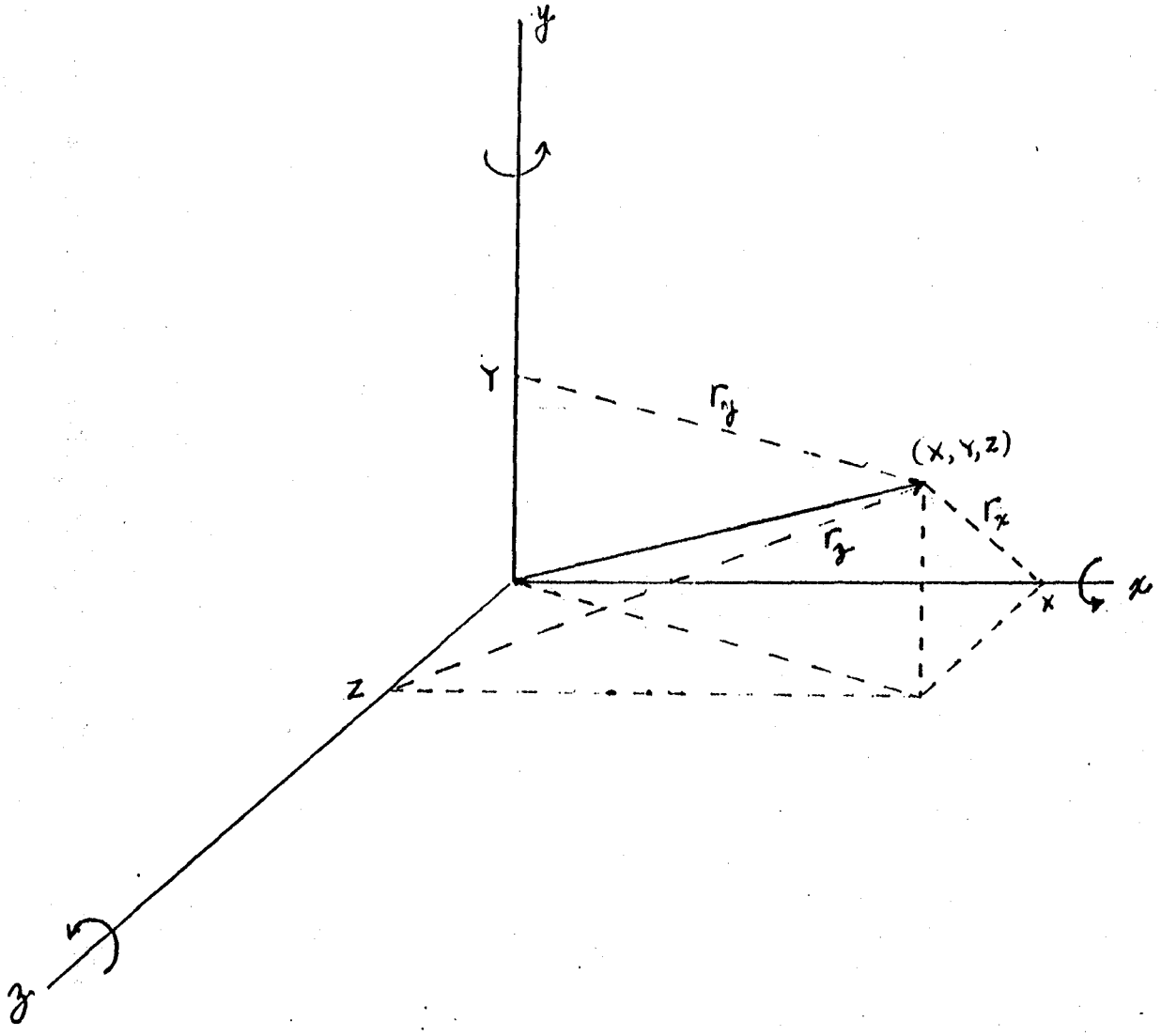

Figure 3-4 Vector In 3 Space With Positive Angular Rotation

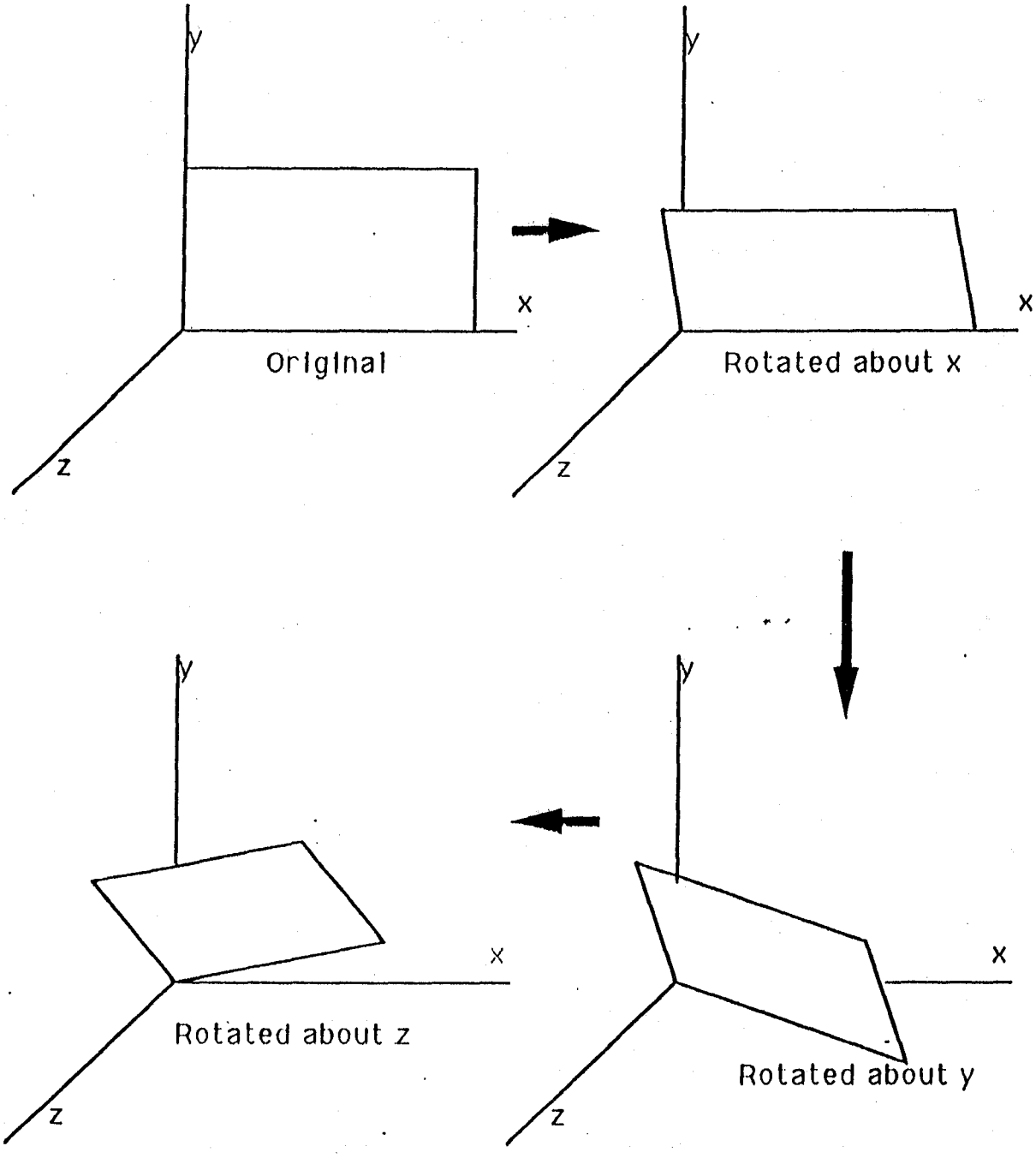

Figure 3-5 Rotation of a Surface In Space

IV. CONCLUSIONS

It was the purpose of this study to evaluate a method which was under consideration for use in compensating for rotation, translation and warp. The findings were that the method could not satisfy the stringent requirements. It therefore was necessary to develop a more reliable, accurate method.

