Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

The Van Houten library has removed some of the personal information and all signatures from the approval page and biographical sketches of theses and dissertations in order to protect the identity of NJIT graduates and faculty.

ESTIMATING 3-D MOTION AND STRUCTURE PARAMETERS OF A RIGID PLANAR OBJECT BY ESTABLISHING LANDMARK CORRESPONDENCES

To My Children, Jemmy and Diemas

and Wife Karen

Sister Rika

for all their support, faith and patience...

APPROVAL SHEET

Title of Thesis: 2) ESTIMATING 3-D MOTION AND STRUCTURE PARAMETERS OF A RIGID PLANAR OBJECT BY ESTABLISHING LANDMARK CORRESPONDENCES

1)

Name of Candidate: Herman Agustiawan
Master of Science in Electrical Engineering, 1990

Thesis and Abstract Approved: ______________ 11/28/90

Dr. Nirwan Ansari — Date
Assistant Professor
Department of Electrical & Computer Engineering

______________ 11/28/90

Dr. Andrew U. Meyer — Date
Proffesor
Department of Electrical & Computer Engineering

______________ 11/28/90

Dr. Yun-Qing Shi — date
Assistant Professor
Department of Electrical & Computer Engineering

VITA

Name: Herman Agustiawan

Address:

Degree and date to be conferred: MSEE, 1990

Date of birth:

Place of birth:

Collegiate institutions attended	Dates	Degree	Date of Degree
I.T.B. INDONESIA	1977-1983	B.E.Physics	1983
New Jersey Institute of Technology	1988-1990	MSEE	1990

Major: Electrical Engineering

Minor: Computer and Information Science

ABSTRACT

Title of Thesis: ESTIMATING 3-D MOTION AND STRUCTURE PARAMETERS OF A RIGID PLANAR OBJECT BY ESTABLISHING LANDMARK CORRESPONDENCES

Herman Agustiawan, Master of Science in Electrical Engineering, 1990

Thesis directed by: Dr. Nirwan Ansari, Assistant Professor

The application of a landmark based approach in recognizing 3-D rigid planar objects[3] along with two different motion and structure estimation algorithm [21][23] have been studied to perform motion and structure estimation tasks of moving objects. The recognition algorithm is based on a sphericity value derived from affine transformation that maps feature points in the object under observation from first to second view in the 2-D image space. The motion and structure estimation algorithm is based on a unique mapping given four or more feature point correspondences.

More than ten experiments have been done including small and large rigid motions. From all numerical experiments that have been tried, the recognition algorithm can handle the landmark matching tasks well if the object follows only small rigid motions. The algorithm, without too much destroying the sequential order of the original landmarks, is also capable of detecting the correct matches of landmarks when missing and some extraneous landmarks in the second view (after

motion), are taken into account. The algorithm, however, will fail to detect the correct matches of landmarks if the object under observation moves such that the normal of the object surface is almost perpendicular to the optical axis.

When a motion is small, only a relative depth of the observed object can be determined. The number of solutions, aside from the scale factor for the translational vector, depend on the number of the real roots of the sixth order polynomial equation. In our experiment, we have found two real and four complex roots. From the two resulting solution only one represents the actual motion parameters. If the motion is large, the number of solutions is either one or two depending on the multiplicity of the singular values of the 3×3 real matrix consisting of the pure parameters obtained from the unique mapping.

Due to the nonlinearity of the computation procedure that have been used and roundoff error generated by the computer, the results deviate up to 10% from the predefined parameters. The motion algorithm, however, is convenient, inexpensive and not shaken by missing or extraneous landmarks as long as there are more than three landmark corresponding pairs. That is, regardless of which landmarks in the observed object being used, the estimated motion and structure parameters will remain relatively the same.

Acknowledgement

I am grateful to Professor Nirwan Ansari for his invaluable guidance. I would like also to express my gratitude to OFP BPP Teknologi (Dr. Wardiman & staff), Direktorat K.K.E (Dr. L.M. Panggabean), IIE-New York (Mrs. Sandra Cervera) for giving me an opportunity to study at New Jersey Institute of Technology and for all supports that have been granted to me.

Contents

Chapter 1.

Intoduction 1

Chapter 2.

Mathematical Elements 6
- 2.1 Introduction 6
- 2.2 Affine Transformation and Sphericity 6
- 2.3 Motions 8
  - 2.3.1 Translation 9
  - 2.3.2 Rotations 11
  - 2.3.3 Rigid Motions 16

Chapter 3.

Estimating Motion Parameters From Image Sequences 17
- 3.1 Literature Review 17
- 3.2 Direct vs Two Stage Estimation 22
  - 3.2.1 Direct Estimation 22
  - 3.2.2 Two Stage Estimation 25
- 3.3 Probe and Block Matching Algorithm 28
- 3.4 Estimating 3-D Motion of a Rigid Planar Patch 35
  - 3.4.1 Small Rotation 35

3.4.2 Large Rotation 37

Chapter 4.

Computation Procedures And Experimental Results 43
4.1 Computation Procedures 43
4.2 Experimental Results 45
4.3 Further Experimental Results 54

Chapter 5.

Conclusion And Recommendation 57

Appendix:

A. Linear Equation AX=B : Programming Code in C 61
B. Sixth Order Polynomial Equation : Programming Code in FORTRAN 64
C. Singular Value Decomposition : Programming Code in C.................. 68
Bibliography 74

List of Figures

Figure 1.1: Landmarks of Various Objects 3

Figure 2.1: Rotation About Point O by θ 12

Figure 2.2: Rotation About x and y axes by 90^o 14

Figure 2.3: Direction Cosine 15

Figure 3.1: Basic Geometry for Two Stage Motion Estimation 27

Figure 3.2: Landmarks of An Aircraft 32

Figure 3.3: Flowchart: Probe And Block Matching Algorithm 33

Figure 3.4: Transformation of Coordinate System
With Orthonormal Matrix V 39

Figure 4.1: Flowchart for Estimating 3-D Motion And Object Parameters
By Establishing Landmark Correspondences 46

Figure 4.2: Landmarks of A Jeep
Before (a) and After (b) A Small Rigid Motion 47

Figure 4.3: Landmarks of A Jeep
Before (a) and After A Large Rigid Motion:
(b). Case 1; (c). Case 2; (d). Case 3 49

Figure 4.4: Landmarks of A Jeep
Before (a) and After A Large Rigid Motion - Case 2:
(b). Missing One Point; (c). Two Extraneous Points 55

List of Tables

Table 3.1: Compatibility Table 28
Table 3.2: Diagonal Support Table 29
Table 4.1: Predefined Motion And Object Parameters 50
Table 4.2: Estimated Motion And Object Parameters 51
Table 4.3: Solution To The Small Rigid Motion 52
Table 4.4: Predefined Parameters (Further Experiments) 56
Table 4.5: Estimated Parameters (Further Experiments) 56

Chapter 1

Introduction

The computer vision task in determining the movement of objects from a sequence of images is the process of replacing the interpretation of visual data perceived by human. The interpretation includes the description of the environment in terms of objects, their shape, and their motion are estimated on the basis of image space shifts that reaches a scene.

The fact that computer cannot be as capable as human visual system in interpreting moving objects leads to the need of investigation on what kinds of mathematical formulation are adequate and lead to a biologically plausible model of computation for the problem.

Some research work on these have been done. Experiments on the ability of human visual system to distinguish shape from motion stimulus was demonstrated by Wallach and O'Connell in the 1950's[26]. Subsequently, the ability to recognize the human shape from the projected motion of as many as ten points such as the various joints like elbows, shoulders and knees was discovered by Johansson[14].

Meanwhile, Attneave[5] suggested that some dominant points along an object contour are rich in information content and sufficient to characterize the structure of the object. This concept of dominant points, usually referred to as *landmarks* of the objects, was used by Bookstein[6] to study and observe the growth of biological objects. Thus it does not seem that the perception of the object shape should need an extraction of the projected trajectory of too many points. It requires only the knowledge of the positions of the landmarks in the image frame as shape features. Some commonly used features are holes, points, line, segments, curve segments or a combination of them. These features are obtained by preprocessing step such as edge detection, polygonal approximation and corner extraction[4]. Therefore, the interpretation of the moving object may become the correspondences problem of dominant points of the object which appears in a sequences of images, and the estimation of its motion parameters. Figure (1-1) shows the landmarks of various objects.

In the past, most research work on motion estimation has mainly been concerned to 2-D space. This is because the interpretation of moving objects in 3-D from 2-D images is much more complicated than 2-D motion since rotation and translation in depth are difficult to analyze. In addition, as a result of rotation in space, parts of the moving object can disappear from view. Recently, methods of estimating 3-D motion parameters, including 3-D rotations, of rigid bodies with different kind of shape have been investigated [21][22][27]. However, all of the above works have assumed that the recognition task, point matching, between two or more image