Given information about the position and warp of the radiator panels, the ARID system would be able to make adjustments for inexact parking attitude, door opening angle and warpage. This information was to be used to allow the ARID system to compensate for these inaccuracies and to locate the desired inspection points. The method described in the report proved to be easy to implement and accurate to within a fraction of a degree and small fraction of an inch. The procedure is based on a linear warp model.

The method is also applicable to higher order warp. The procedure needs accurate measurements of only four fiducial points and knowledge of their ideal placement for the linear warp model. It is possible to simplify the complex trigonometry needed to solve this problem by the fact that the actual rotation angles and warpage are small. The small angle approximations were evaluated for the expected angles. It was found that small angle approximations could often be used. This replaces the computing of trigonometric functions with simple numbers. This will greatly reduce the computational demands on the computer.

The measurement of the four fiducial points is sufficient data to allow the method to compute the displacement in the three axes, the rotations about the three axes and the amount of linear warp. When the ARID system needs to inspect a point whose coordinates are in its memory it must be able to make the adjustments from ideal to actual. The inspection point is read from memory then compensated for the warp, rotation and translation. The ARID system then has the exact location of the desired inspection point. It should be mentioned that the application of this procedure with the ARID system may require some changes in the radiator panels alignment procedure.

APPENDIX A

THE BOEING FRAME SHIFT/WARP COMPENSATION METHOD

The procedure devised by Boeing, was intended to automatically deal with both the problem of frame shift and warp at the same time. The method will be described in conjunction with Figure A-1. Given the ARID system has its origin at x,y,z = 0,0,0. In this system there are four Fiducial points labeled F1, F2, F3 and F4. No three of these points may be colinear.

Two coordinate systems are constructed at dimetrically opposed fiducials, i.e. F1 and F3. The origin of the u1,u2,u3 coordinate system is at F1. u1 is defined as the unit vector pointing from F1 to F2. u2 is defined as the unit vector perpendicular to u1 in the plane defined by u1 and fiducial F4. u2 is arbitrarily selected to point in the general direction of F4. u3 is defined as the cross product of u1 and u2 using the right hand rule. v1, v2 and v3 are similarly defined using fiducials F3, F2 and F4. Every inspection point Pi is defined in the two coordinate systems. Pi is vector u when seen from u1, u2, u3, and v if seen from the v1, v2, v3 coordinate system. When the surface is displaced, rotated and warped every point Pi becomes some Pi'.

To find this point, the ARID system must, based on the new fiducial points F1', F2' F3' and F4', construct the new u and v coordinate systems. Since all fiducials are known to ARID, the origins of the two coordinate systems is known. Using the old (pre warp) values for vector u, within the new coordinate system some point in space is identified. This does not coincide with Pi' because of warp. Repeating the process, this time with vector v again fails to locate Pi'. It was hoped that the average or weighted average of these two vectors would be close to the true value of Pi'. The evaluation of the procedure was done by applying it to a rectangualr planar surface like that in Figure 3-2 and determine what error values resulted.

Applying a linear warp approximation to a rectangular surface with one corner lifted by some small value W resulted in errors of magnitude W/2. Since it is known that W may be in the order of inches while tolerances are in the order of 0.125 inches the method was discarded. It was also concluded that attempts to weight the

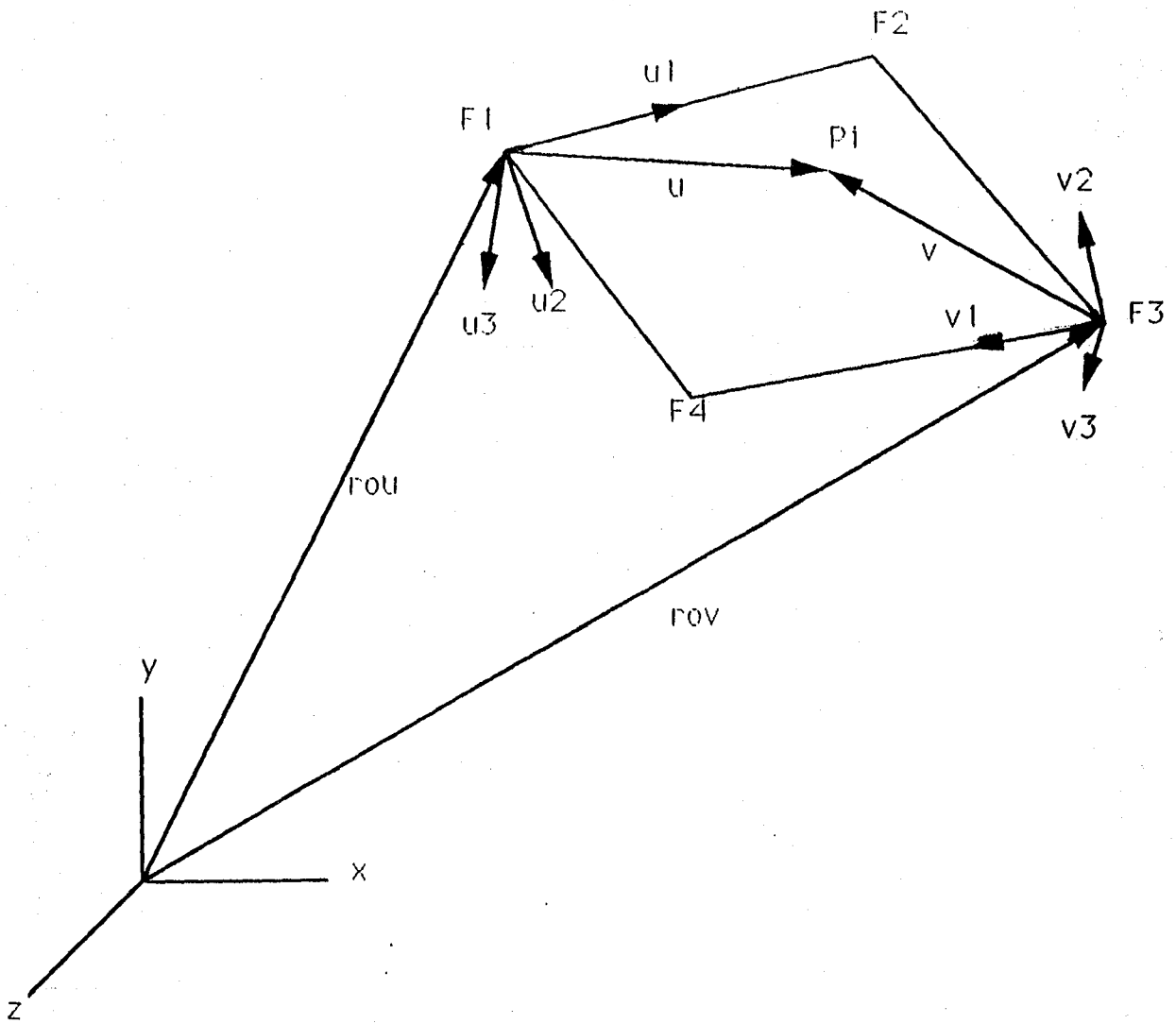

Figure A-1 The Boeing Method of Frame Shift/ Warp Compensation

two vectors were doomed to failure since, for this example at least, all errors were of the same sign.

APPENDIX B

SOFTWARE

Some brief programs were written in Turbo C++ during the course of this project. The first one, called WARP1.C, was used to generate synthetic data for the evaluation of the Boeing Frame Shift/Warp Compensation method. This program was based on the vertical displacement approximation. It applied the method to a rectangular plane which was "lifted" at one corner to simulate a linear warp of the surface. The exact vertical displacement of the surface at each xy location can be calculated given the linear warp assumption. The program also located the exact displacements as calculated by the two separate coordinate systems. The results showed that the error was one half the maximum warp displacement. The method is also dependent on which fiducials are selectd as the two coordinate origins. It was based on these findings that the method was rejected as a viable alternative.