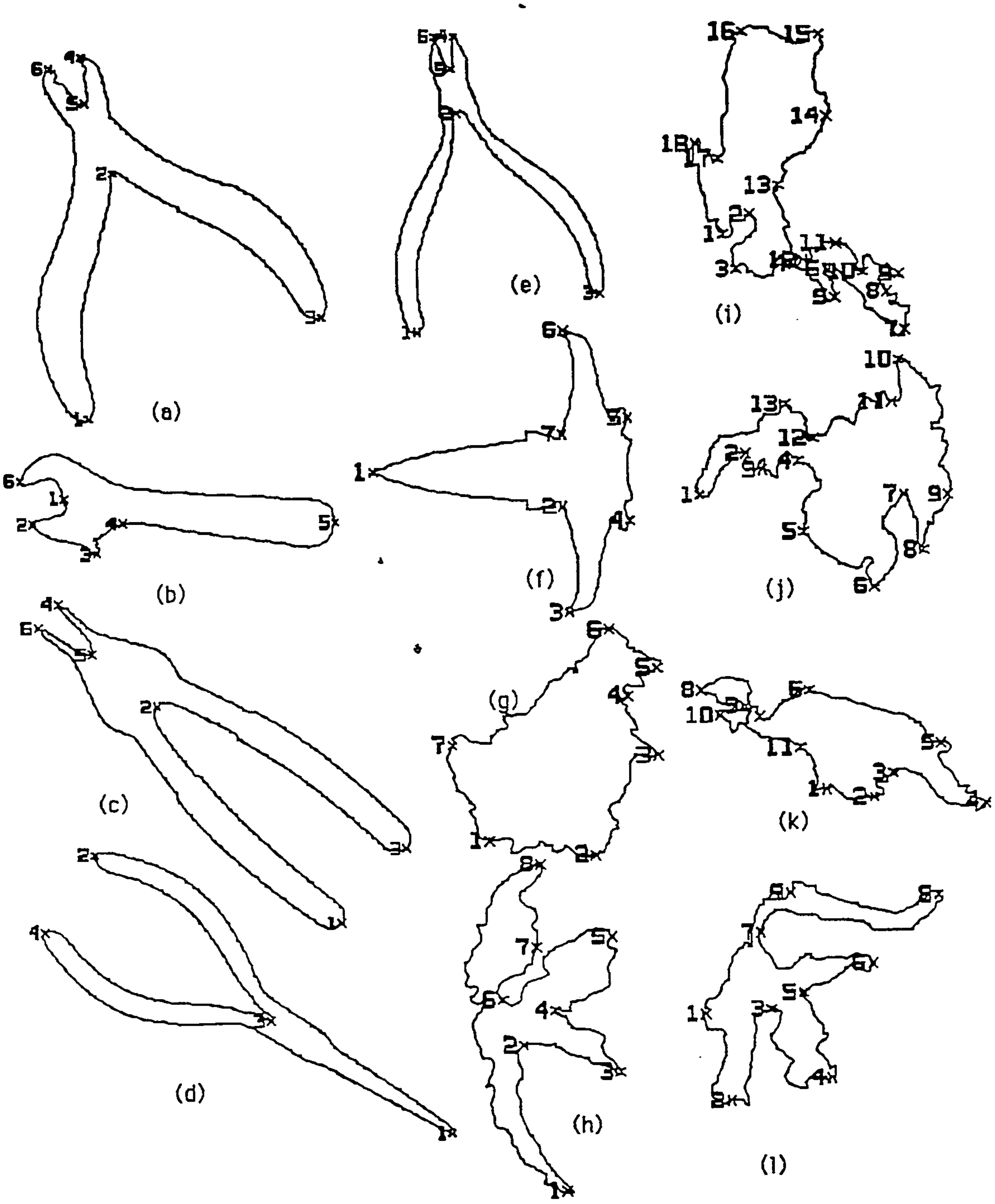


Figure 1.1: Landmarks of Various Objects: (a). Wire Stripper, (b). Wrench, (c). Specialty Plier, (d). Needle-Nose Plier, (e). Wire Cutter, (f). Spacecraft, (g) Island of Borneo, (h). Island of Halmahera, (i). Island of Luzon, (j). Island of Mindanao, (k). Island of New Guinea, (l). Island of Sulawesi.

frames are already given.

The study described here is to estimate 3-D motion and structure parameters of rigid planar objects, by first establishing point correspondences on the objects at different image frames. Hence, *probe and block* algorithm [3] is used to perform the matching task among the points on the objects, and 3-D rigid motion are estimated for the case either the motion is small[21] or large[23]. A shape measure, known as sphericity, derived from the affine transformation[8] is used to indicate the quality of match among dominant points. While, in estimating the motion parameters, a set of so called *eight-pure parameters* are defined and computed for both cases. Once we have determined these parameters, the motion parameters can be computed by solving a sixth order polynomial equation if the rotation is small, and decomposing the singular values of 3×3 real matrix of the pure parameters if the rotation is large.

From all numerical examples that have been tried, the number of solutions depends on the number of real roots of the sixth order polynomial equation, and only a relative depth of the object can be recovered when the motion is small. When the motion is large, the number of solution depends on the multiplicity of the singular values of the 3×3 real matrix of the pure parameters. If the multiplicity is two the motion parameters is unique aside from a scale factor for the translation parameters, otherwise the motion parameters are two if the singular values are all distinct. If there is no translation at all, the singular values are all identical and the motion parameters are unique.

In the remaining chapters, we will discuss the mathematical elements that supports the computation methods in Chapter 2. Point matching and motion estimation algorithm will be presented in Chapter 3. The computation procedures and experimental results will be given in Chapter 4. Finally, the conclusion of the results and the recommendation will be given in Chapter 5.

Chapter 2

Mathematical Elements

2.1 Introduction

In this chapter, we will discuss the mathematical elements that support the computation method presented in Chapter 3. In the following section a shape measure, known as sphericity, derived from affine transformation is briefly discussed. Detailed proofs and explanation can be found in [4] and [8]. Rigid motion which consists of translation and rotation is presented subsequently.

2.2 Affine Transformation and Sphericity

An affine transformation is a one-to-one mapping with the equation of the form :

$$P' = AP + t \tag{2.1}$$

where

$$P' = \begin{pmatrix} x' \\ y' \end{pmatrix};\ P = \begin{pmatrix} x \\ y \end{pmatrix};\ t = \begin{pmatrix} m \\ n \end{pmatrix}$$

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix},\ and\ det(A) \neq 0.$$

The following properties are preserved under the transformation[8] :

- The set of all transformation is a group.
- Collinearity and noncollinearity.
- The betweenness-relation.

Therefore, the resultant of two affine transformation is an affine transformation , the inverse of an affine transformation is an affine transformation; a point is mapped into a point, a line into a line; a mid-point of a line remains the mid-point of the transformed line.

A set of values $a, b, c, d, m,$ and n in Equation (2.1) can be uniquely determined if three noncollinear points P_1, P_2, P_3 and three noncollinear transformed points P_1', P_2', P_3' are given. Substituting these points in Equation (2.1) gives :

$$LX = B \tag{2.2}$$

where :

$$L = \begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ x_3 & y_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ 0 & 0 & 0 & x_3 & y_3 & 1 \end{pmatrix}$$

$$X = \begin{pmatrix} a \\ b \\ m \\ c \\ d \\ n \end{pmatrix}, B = \begin{pmatrix} x_1' \\ x_2' \\ x_3' \\ y_1' \\ y_2' \\ y_3' \end{pmatrix}.$$

Equation (2.2) is nonsingular, since the points are chosen to be non-collinear, i.e, the row vectors as well as the column are linearly independent, thus $det(L) \neq 0$ and L^{-1} exists.

The sphericity value (S) of an affine transformation given by Equation (2.1) is defined by[4] :

$$S = \sqrt{\frac{det(A^t A)}{(\frac{tr}{2}(A^t A))^2}} \tag{2.3}$$

where det() and tr() are the determinant and the trace of a matrix. Substituting A into Equation (2.3) gives[4]:

$$S = \frac{t_1^2 + t_3^2 - (t_2^2 + t_4^2)}{t_1^2 + t_2^2 + t_3^2 + t_4^2} \tag{2.4}$$

where : $t_1 = a + d$, $t_2 = a - d$, $t_3 = b - c$, and $t_4 = b - c$.

2.3 Motions

There are four types of motions : translation, rotation, reflection and shear. If only translation and rotation are taken collectively, it is called *rigid motion.*

Motion is also a one-to-one mapping and is a subset of the affine transformation. Thus it has the same properties as that of the affine transformation as follows[8] :

- The set of all motion is a group.

- Motions preserve collinearity and noncollinearity.

- Motions preserve the between relation.

The first property implies that the inverse of any motion is a motion, and the resultant of any two motions is also a motion. The second property implies that the image of a point is a point, a line is a line. Finally, the last describes the relation of three points, in which one is in between the others. It preserves the between relation, thus the image of a triangle, a segment is also a triangle, a segment, respectively.

Since the motions we are about to study are 3-D motion for rigid planar objects, only 3-D translations and rotations are considered here. The terms rotation(s) and translation(s), unless specified, always mean 3-D rotation(s) and translation(s). In the following subsection, we will discuss the first type of motions, translations.

2.3.1 Translations

A translation (T) is a transformation with the equation of the form :

$$x' = x + p \; ; \;\; y' = y + q \; ; \;\; z' = z + r \tag{2.5}$$

where p, q, and r are arbitrary constants. Some properties that follow under translations can be seen by investigating Equation (2.5). Substituting $x = x'$, $y = y'$ and $z = z'$ into Equation (2.5), it is readily seen that there is no fixed point under translations since the equation has no solution. Another property is that after translation each line has the same inclination as the original. To check this property, let :

$$l = ax + by + cz = 0 \tag{2.6}$$

be any plane with a, b, and c are not all zero. Combining Equation (2.5) and (2.6) gives :

$$l^{'} = ax^{'} + by^{'} + cz^{'} - ap - bq - cr = 0 \tag{2.7}$$

which has the same inclination as l, since the coefficient x, y, and z in Equation (2.6) are the same as those of $x^{'}, y^{'}$, and $z^{'}$ in Equation (2.7).

For the purpose of object representation that undergoes 3-D translations, it is more convenient to use vector notation. That is, for n points on the object Equation (2.5) can be rewritten as :

$$\begin{pmatrix} x_i^{'} \\ y_i^{'} \\ z_i^{'} \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} + \begin{pmatrix} p \\ q \\ r \end{pmatrix} \tag{2.8}$$

where $i = 1, 2, 3, ..., n$. Solving for $(p\ q\ r)^T$ yields :

$$\begin{pmatrix} p \\ q \\ r \end{pmatrix} = \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} = \begin{pmatrix} x_i^{'} - x_i \\ y_i^{'} - y_i \\ z_i^{'} - z_i \end{pmatrix} \tag{2.9}$$

Equation (2.5) can also be written in different form by using homogeneous coordinates[18] :

$$(X\ Y\ Z\ H) = (x\ y\ z\ 1) \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & q & r & 1 \end{pmatrix} \tag{2.10}$$

Multiplying right hand side, we have:

$$(X\ Y\ Z\ H) = ((x+p)\ (y+q)\ (z+r)\ 1) \tag{2.11}$$

The transformed point is then given by:

$$\begin{aligned} x' &= \frac{X}{H} = x + p \\ y' &= \frac{Y}{H} = y + q \\ z' &= \frac{z}{H} = z + r \end{aligned} \tag{2.12}$$

which is exactly the same as Equation (2.5).

It is obvious that to produce a 3-D translation of any general point O, the original coordinate (x, y, z) of the point and the amount of translation $(\Delta x,\ \Delta y,\ \Delta z)$ are needed to define T.

2.3.2 Rotations

A rotation about any point O in 2-D space is shown in figure 2.1. It is a mapping in which O is fixed, and any other point P of the object goes into the point P' such that $OP = OP'$ and $POP' = \theta$. O is called the center of rotation, and θ is the angle of rotation. If O is the origin of the 2-D Cartesian coordinate system, it is the 2-D rotation about the origin. In general, the 2-D rotation about an arbitrary point can be accomplished by first translating the center of rotation to the origin, performing the required rotation, and then translating the result back to the original center of rotation. Thus the center of rotation never changes.