CARL1.C is a program which takes four points in space, rotates and translates them to locate the new points. This computes the "exact" positions of the translated surface.

The third program, FRAMESHI.C, uses two sets of points, the original and shifted ones. Based on this data the warpage is approximated by the linearized approach developed in the report.

```
/*      Program Name: ### WARP1 ###  June 18, 1991 update 7/3/91      */
/*      This program computes the vertical displacements from a fixed */
/* plane to the corresponding point on a warped surface.  The actual  */
/* displacement is calculated then compared to the value obtained by  */
/* the method suggested by Alex Ladd.                                 */
/*      The program writes the data into a file named trash.dat.  To  */
/* view the data exit to the DOS shell and do a TYPE trash.dat.  For  */
/* a hard copy, redirect the output to the printer.                   */
/*                                                                    */

#include <stdio.h>
#include <conio.h>

int main()

{
        FILE *fp;
        int X,Y,DZ;
        float x1,y1,za,vd,ud,avg;
        fp= fopen( "trash.dat","wt" );
        DZ=2;        /*  Displacement of corner (X,Y) in inches */
        fprintf(fp, "\n Disp.(in inches) of corner (X,Y) is %4d ",DZ);
        fprintf(fp, "\n Xdisp Ydisp Actual DispV DispU AvgUV Error \n");
        X=16*12;     /*  Size of panel in inches X dimension */
        Y=12*12;     /*  Size of panel in inches Y dimension */
        for (y1 = 0; y1 <= Y; y1 = y1 + 24)
         {
         for (x1 = 0; x1 <= X; x1 = x1+24)
           {
            za=(x1/X)*(y1/Y)*DZ;
            vd=(y1/Y)*DZ;
            ud=(x1/X)*DZ;
            avg=(vd+ud)/2;
            fprintf(fp, "\n %3.0f %3.0f      %3.3f %3.3f %3.3f ",x1,y1,za,vd,ud,
            fprintf(fp, "%3.3f %3.3f ",avg,za-avg);
            }
         }
         fprintf(fp, "\n ");
        fclose( fp );
}
```

```
}/*     Program name  ###    CARL1    ###  July 3,1991      11:24 A.M.     */
/*                                                                          */
/*      In this program uses four original points.  It rotates and          */
/* shifs them into four new points.  Each of the four point systems use     */
/* three points to define a plane and the fourth to define warp.            */
/*      The two planes are defined by taking the cross product of two       */
/* vectors (which are defined by the three points).  The output is the      */
/* x, y and z components of the resultant vector as well as the square      */
/* of the magnitude of the cross product vector.                            */
/*                                                                          */
/*      This program converts four 3 X 1 vectors (iv[3][4]), in the xyz     */
/* coordinate system into four that are rotated in the three axes then      */
/* translated to become zz[3][4].                                           */
/*      The rotation is first about the x axis, the y axis the z axis       */
/* and finally translated by a vector.                                      */
/*      To run this program, it must be edited to define angles thx,        */
/* thy, thz and vector xyz.                                                 */
/*      To reverse this effect, the operations must be taken in reverse     */
/* order.  The vector must be the negative of the original vector and       */
/* the rotation angles the negative of the original angles).                */

#include <stdio.h>
#include <conio.h>
#include <math.h>

/*      Rotation angles about the three coordinate axes                     */

#define thx 90*3.141592654/180  /* Theta is in radians i.e. 5 deg   */
#define thy 0*3.141592654/180  /* Theta y in radians  or 10 degrees */
#define thz 0*3.141592654/180  /* Theta z in radians  or 15 degrees */

int main()
{
        int N,x,y,k;
        double rotx[3][3]={
        {1, 0, 0},
        {0, 1, 0},
        {0, 0, 1}
        };
        double roty[3][3]={
        {1, 0, 0},
        {0, 1, 0},
        {0, 0, 1}
        };
        double rotz[3][3]={
        {1, 0, 0},
        {0, 1, 0},
        {0, 0, 1}
        };

/*      Input vectors are called iv[][].  output vectors zz[][].            */

        double v[3],w[3],xx[3],yy[3],zz[3][4],I,J,K,LSQ;
        double iv[3][4]={
        {0, 10, 10, 0},
        {0,  0,  5, 5},
        {0,  0,  0, 0}
        };
```

```
/*    The displacement vector                          */
        double xyz[3] = {1, 11, 3};

        rotx[1][1]=cos(thx);
        rotx[1][2]=-sin(thx);
        rotx[2][1]=sin(thx);
        rotx[2][2]=cos(thx);

        roty[0][0]=cos(thy);
        roty[0][2]=sin(thy);
        roty[2][0]=-sin(thy);
        roty[2][2]=cos(thy);

        rotz[0][0]=cos(thz);
        rotz[0][1]=-sin(thz);
        rotz[1][0]=sin(thz);
        rotz[1][1]=cos(thz);

/*      the four vectors are already defined     */

       for (k=0;k<4;k++)
        {


/*      Rotate about x axis      */
/*      printf("\n The vector rotated about the x axis is:\n");*/

        for (x=0;x<3;x++)
          { v[x] = 0;
            for (y=0;y<3;y++)
              {
              v[x]=v[x] + rotx[x][y] * iv[y][k];
              }

          }

/*      rotate about y axis                          */
/*      printf("\n The vector rotated about the y axis is:\n");*/


        for (x=0;x<3;x++)
          { w[x] = 0;
            for (y=0;y<3;y++)
              {
              w[x]=w[x] + roty[x][y] * v[y];
              }
          }

/*      rotate about z axis                          */

        for (x=0;x<3;x++)
          { xx[x] = 0;
            for (y=0;y<3;y++)
              {
              xx[x]=xx[x] + rotz[x][y] * w[y];
              }
          }

/*      Translate in xyz coordinates                 */
        printf(" The translated vector is:\n");
```

```
        for (x=0;x<3;x++)
              {
              yy[x]=xx[x] + xyz[x];
              zz[x][k]=yy[x];
              printf("%8.8e\n",zz[x][k]);
              }
    }

/*  Get the xyz components of the two input vectors that will be used in the
  cross product calculation.  v[0] is the x component, v[1], the y component
  and v[2] the z component ov vector v.  Similarly for vector w.          */

        for (x=0;x<3;x++)
          {
          v[x]=iv[x][1]-iv[x][0];
          w[x]=iv[x][3]-iv[x][0];
          printf("\n v[x] w[x] %8.8f %8.8f",v[x],w[x]);
          }
  /*  This portion takes the two vectors v and w and computes their cross
    product.  I is the x component of resulting vector, J the y component
    and K the z component.                                                */

          I=v[1]*w[2]-v[2]*w[1];
          J=-(v[0]*w[2]-v[2]*w[0]);
          K=v[0]*w[1]-v[1]*w[0];

        LSQ=I*I+J*J+K*K;          /* LENGTH SQUARED */
        printf("\n Plane defined by the input vectors");
        printf("\n I J K LSQ %8.8f %8.8f %8.8f %8.8f \n",I, J, K, LSQ);

        for(x=0;x<3;x++)
          {
          for(y=0;y<4;y++)
            {
            printf("  %3.3f %3.3f ",iv[x][y],zz[x][y]);
            }
            printf("\n");
          }

/*  Get the xyz components of the two output vectors that will be used in the