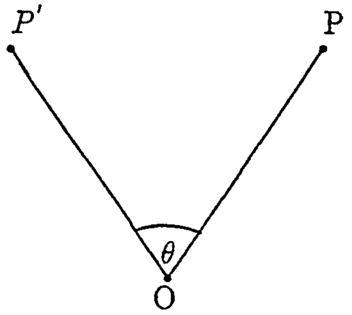


Figure 2.1: Rotation about O by θ

In 3-D space the rotation about origin is meaningless, because there are many directions can be taken by the moving object. Thus, we need to define an axis of rotation before the rotation is performed. The concept in 2-D above is also true for 3-D rotation case. That is, if the rotation occurs about the x axis of the Cartesian coordinate system, the x coordinates of the points on the object do not change likewise rotation about y and z. Figure 2.2 shows the 3-D rotations about x and y axes by -90^o and 90^o, respectively, from the original position. In a similar manner the rotation about arbitrary axis can be done by translating the object and the desired axis of rotation such that the rotation is done about an axis through the origin, and then translating the result back to the initial position. For example, if the axis of rotation is desired to pass through the point O =

(p, q, r), then the equation of the transformation matrix is given by[18] :

$$(X\,Y\,Z\,H) = \begin{pmatrix} x & y & z & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -p & -q & -r & 1 \end{pmatrix} \begin{pmatrix} R \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & q & r & 1 \end{pmatrix} \tag{2.13}$$

R is the rotation matrix (orthonormal), neglecting the fourth row and column, given by [18] :

$$R = \begin{pmatrix} n_1^2 + (1 - n_1^2)cos\theta & n_1 n_2(1 - cos\theta) + n_3 sin\theta & n_1 n_3(1 - cos\theta) + n_1 sin\theta \\ n_1 n_2(1 - cos\theta) - n_3 sin\theta & n_2^2 + (1 - n_2^2)cos\theta & n_2 n_3(1 - cos\theta) + n_1 sin\theta \\ n_1 n_3(1 - cos\theta) + n_2 sin\theta & n_2 n_3(1 - cos\theta) - n_1 sin\theta & n_3^2 + (1 - n_3^2)cos\theta \end{pmatrix} \tag{2.14}$$

where n_1, n_2, and n_3, as shown in Figure 2.3, are the direction cosine of the rotation axis in x, y, and z directions respectively, and θ is the amount of rotation.

If the rotation is small ($\theta \approx sin\theta$), Equation (2.14) becomes :

$$R = \begin{pmatrix} 1 & n_3 sin\theta & -n_2 sin\theta \\ -n_3 sin\theta & 1 & n_1 sin\theta \\ n_2 sin\theta & -n_1 sin\theta & 1 \end{pmatrix} \tag{2.15}$$

If the rotation occurs about one of the orthogonal axes of the coordinate system, then two of the direction cosine are equal to zero. For example, rotation about the x axis by 0^o : $n_1 = 1$, $n_2 = 0$, and $n_3 = 0$, Equation (2.14) becomes:

$$R_{x,\theta} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\theta & sin\theta \\ 0 & -sin\theta & cos\theta \end{pmatrix} \tag{2.16}$$

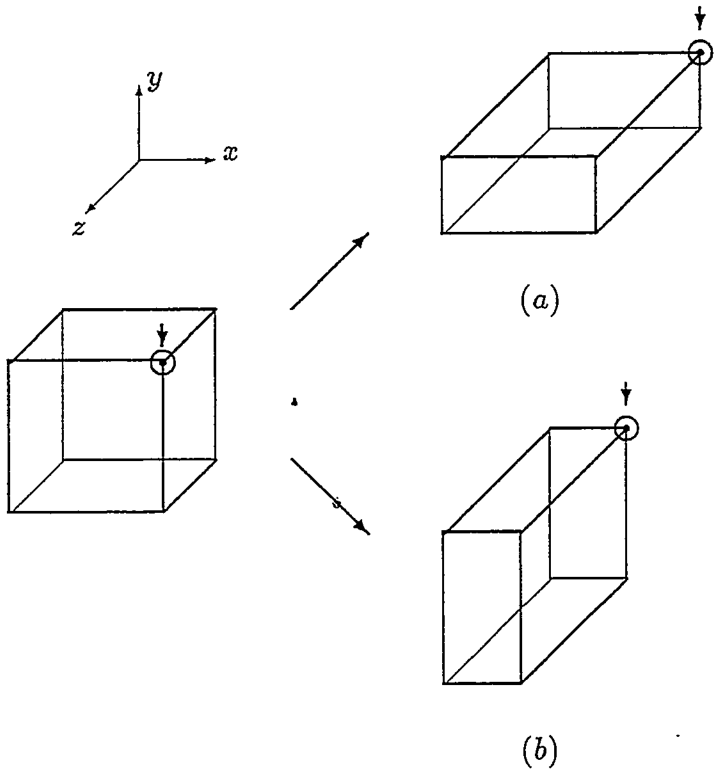


Figure 2.2: Rotation about $x - axis$ (a) and $y - axis$ (b) by -90^o and 90^o

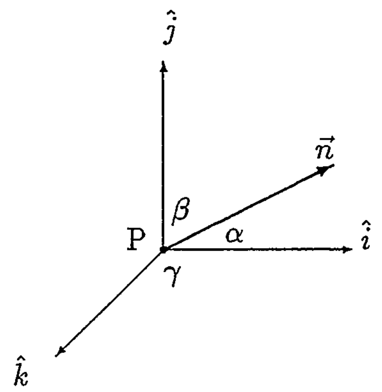


Figure 2.3: Direction cosine

In a similar manner, we have :

$$R_{y,\theta} = \begin{pmatrix} cos\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{pmatrix} \tag{2.17}$$

and,

$$R_{z,\theta} = \begin{pmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.18}$$

It is obvious that in order to define R by using Equation (2.14), we need to know ahead of time the (x, y, z) coordinates of the point to be rotated, the amount of rotation θ and the direction cosine (n_1, n_2, n_3) of the rotation axis. For any general point $\vec{P} = p_1\hat{\imath} + p_2\hat{\jmath} + p_3\hat{k}$ that lies on the rotation axis, the direction cosine are computed by :

$$n_1 = \frac{p_1}{|\vec{P}|} = cos\alpha$$

$$n_2 = \frac{p_2}{\left|\vec{P}\right|} = cos\beta \tag{2.19}$$

$$n_3 = \frac{p_3}{\left|\vec{P}\right|} = cos\gamma$$

where : $\left|\vec{P}\right| = \sqrt{p_1^2 + p_2^2 + p_3^2}$

2.3.3 Rigid Motions

As mentioned in the previous section, rigid body motion can be expressed as a combination of translations and rotations. It can be either a rotation(s) by an angle θ around an axis with directional cosine n_1, n_2, n_3, followed by a translation(s) or vice-versa. In matrix form, it has the equation of the form :

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} R \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} T \end{pmatrix} \tag{2.20}$$

where R is an orthonormal rotation matrix (first type, i.e, $det(R) = 1$), and T is a translation vector given as in the Equation (2.14) and (2.9), respectively.

Chapter 3

Estimating Motion Parameters from Image Sequences

3.1 Literature Review

In the past, most research work on motion estimation has been concerned with 2-D motion only. This is because the interpretation of moving object images in 3-D is much more complicated than 2-D motion since rotation and translation in depth are difficult to be analyzed. In addition, as a result of rotation in space, parts of moving object can disappear from view. Recently, methods of estimating 3-D motion parameters, including 3-D rotations, of rigid bodies with different kind of shape have been investigated.

There are two ways in determining the movement of objects from a sequence of images[11], *a direct estimation* and *two stage estimation.* The direct estimation is based on the time-difference image. That is, the displacement vector of points in an image is constrained by the spatial and temporal gradients of the intensity distribution of the image, and is expressed as a function of several motion parameters. These parameters can be directly estimated by solving a system of equations.

In the two stage estimation, the interframe correspondences, of the apparent motion of the image is computed first. Then its 3-D motion parameters of the moving object in the scene are estimated based on those correspondences. Therefore, from the viewpoint of instantaneous motion analysis , the direct estimation algorithm is relatively simpler than the two stage estimation, since it requires only the spatial gradients of the image, and the solution of a set of system equations.

Some Examples of previous research work using direct estimation for 2-D case are the studies of Limb and Murphy[15][16], and Cafforio and Rocca[7]. Both studies investigated the movement of object in television images, and can be considered as parallel translations. Subsequently, Huang and Tsai[11] have extended the idea to 3-D motion algorithm where the displacement vector can be represented by the use of affine transformation. The same idea as Huang's for 2-D case has also been investigated by Schalkoff and Mc. Vey[19]. Their model was intended for a class of video targets.

Although the direct estimation has the above advantage, it has the two following disadvantages compared with the two stage method. *First*, it is very difficult to get the spatial and temporal gradients with high accuracy. *Second*, the direct method needs one more constraint associated with the depth information of the moving object in order to estimate the motion parameters in the 3-D case.

A number of research have been done to obtain a depth information from the moving object. Yamamoto[27] has used some range-finder such as binocular vision, and derived a linear equation and its solution to estimate 3-D motion parameters using the direct method. Horn and Weldon[10] have derived an algorithm for recovering translational motion where a part of motion parameter related to the depth is given. On the other hand, Tsai and Huang[21][23] imposed the surface of the object as a collection of planar patches, and examined its 3-D motion parameters and a relative depth of each patch.

Another problem that had been seriously discussed by many researchers in estimating the 3-D motion parameters was the question of the uniqueness of the number of solutions. Regardless of the method is used, the uniqueness of the estimated motion parameters is, of course, independent and depends only on the motion characteristic of the moving object under investigation. In the direct estimation, for example, the uniqueness of motion interpretation can be algebraically determined by examining the solution of a set of system equations. Yamamoto[27] considered a general aperture problem for direct estimation, and derived the necessary and sufficient conditions for the structure and surface pattern of the object such that 3-D motion can be uniquely determined. According to his discussion, it was shown that 3-D motion parameters can not be uniquely determined for only eight kinds of objects with special geometrical structure and surface pattern. For example, we can not uniquely determine 3-D motion parameters of the object patterns such as barber's pole and a simple bucket.

In the two stage approach, in contrast, the uniqueness can only be examined when the first stage is already done, i.e, after the correspondences among points in the image frames are known. For example, when structure and surface patterns of the object can be of any form, Ullman on his classical book on visual motion[25] showed that at least four point correspondences over three image frames to uniquely determine 3-D motion and structure (plus a reflection) of the four point rigid configurations. He assumed that the correspondences among points are Balready given, and proposed the requirement that those points must be noncoplanar, otherwise the motion parameters can not be uniquely determined, that is, it has infinitely many solutions. Huang and Lee[12], derived a linear algorithm for three and four point correspondences over three image frames. They also showed the same results as Ullman's that the four point correspondences over three frames ensure a unique solution to motion and structure (plus a reflection) only if the points under observation are noncoplanar. For three point correspondences their algorithm yields sixteen solutions to motion and four solutions to structure (two plus their reflections). If only two frames is used, they pointed out that no matter how large the point correspondences the motion parameters allow an infinite number of solutions.

For the case where the surface pattern is restricted to special form, there also have been many results where the two stage method is discussed and applied to actual motion. For example, when the surface pattern is a curve, Huang and Tsai[22] pointed out that seven point correspondences over two image frames are sufficient to uniquely determine the 3-D motion parameters, and those points should not be traversed by two planes with one plane containing the origin, nor by a cone containing the origin. Another result of their algorithm introduced a set of parameters, called *essential parameters*, derived from the eight image point correspondences to determine motion parameters up to a scale factor for the translations.

In the planar surface, some researchers such as Tsai, Huang and Zhu[23][24], Longuet and Higgins[17], Subbarao and Waxman [20] worked with different objective but they obtained the same result. That is, if only two image frames are considered there at most two solutions for the motion parameters. The number of solution can be reduced to be unique if one more image frame is included.

Many results in motion analysis using the two stage method have been achieved within the course of time. However, no one of them has considered both problem of correspondences task and estimation of its motion parameters at the same time, that is, all of the above works have assumed that the point correspondences between two or more image frames are already given. The study described here considers both problems and uses[3] to perform the correspondences problem and[21][23] to determine its motion parameters.

3.2 Direct vs Two Stage Estimation

In this section, the direct and two stage estimation algorithm which are taken from [27] and [21], respectively, will be presented. The purpose of this section is to give an illustration about the review presented above. The correspondences problem, probe and block algorithm[3], will be presented right after. Finally, this chapter will be concluded by the motion algorithm for a rigid planar patch.

3.2.1 Direct Estimation

Let (x,yB,z) be the object space coordinate, xy plane be the image of the camera and z axis be the optical axis. In [27], the object is orthogonally projected on the xy plane, and the image sequence is a function of time. Let

$E(x, y, t)$= the intensity of the image at point (x,y) and time t.

$(E_x, E_y) = (\frac{\partial E}{\partial x}, \frac{\partial E}{\partial y})$ = the spatial gradient of the intensity.

$E_t = \frac{\partial E}{\partial t}$= the temporal gradient of the intensity.

Then from the total differential equation of E with respect to time t, we have :

$$dE(x(t), y(t), t) = 0 \tag{3.1}$$

and

$$E_x u + E_y v + E_t = 0 \tag{3.2}$$

where :

$(u, v) = (\frac{dx}{dt}, \frac{dy}{dt})$= the optical flow (orthogonal projection assumption).

The velocity vector $V_p = (u, v, w)$ of the point P on the rigid object is given by :

$$V_p = \Omega \times r + V_o \tag{3.3}$$

where :

$r = (x, y, z)$ = vector position.

$\Omega = (\omega_x, \omega_y, \omega_z)$ = the angular velocity.

$V_o = (u_o, v_o, w_o)$= velocity vector at the origin.

Substituting Equation (3.3) into (3.2), we get 3-D motion equation as follows :

$$E_x u_o + E_y v_o - zE_y \omega_x + zE_x \omega_y + (xE_y - yE_x)\omega_z + E_t = 0 \tag{3.4}$$

which is linear in the five unknowns $u_o, v_o, \omega_x, \omega_y$, and ω_z. Those unknowns can be determined only if the depth z is known. If the motion is restricted to the plane perpendicular to the optical axis, $\omega_x = \omega_y = 0$, 2-D motion can then be obtained from 3-D motion, i.e :

$$E_x u_o + E_y v_o + (xE_y - yEx)\omega_z + E_t = 0 \tag{3.5}$$

If only translation is considered, $\omega_z = 0$, 1-D motion equation is then given by :

$$E_x u_o + E_y v_o + E_t = 0 \tag{3.6}$$

For n points on the image (where n is greater than the number of unknowns), then we have an overdetermined system of linear equations, in matrix form :

$$Ax = b \tag{3.7}$$

where (for 3-D case) :

$$A = \begin{pmatrix} E_{x_1} & E_{y_1} & E_{y_1} & E_{x_1} & k_1 \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ E_{x_n} & E_{y_n} & E_{y_n} & E_{x_n} & k_n \end{pmatrix}, \; k_i = (x_i E_{y_i} - y_i E_{x_i})$$

$$x = \begin{pmatrix} u_o \\ v_o \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}, \; b = \begin{pmatrix} -E_{t_1} \\ -E_{t_2} \\ -E_{t_3} \\ -E_{t_4} \\ -E_{t_5} \end{pmatrix}, \; i = 1, 2, ...5.$$

Since Equation (3.7) is a singular system, i.e, two of the column vectors are linearly dependent, also we have more equations than unknowns, we normally do not expect a solution, x, of $Ax = b$ to exist. Thus, we can reformulate the problem to seek for the vector x^* that somehow minimizes the vector expression $(Ax^* - b)$. That is, find x^* such that $\|Ax^* - b\|$ is minimized for some vector norm $\|.\|$. In this case $\|.\|$ denote the l_2 norm. Multiplying both side with A^T, where the superscript T stands for the transpose, we have :

$$A^T A x^* = A^T b \tag{3.8}$$

The least square solution, x^*, may be obtained by Gauss elimination or Gauss-Siedel, and can be expressed as :

$$x^* = \left(A^T A\right)^{-1} A^T b \tag{3.9}$$

where $(A^T A)^{-1}$ exists, i.e, $det\left(A^T A\right) \neq 0$. If $(A^T A)^{-1}$ does not exist, it is impossible to determine the solution uniquely. This is the case where the motion can not be uniquely determined from the image sequence. However, since the coefficient matrix of $A^T A$ is a function of the spatial gradient of the intensity, the uniqueness of the motion parameters depends on the geometrical property, aperture problem, of the image pattern. Therefore by examining the geometrical property of the image pattern, we may hope to obtain a solution(s) for the motion parameters. Details explanation about this problem is given in[27].

3.2.2 Two Stage Estimation

First of all the basic motion equation for rigid planar object given in Equation (2.20) is rewritten as :

$$
\begin{aligned}
x' &= r_{11}x + r_{12}y + r_{13}z + \Delta x \\
y' &= r_{21}x + r_{22}y + r_{23}z + \Delta y \\
z' &= r_{31}x + r_{32}y + r_{33}z + \Delta z
\end{aligned}
\tag{3.10}
$$

where r_{i_j} $(i = 1,2,3; j = 1,2,3)$ is the element in the ith row and jth column of R as given in Equation (2.14). Let:

(x, y, z) = the object-space coordinates of a point P at first view

(x', y', z') = the object-space coordinates of P at second view

(X, Y) = the image-space coordinates of P at first view

(X', Y') = the image-space coordinates of P at second view

Then from the basic geometry shown in Figure 3.1, we obtain :

$$X = \frac{x}{z}F\,,\ Y = \frac{y}{z}F\ ,\ X' = \frac{x'}{z'}F\,,\ Y' = \frac{y'}{z'}F \tag{3.11}$$

where F is the focal length. Let the equation of a rigid planar patch be :

$$ax + by + cz = 1 \tag{3.12}$$

Substituting Equation (3.11) into (3.12) and solving for z, gives :

$$z = \frac{F}{aX + bY + cF} \tag{3.13}$$

For simplicity we select $F = 1$, combining Equation (3.10),(3.11) and (3.13) all together we have :

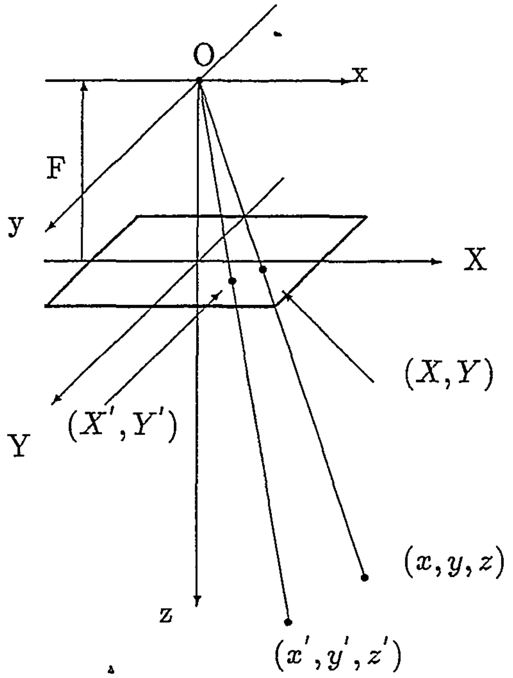


Figure 3.1: Basic Geometry for Two Stage 3-D Motion Estimation

$$X' = \frac{a_1X + a_2Y + a_3}{a_7X + a_8Y + 1}, \; Y' = \frac{a_4X + a_5Y + a_6}{a_7X + a_8Y + 1} \tag{3.14}$$

where :

$$\begin{aligned} a_1 &= \frac{r_{11} + a\Delta x}{r_{33} + c\Delta z}, \; a_2 = \frac{r_{12} + b\Delta x}{r_{33} + c\Delta z} \\ a_3 &= \frac{r_{13} + c\Delta x}{r_{33} + c\Delta z}, \; a_4 = \frac{r_{21} + a\Delta y}{r_{33} + c\Delta z} \\ a_5 &= \frac{r_{22} + b\Delta y}{r_{33} + c\Delta z}, \; a_6 = \frac{r_{23} + c\Delta y}{r_{33} + c\Delta z} \\ a_7 &= \frac{r_{31} + a\Delta z}{r_{33} + c\Delta z}, \; a_8 = \frac{r_{32} + b\Delta z}{r_{33} + c\Delta z} \end{aligned} \tag{3.15}$$

Equation (3.14) defines a mapping from a point P = (X,Y) to $P' = (X',Y')$. Thus, if four or more corresponding point pairs are given (or obtained from the first stage of this algorithm), the parameters $a_i's$, known as the pure parameters, can be determined by solving a set of linear equations. Once we have determined the $a'_i s$, the actual motion parameters $r'_{ij}s, \Delta x, \Delta y, \Delta z, a, b$ and c can be computed by using Equation (3.15). It is to be noted that the mapping from the space (X,Y) onto the space (X', Y') in Equation (3-14) is unique given the pure parameters $a'_i s$. Detail proof using a Lie Group of Transformation is discussed in[21].

3.3 Probe and Block Matching Algorithm

Let $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ and $(x'_1, y'_1, (x'_2, y'_2), ...(x'_m, y'_m)$ be the coordinates of a sequence of landmarks in the image spaces at first and second view, where n is the number of landmarks on the object at first view and m is the number of landmarks of the same object at second view. Those landmarks are obtained by tracing sequentially along the object boundary, and projecting them from 3-D space to 2-D images.

	1	2	3	4	5	6	7	8	9
1	-0.05	-0.04	-0.28	0.85	0.12	0.14	0.81	-0.06	0.03
2	0.23	0.04	0.99	-0.40	-0.62	-0.61	-0.48	0.25	-0.04
3	-0.07	-0.06	-0.45	0.99	0.21	0.24	0.99	-0.10	0.05
4	-0.09	-0.04	-0.52	0.21	0.99	0.98	0.24	-0.09	0.05
5	-0.08	-0.04	-0.49	0.21	0.96	0.99	0.23	-0.08	0.05
6	-0.08	-0.05	-0.46	0.97	0.21	0.23	0.99	-0.10	0.04
7	0.07	0.14	0.52	-0.46	-0.52	-0.62	-0.48	0.08	-0.14

Table 3.1: Compatibility Table

	1	2	3	4	5	6	7	8	9
1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.08
3	0.08	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
4	1.00	1.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00
5	1.00	1.00	1.00	1.00	1.00	0.16	1.00	1.00	1.00
6	1.00	1.00	1.00	1.00	1.00	1.00	0.21	1.00	1.00
7	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.19	1.00

Table 3.2: Diagonal Support Table

Consider a simple example of a moving aircraft as shown in Figure 3.2. The aircraft is rotated about arbitrary axis by 10° with the direction cosine $(\cos 45, \cos 60, \cos 120)$ and then translated by $(10, 20, 15)$ from the original position. First of all, a table of compatability between landmarks in the first and second view, as shown in Table 3.1, is constructed. Hence, we have $n = 7$ and $m = 9$.