  cross product calculation.  v[0] is the x component, v[1], the y component
  and v[2] the z component ov vector v.  Similarly for vector w.          */

        for (x=0;x<3;x++)
          {
          v[x]=zz[x][1]-zz[x][0];
          w[x]=zz[x][3]-zz[x][0];
          printf("\n v[x] w[x] %8.8f %8.8f",v[x],w[x]);
          }
  /*  This portion takes the two vectors v and w and computes their cross
    product.  I is the x component of resulting vector, J the y component
    and K the z component.                                                */

          I=v[1]*w[2]-v[2]*w[1];
          J=-(v[0]*w[2]-v[2]*w[0]);
          K=v[0]*w[1]-v[1]*w[0];
```

```
LSQ=I*I+J*J+K*K;          /* LENGTH SQUARED */
printf("\n Plane defined by the rotated vectors");
printf("\n I J K LSQ %8.8f %8.8f %8.8f %8.8f \n",I, J, K, LSQ);

for(x=0;x<3;x++)
  {
  for(y=0;y<4;y++)
    {
    printf("  %3.3f %3.3f ",iv[x][y],zz[x][y]);
    }
    printf("\n");
  }
```

```
/*      Program name  ###    FRAMESHI(FT).C   ###  July 10,1991 4:50 P.M. */
/*      In this program uses four original points.  The rotated and     */
/* shifted points are also known.  Each of the four point systems use */
/* three points to define a plane and the fourth to define warp.        */

#include <stdio.h>
#include <conio.h>
#include <math.h>

int main()
{
        int x,y;

/* tv[3][4] is temporary vectors, dsp[3] is displacement vector          */
/* tx is x displacement, ty and tz are y and z displacements             */

        double tv[3][4],dsp[3],tx,ty,tz,W;

/* ov[3][4] is original vectors (or points), dv[3][4] is displaced vect.*/
/* ov[x,y,z][vect.#-1].  i.e.  ov[1][3] is the y comp. of vector 4       */

        double ov[3][4]={
        {0, 180, 180,   0},
        {0,  0,  144, 144},
        {0,  0,    0,   0}
        };
/*  The following assumes that X=180,Y=144,tx=.01,ty=.015,tz=.02
        W=4, dx=1, dy=3 and dz=5                                  */

        double dv[3][4]={
        {1, 181, 181,   2.12},
        {3,  6.6,  147, 149},
        {5,  7.7,    7.74,  6.44}
        };

/*  P1=(0,0,0)  PD1=(dx,dy,dz)
    P2=(X,0,0)  PD2=(X+dx,X*tz+dy,-X*ty+dz)
    P3=(X,Y,0)  PD3=(X+dx,Y+dy,Y*tx-X*ty+dz+W)
    P4=(0,Y,0)  PD4=(-Y*tz+dx,Y+dy,Y*tx+dz)
    where dx, dy and dz are linear displacements in the x, y and z dir.
    tx, ty and tz are angular rotations about the x, y and z axes using
    the right hand rule.  W is the warp value.
*/

/* Update each point by the displacement (dx,dy,dz)     */

       for (y=0;y<4;y++)
        {
          for (x=0;x<3;x++)
            {
              tv[x][y]=dv[x][y]-dv[x][0];
              printf ("\n x, y, tv[x][y] %3d %3d %3.8f",x,y,tv[x][y]);
            }
           printf ("\n");
        }

/*  Calculate theta z (tz) [ PD2y-P2y = X*tz]          */
        tz=(tv[1][1]-ov[1][1])/ov[0][1];
        printf ("\n Theta z in radians is = %3.6f",tz);
```

```
/* Calculate theta y (ty) -[PD2z-P2z = X*ty]                    */
        ty=(tv[2][1]-ov[2][1])/ov[0][1];
        printf ("\n Theta y in radians is = %3.6f",ty);

/* Calculate theta x (tx) -[PD4z-P4z = Y*tx]                    */
        tx=(tv[2][3]-ov[2][3])/ov[1][3];
        printf ("\n Theta x in radians is = %3.6f",tx);

/*  Calculate the warp W.  Y*tx-X*ty+W = PD3z                   */
        W=(tv[2][2]-tv[2][3]+tv[0][1]*ty);
        printf ("\n The warp value W is = %3.6f",W);

}
```

REFERENCES

[1] Paul, Richard P. "Robot Manipulators", The MIT Press, Cambridge, Massachussets, 1982

[2] Koren, Yoram "Robotics for Engineers", McGraw-Hill, 1985

[3] Swokowski, Earl W. "Calculus With Analytic Geometry," second edition, Prindle Weber and Schmidt, 1979