The row index i corresponds to the landmarks at the first view while the column index j corresponds to the landmarks at the second view. The (i,j) entry of the table is the sphericity value of the affine transformation mapping the ith and two adjacent landmarks to the jth and two adjacent landmarks. Thus the matched landmarks correspond to the sequences of the entries in the table that are diagonal to each other which have values close to each other and should include the two adjacent diagonal entries. The flowchart of the probe and block matching algorithm is shown in Figure (3.3). By using the fact that one landmark in the first view can not match to more then one landmark in the second view, the probe and block algorithm is summarized as follows :

- Determine the first support entry. The (i, j) entry is said to be the support entry if the sample variance in entries $(i-2, j-2), (i-1, j-1), (i, j), (i+1, j+1)$, and $(i+2, j+2)$ given by :

  $$v(x_i) = \sum_{i=1}^{N} \frac{x_i - \bar{m}}{N} \tag{3.16}$$

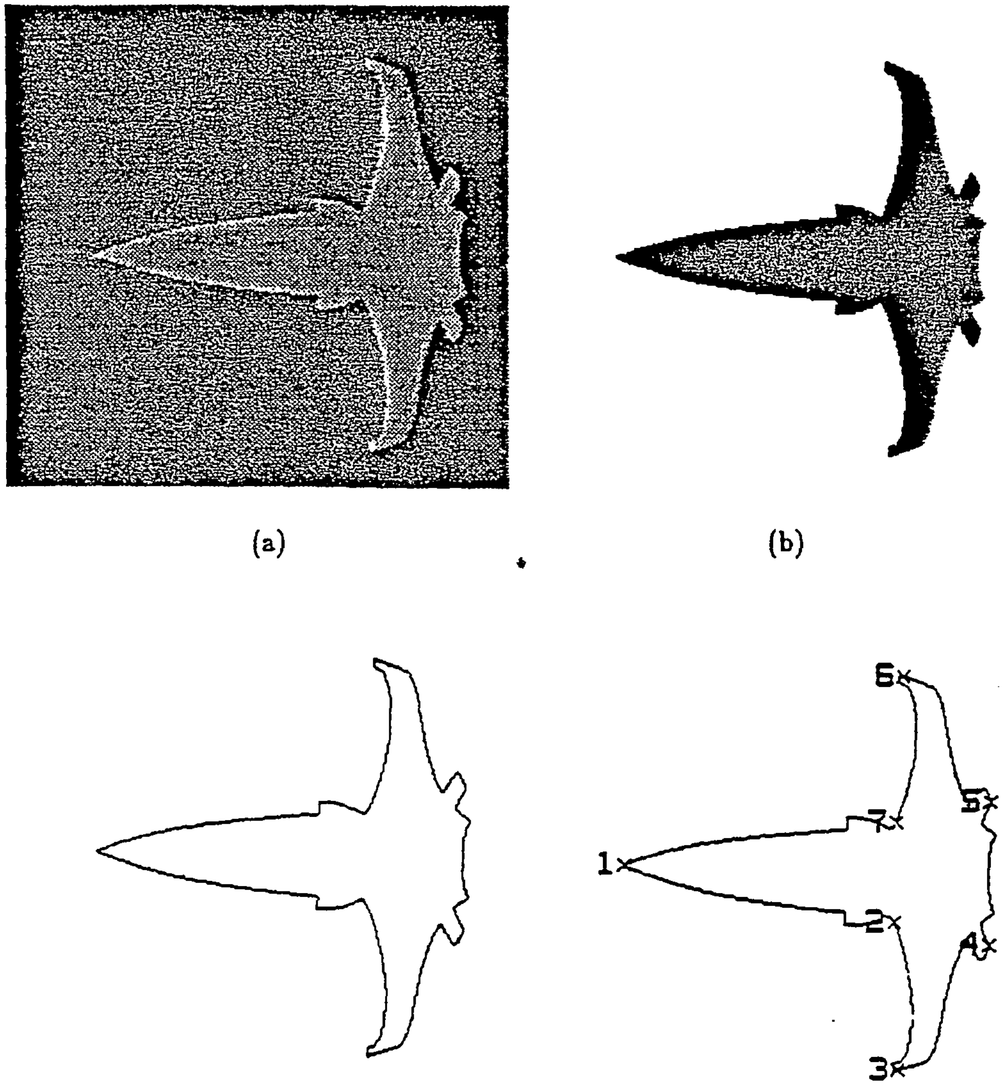
  is *minimum* and below a *threshold*, where $\bar{m}$ is the mean of those entries. If the entries are not of the same sign, i.e, the mapping is not of the same sense, then sample variance is set to 1, as shown in Table 3.2. In the example shown in Table 3.1, the first support entry is $(4, 5)$. If no such entry can be found, the procedure stops.

- Compute the mean of the diagonal entries that are used to determine the first support entry mentioned in Step 1. If its value in the diagonal is close to the mean, that entry is considered as a possible match. In the example, entries (2,3), (3,4), (4,5), (5,6) and (6,7) are close to the mean mentioned above and considered as possible matches. Note that diagonal entries that are considered as possible matches must be consecutive and diagonal to each other, and because of the sequential order of the landmarks we are matching, the diagonal can wrap around the table.

- Determine the possible matches of two adjacent diagonal entries. In the example entries $(1,2)$ and $(7,8)$ which is diagonal to entry $(4,5)$ are two adjacent entries mentioned above.

- After determining the diagonal entries that are considered as possible matches from Step 2, block the region from further search. For example, if diagonal entries $(2,3),(3,4),...,(7,8)$ are considered as possible matches, regions $(1 \leq i \leq n, 3 \leq j \leq 8)$ and $(2 \leq i \leq 7, 1 \leq j \leq n)$ are block from further search of matches, where n and m are the number of landmarks at the first and second view, respectively. If no diagonal entries can be considered as possible matches, block the support entry from further consideration. This will avoid the infinite looping. In the example shown in Table 3.1, the whole region is blocked.
- Look for another support entry in the regions that have not been blocked. If no such support entry can be found, the procedures stop, otherwise, return to Step 2. In the example shown in Table 3.1, no further support entry is found because the whole region is already blocked.

After determining the possible matches between landmarks at the first and second view, we will next verify the results in the least square sense. Since we use affine transformation to map landmarks from the first to second image, it's least square error should also be derived from this affine transformation. In the least square sense the coefficient of the affine transformation is given by [3]:



(a) (b)

(c) (d)

Figure 3.2: Image (a), Sihouette (b), Contour (c) and Landmarks (×) of an Aircraft

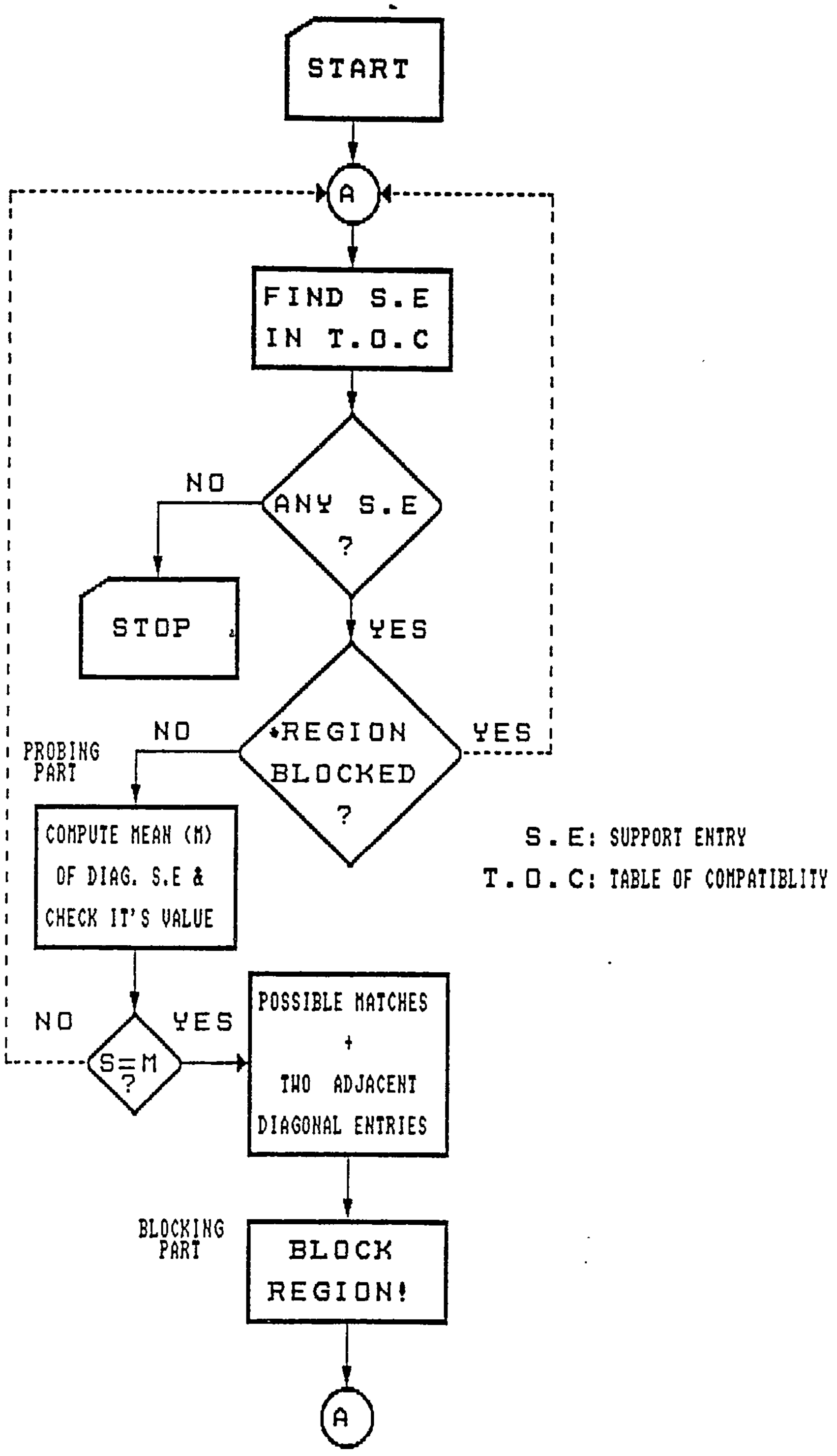


Figure 3.3: The Flowchart of Probe And Block Matching Algorithm

$$\begin{pmatrix} a \\ b \\ m \end{pmatrix} = M^{-1} \begin{pmatrix} \Sigma x_i x_i^{'} \\ \Sigma y_i x_i^{'} \\ \Sigma x_i^{'} \end{pmatrix}, \begin{pmatrix} c \\ d \\ n \end{pmatrix} = M^{-1} \begin{pmatrix} \Sigma x_i y_i^{'} \\ \Sigma y_i y_i^{'} \\ \Sigma y_i^{'} \end{pmatrix} \qquad (3.17)$$

where:

$$M = \begin{pmatrix} \Sigma x_i^2 & \Sigma x_i y_i & \Sigma x_i \\ \Sigma x_i y_i & \Sigma y_i^2 & \Sigma y_i \\ \Sigma x_i & \Sigma y_i & k \end{pmatrix}$$

(Σ goes from $i = 1$ to $i = k$).

The above least square error does not account for the overall goodness, i.e., it only indicates how well a portion of landmarks in first view match to the corresponding landmarks in second view. We need therefore to account for the overall goodness of match as follows :

$$\varepsilon^{'} = [1.0 + \frac{n-2}{k-2} log_2(\frac{n-2}{k-2})]\bar{\varepsilon} \qquad (3.18)$$

where :

$\varepsilon^{'}$= match error (overall goodness of match)

n= the total number of landmarks on first image ($n \geq k$)

k= the total number of matched landmarks ($k \geq 3$)

$\bar{\varepsilon} = \varepsilon/(k.scalefactor)$ = the normalized square error.

ε= unormalized square error.

In the earlier example, $k = n = 7$ and $\bar{\varepsilon} = 0.13$, therefore $\varepsilon^{'} = \bar{\varepsilon} = 0.13$. If $0 \leq k < 3$, $\varepsilon^{'} = \infty$.

3.4 Estimating 3-D Motion of a Rigid Planar Patch

3.4.1 Small Rotation

The basic geometry shown in Figure 3.1 is repeated here. It was shown in Section 3.2 that the image space coordinates of a rigid planar patch before and after undergoing 3-D motion are related by Equation 3.14. Hence, the parameters $a_i's$ in Equation (3.15) are a function of the motion parameters and applicable for any kind of 3-D rigid planar motion. For the case where rotation is small, the elements of the rotation matrix in Equation (2.14) and in fact Equation (3.10) are replaced by those in Equation (2.15), then the $a_i's$ have the equation of the forms :

$$\begin{aligned}
a_1 &= \frac{1 + a\Delta x}{1 + c\Delta z} \ , \ a_2 = \frac{-n_3 \sin\theta + b\Delta x}{1 + c\Delta z} \\
a_3 &= \frac{n_2 \sin\theta + c\Delta z}{1 + c\Delta z} \ , \ a_4 = \frac{n_3 \sin\theta + a\Delta y}{1 + c\Delta z} \\
a_5 &= \frac{1 + b\Delta y}{1 + c\Delta z} \ , \ a_6 = \frac{-n_1 \sin\theta + c\Delta y}{1 + c\Delta z} \\
a_7 &= \frac{-n_2 \sin\theta + a\Delta z}{1 + c\Delta z} \ , \ a_8 = \frac{n_1 \sin\theta + b\Delta z}{1 + c\Delta z}
\end{aligned} \tag{3.19}$$

From Equation (3.19) it is obvious that Δz can never be determined, and we can hope to determine the $z_i's$ to only within a relative depth. We therefore let

$$a^* = a\Delta z \ , \ b^* = b\Delta z \ , \ c^* = c\Delta z, \Delta x^* = \frac{\Delta x}{\Delta z}, \Delta y^* = \frac{\Delta y}{\Delta z}, \tag{3.20}$$

the motion parameters now becomes : $n_1, n_2, n_3, \theta, \Delta x^*, \Delta y^*, a^*, b^*$ and c^*, which are larger than the number of equations, i.e., eight. To solve Equation (3.19) we need one more constraint which comes from the fact that the square root of the sum of the squared direction cosine is always equal to one, another constraint is that for small rotation $\theta = \sin\theta$. Considering these parameters into Equation (3.19), the unknown motion parameters now become: $\Phi_1, \Phi_2, \Phi_3, \Delta x^*, \Delta y^*, a^*, b^*$ and c^* where $\Phi_i = n_i\theta$. These parameters can be computed by manipulating Equation (3.19) into a form:

$$P_6\Delta x^{*6} + P_5\Delta x^{*5} + P_4\Delta x^{*4} + P_3\Delta x^{*3} + P_2\Delta x^{*2} + P_1\Delta x^* + P_o = 0 \quad (3.21)$$

where :

$P_6 = WM^2 - N(SM - NU)$

$P_5 = M(S^2 + N^2 + M^2 - 4UW)$

$P_4 = -M^2(W + 5U) + U(2N^2 - S^2) - 3SMN + 4UW$

$P_3 = 2M(S^2 + N^2 - M^2 + 4U^2)$

$P_2 = (-M^2 + 4U^2)W + (6M^2 - 4U^2 - 2S^2 + N^2)U - 3SMN$

$P_1 = M(S^2 + N^2 + M^2 - 4U^2 + 4UW)$

$P_o = M^2(a_5 - a_1) - S(SU + SM)$ and

$S = a_2 + Ba_4$

$M = a_3 + a_7$

$N = a_6 + a_8$

$U = a_1 - 1$

$W = a_5 - 1$ and, furthermore,

$$
\begin{aligned}
\Delta y^* &= \frac{-N\Delta x^{*3} + S\Delta x^{*2} - N\Delta x^* + S}{-M\Delta x^{*2} + 2U\Delta x* + M} \\
c^* &= \frac{N\Delta y^* - W}{\Delta y^{*2} - N\Delta y^* + a_5} \\
b^* &= c^*(N - \Delta y^*) + N \\
a^* &= c^*(M - \Delta x^*) + M \\
\Phi_1 &= (c^* + 1)a_6 - \Delta y^* c^* \\
\Phi_2 &= -(c^* + 1)a_3 + \Delta x^* c^* \\
\Phi_3 &= -\Delta x^* b^* + a_2(c^* + 1)
\end{aligned}
\tag{3.22}
$$

In order to solve the unknown motion parameters, we first solve Equation (3.21) for Δx^* which is a sixth order polynomial equation, then the others are obtained from Equation (3.22). Note that the polynomial can have potentially six real roots which may give six solutions for the motion parameters. The actual roots are those that satisfy both Equation (3.21) and (3.22).

3.4.2 Large Rotation

When rotation is large, the rotation matrix is given by Equation (2.14). Following exactly the same procedures as in Subsection (3.2.2), the unique mapping given by Equation (3.14) is again valid if the pure parameters are replaced by :

$$a_1 = [n_1^2 + (1 - n_1^2)\cos\theta + a\Delta x]/k$$

$$a_2 = [n_1 n_2(1 - \cos\theta) - n_3 \sin\theta + b\Delta x]/k$$

$$a_3 = [n_1 n_3(1 - \cos\theta) + n_2 \sin\theta + c\Delta x]/k$$

$$a_4 = [n_1 n_2(1 - \cos\theta) + n_3 \sin\theta + a\Delta y]/k$$

$$a_5 = [n_2^2 + (1 - n_2^2)\cos\theta + b\Delta y]/k$$

$$a_6 = n_2 n_3[(1 - \cos\theta) + n_1 \sin\theta + c\Delta y]k$$

$$a_7 = n_1 n_3[(1 - \cos\theta) - n_2 \sin\theta + a\Delta z]/k$$

$$a_8 = [n_2 n_3(1 - \cos\theta) + n_1 \sin\theta + b\Delta z]/k \tag{3.23}$$

where $k = n_3^2 + (1 - n_3^2)\cos\theta + c\Delta z$.

We shall look for the number of possible solutions for the motion parameters by decomposing the following matrix A that consists of the eight pure parameters $a_i's$ as follows :

$$A = \begin{pmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & 1 \end{pmatrix} \tag{3.24}$$

The singular value decomposition of A is given by :

$$A = U\Lambda V^T \tag{3.25}$$

where Λ is a 3×3 diagonal matrix of the singular values ($\lambda_i's, i = 1, 2, 3$) of A, and U,V are the 3×3 orthonormal matrices. It can be shown that :

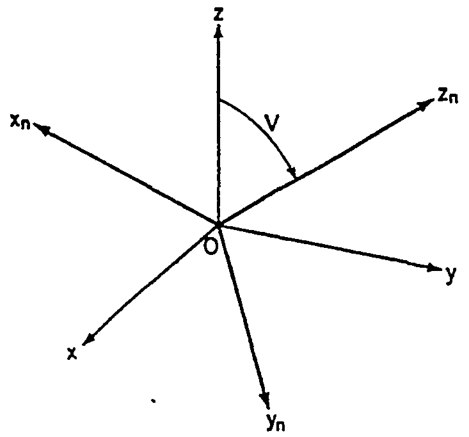


Figure 3.4: Transformation of Coordinate System with Matrix V

$$kA = R + \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} (a \;\; b \;\; c) \tag{3.26}$$

and from Equation (2.20) and (3.12) :

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = kA \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{3.27}$$

If the original coordinate systems at time t_1 and t_2, (x, y, z) and (x', y', z'), respectively, are transformed by the orthonormal matrix V in Equation (3.25), as shown in Figure 3.4, then we have a relation :

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = V \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix} , \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = V \begin{pmatrix} x'_n \\ y'_n \\ z'_n \end{pmatrix} \tag{3.28}$$

where (x_n, y_n, z_n) and (x'_n, y'_n, z'_n) are the transformed coordinate systems.

Substituting Equation (3.26) into (3.25) gives :

$$V \begin{pmatrix} x_n' \\ y_n' \\ z_n' \end{pmatrix} = kAV \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix} \tag{3.29}$$

Taking the norms of the vectors on both sides of Equation (3.29), we have :

$$\begin{pmatrix} x_n' \, y_n' \, z_n' \end{pmatrix} \; V^T V \begin{pmatrix} x_n' \\ y_n' \\ z_n' \end{pmatrix} = k^2 \left(x_n \, y_n \, z_n \right) V^T A^T A V \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix} \tag{3.30}$$

Substituting Equation (3.25) to (3.30) and replacing the expression $V^T V$ and $U^T U$ by an identity matrix (U and V are both orthonormal), we obtain:

$$x_n'^2 + y_n'^2 + z_n'^2 = k^2 \left(\lambda_1^2 x_n^2 + \lambda_2^2 y_n^2 + \lambda_3^2 z_n^2 \right) \tag{3.31}$$

Equation (3.31) defines a relationship between the coordinates before and after transformation associated with the singular values of matrix A. Since the left hand side of Equation (3.31) defines a sphere in the (x_n', y_n', z_n') space, while the right hand side defines an ellipsoid in the (x_n, y_n, z_n) space, the uniqueness of the motion parameters given the pure parameters $a_i's$ depends on the geometrical motion characteristics, i.e., the multiplicity of singular values of matrix A.

Closed forms of the solutions including necessary and sufficient conditions (NSC) for the motion parameters, with different multiplicities of singular values of A are as follows :

- *Multiplicity* $= 2, e.g., \lambda_1 = \lambda_2 \neq \lambda_3$.

$$R = \lambda_1^{-1} A \left(\frac{\lambda_3}{\lambda_1} - s\right) U_3 V_3$$

$$T = w^{-1} \left(\frac{\lambda_3}{\lambda_1} - s\right) U_3 \, , \begin{pmatrix} a \\ b \\ c \end{pmatrix} = w V_3. \tag{3.32}$$

NSC : Rotation around an axis through the origin followed by a translation along the normal direction of the object surface (unique solution).

- *Multiplicity* $= 1, e.g., \lambda_1 > \lambda_2 > \lambda_3$.

$$R = U \begin{pmatrix} \alpha & 0 & \beta \\ 0 & 1 & 0 \\ -s\beta & 0 & s\alpha \end{pmatrix} V^T$$

$$T = w^{-1} \left[-\beta U_1 + \left(\frac{\lambda_3}{\lambda_2} - s\alpha\right) U_3\right]$$

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = w\left(\delta V_1 + V_3\right) \tag{3.33}$$

NSC : Rotation around an axis through the origin, followed by a translation along a direction different from the normal direction of the object surface (two solutions : in each of the two solutions, $sgn(\beta) = -sgn(\delta)$).

- *Multiplicity* $= 3, e.g., \lambda_1 = \lambda_2 = \lambda_3$.

$$R = \lambda_1^{-1} A$$

$$T = 0 \tag{3.34}$$

NSC : Rotation around an axis through the origin only (unique solution).

where :

$s = det(U)det(V)$

w =scale factor (constant)

a, b and c = planar object parameters

U_3, V_3 =third column of U and V, respectively

$\delta = \pm\sqrt{\frac{\lambda_1^2-\lambda_2^2}{\lambda_2^2-\lambda_3^2}}$

$\alpha = \frac{\lambda_1+s\lambda_3\delta^2}{\lambda_2(1+\delta^2)}$

$\beta = \pm\sqrt{1-\alpha^2}$

(for proof see [23]; Computer program for decomposing the singular values of matrix A is given in the Appendix).

Chapter 4

Computation Procedures And Experimental Results

4.1 Computation Procedures

More than ten different type of experiments consisting of both small and large motion estimations have been done. In all experiments, raw data, i.e., landmarks in the 3-D object space before motion, first view, are selected manually; whereas landmarks after motion, second view, are first computed by using a set of predefined motion parameters, and then varied to see the effects of missing and some extraneous landmarks in the second view.

By projecting those landmarks into the 2-D image space and processing the two resulting views of image sequence, we intend to estimate the motion and structure of the 3-D object under observation. The computation procedure is summarized as follows :

1. Select points on the observed planar object using Equation (3.12). These points are the feature points and referred to as the landmarks of the 3-D object before 3-D rigid motion (first view).

2. Determine the amount of rotation θ, direction cosine (n_1, n_2, n_3) and the amount of translation $(\Delta x, \Delta y, \Delta z)$. These data are predefined and needed when the motion parameters have been obtained. That is, to check how close the estimated motion parameters compared to the predefined motion parameters.

3. Compute points after motion (second view) by using Equation (2.20) and data from Step 2. Determine the effects of noise, missing and adding extraneous landmarks.

4. Project points obtained in Step 1 and 3 into the 2-D image space. Now we have four collection of points, i.e., two in the 3-D object space (Step 1 and 3), and the other two in the 2-D image space (Step 4).

5. Use probe and block algorithm to perform the correspondences task of points obtained from Step 4. Points in the first view are associated with the model landmarks, and points in the second view are associated with the scene landmarks.

6. Verify the results obtained from Step 5. Use Equation (3.18) to check the overall goodness of match, i.e., matching error, in least square

sense. The higher the matching error, the worse is the match.

7. From the matched points obtained from Step 5, compute the pure parameters using Equation (3.14).

8. Compute 3-D motion parameters using Equation (3.21) and (3.22) for small motion, and Equation (3.32),(3.33) and (3.34) for large motion. Note that calculating 3-D motion parameters for large motion is done after decomposing the singular values of matrix A given in Equation (3.25).

9. Verify the results obtained from Step 8 and those that are predefined in step 2.

The flowchart of the computation procedure is given in Figure (4.1).

4.2 Experimental Results

We shall present the results of four different experiments including both small (one) and large (three) rigid motions. In the experiments, the image coordinates in the second view are obtained from those in the first view after following a rigid motion using Equation (2.20) only. Further experimental results which consider the effects of missing and extraneous landmarks in the second view will be presented in Section (4.3).

As mentioned in Chapter 3, when the rotation is large there are three cases of rigid motions that can be recovered from at least four point cor-

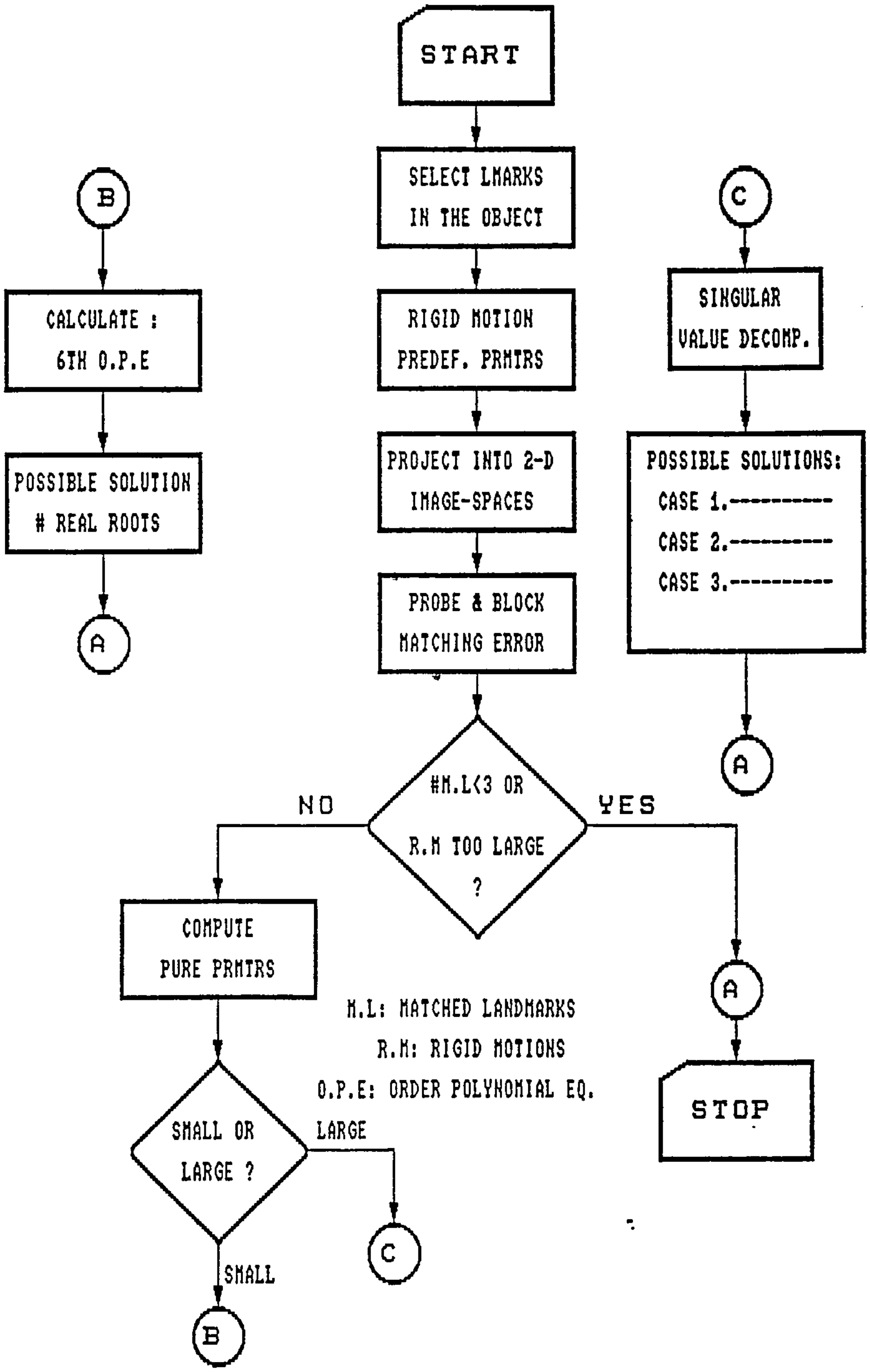

START
SELECT LMARKS IN THE OBJECT
RIGID MOTION PREDEF. PRMTRS
PROJECT INTO 2-D IMAGE-SPACES
PROBE & BLOCK MATCHING ERROR
#M.L<3 OR R.M TOO LARGE ?
NO
YES
COMPUTE PURE PRMTRS
SMALL OR LARGE ?
LARGE
SMALL
B
C
A
STOP
CALCULATE : 6TH O.P.E
POSSIBLE SOLUTION # REAL ROOTS
SINGULAR VALUE DECOMP.
POSSIBLE SOLUTIONS:
CASE 1.
CASE 2.
CASE 3.
M.L: MATCHED LANDMARKS
R.M: RIGID MOTIONS
O.P.E: ORDER POLYNOMIAL EQ.

Figure 4.1: Flowchart for Estimating Motion Parameters By Establishing Landmark Correspondences

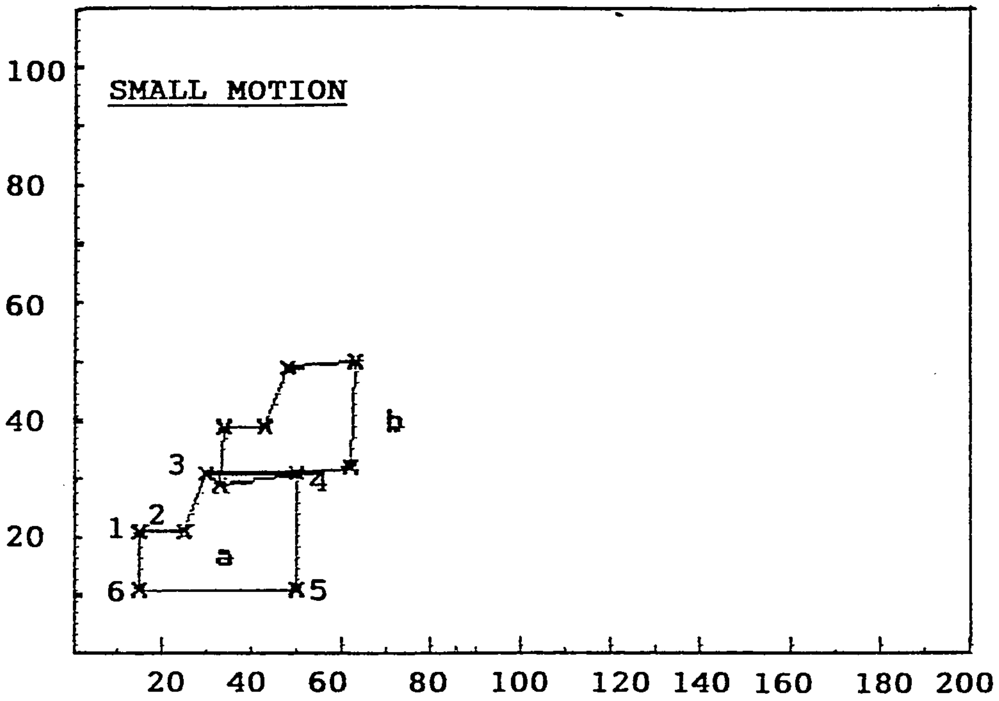


Figure 4.2: Landmarks of the Object (x): (a). Before and (b).After a Small Rigid Motion.

respondences. Hence, the distinction between the cases depend on the multiplicity of the singular values of matrix A as given in Equation (3.24), i.e., it depends on whether or not the translational vector coincides with the normal direction of the object surface. It is categorized Case 1, if the translational vector coincides with the normal direction of the object surface, otherwise it is Case 2. If there is no translation at all, it is categorized Case 3.

Table (4.1) shows the predefined motion and object parameters for both small and large motions, while Table (4.2) shows the estimated motion and structure of the 3-D object. Figure (4.2) shows the landmarks (x) of the object under observation before and after following a small rigid motion, whereas Figure (4.3) shows the landmarks before and after following a large rigid motion : (a) Case 1 ; (b) Case 2 ; (c) Case 3. From Table (4.1) we see that for both type of motions we use the same object parameters, the same axis of rotation which is an arbitrary axis through the origin and the same number of landmarks, but different rotation angle and translational vector.

From Table (4.2), we see that both small and large motion algorithm have their own advantages and disadvantages. In small motion algorithm, all landmarks in the first view match with those in the second view, and the matching error here is 0.36. This error is mainly because of the discrete representation of landmarks. It is smaller than those that we obtain in Case 1 and Case 2 for large motion, but greater than that obtained in Case 3. The reason for that is because when the observed object undergoing a large movement such that the normal of the object surface almost or exactly perpendicular to the view axis (optical axis), the least square error as given by Equation (3.17) is very large. Since the matching error given by equation (3.18) is derived from the least square error plus a penalty term for incomplete matching, this error is going to be large if the least square error is large, i.e., the matching error will fail to determine the correct match of landmarks. To evaluate that the above result is true, one such numerical experiment is also done for the same object and motion parameters, except for the amount of translation. In the experiment, instead of translating the

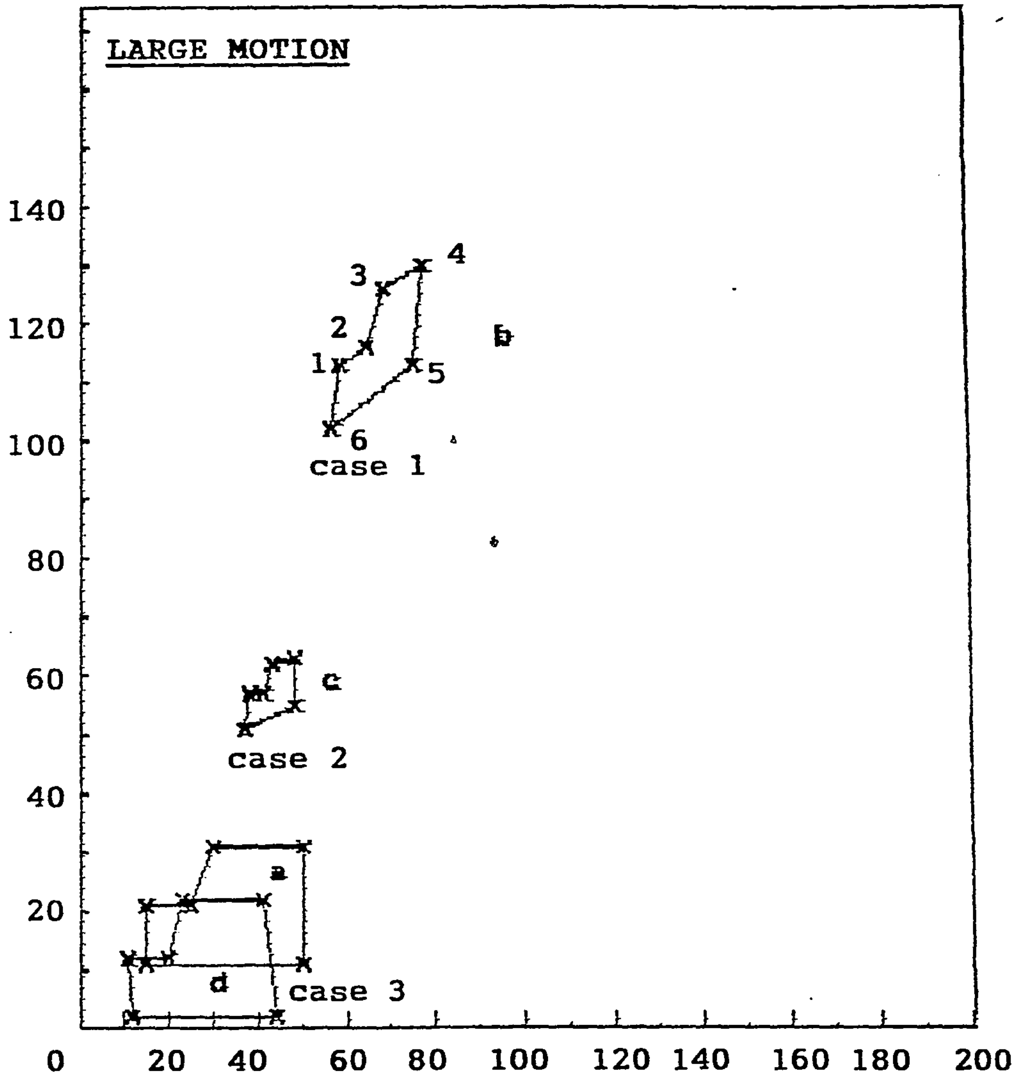


Figure 4.3: Landmarks of the Object: Before (a) and After a Large Rigid Motion : (b) Case 1 ; (c) Case 2 ; (d) Case 3.

Parameters	*Small Motion*	*Large Motion*
Rot. Angle	2^o	10^o (all cases)
Dir. Cosine	$(\cos 58^o, \cos 122^o, \cos 48^o)$	the same (all cases)

Trans. Vector	$(\frac{0.5}{N}, \frac{0.5}{N}, \frac{0.5}{N})$
	$N = \sqrt{a^2 + b^2 + c^2} = \sqrt{6}$
		case 1). $(\frac{2}{N}, \frac{4}{N}, \frac{2}{N})$
		case 2). $(\frac{2}{N}, \frac{3}{N}, \frac{4}{N})$
		case 3). (0,0,0)

Obj. Parameters	(1,2,1)	(1,2,1)
# of Lmarks @ t_1	6	6
# of Lmarks @ t_2	6	6

Table 4.1: Predefined Motion and Object Parameters

object as long as two unit away from the initial position along the normal direction as in Case 1 of Table (4.1), it is translated three unit away from the initial position with the same direction. Hence, the matching error is 4.01 and only four matched landmarks from total six we can get. Therefore, a penalty is added here to the least square error for incomplete matching. In contrast for Case 3, the matching error is 0.06, the reason for this is because the object follows a rotation around an arbitrary axis through the origin only (no translation).

Although the probe and block algorithm can perform the matching task well if the object follows only a small rigid motion, it does not mean that the estimated motion parameters will be entirely dependent on how well the landmarks in the first view can match to those in the second view. That is, it still depends on the algorithm that we use to estimate these unknown motion parameters. For example, in small motion algorithm, Δz in Equation (3.19) is a scale factor and can not be determined, therefore

Parameters	*Small Motion*	*Large Motion*

Rot. Angle	Sol.1 :
	$\phi_1 = 0.0147$
	$\phi_2 = -0.0147$
	$\phi_3 = 0.0185$
		case 1). 9.38°
		case 2). 9.10°
		case 3). 9.57°

Dir. Cosine	Sol.2 :
	$\phi_1 = 0.0087$
	$\phi_2 = -0.0087$
	$\phi_3 = 0.0123$
		case 1). (0.52,-0.52,0.66)
		case 2). (0.504,-0.504,0.655)
		case 3). (0.53,-0.53,0.67)

Trans. Vector	Sol.1 :
	$\Delta x^* = 1.01$
	$\Delta y^* = 1.01$
	Sol.2 :
	$\Delta x^* = 0.073$
	$\Delta y^* = 0.074$
		case 1). $\omega^{-1}(-2.1, -4.49, -2.32)$
		case 2). $\omega^{-1}(0.18, 0.26, 0.38)$
		case 3). $\omega^{-1}(0, 0, 0)$
		ω = arbitrary constant

Obj. Parameters	Sol.1 :
	(0.23,0.45,0.22)
	Sol.2 :
	(0.34,0.50,0.12)
		case 1). $\omega(-0.42, -0.81, -0.41)$
		case 2). $\omega(4.54, 9.30, 4.29)$
		case 3). arbitrary

# of Matched Landmarks	6	6

Matching Error	0.36	case 1). 0.91
		case 2). 2.02
		case 3). 0.06

Table 4.2: Estimated Motion and Object Parameters

Parameters	*Small Motion*
Rot. Angle	Sol.1 : 1.55° Sol.2 : 1°
Dir. Cosine	Sol.1 : (0.545,-0.545,0.684) Sol.2 : (0.5,-0.5,0.707)
Trans. vector	Sol.1 : (0.202,0.202,0.20) Sol.2 : (0.015,0.015,0.20)
Obj. Parameters	Sol.1 : (1.15,2.25,1.10) Sol.2 : (1.7,2.5,0.6)

Table 4.3: Solution to the Small Rigid Motion.

the solution to the motion is a function of Δz which can be computed only if it is assumed as a relative depth of the object. In addition, as a result of solving a sixth order polynomial equation, we may have at most six real roots which yields six solutions to the motion. In the experiment, we have found two real and four complex roots. From these two solutions, we have no way to know the actual motion parameters since we have eight equations with nine unknowns as given in Equation (3.22). Thus, all that we can hope is a multiplicity of the possible solutions which may give an infinite number of solutions since Δz can take any value. In our experiment, however, Δz is given and is equal to 0.2. Substituting this value into the estimated motion and object parameters in Table (4.2), we have two solutions to the motion as shown in Table (4.3). Note that Δz can basically act as a scale factor for the translational vector which will give us an infinite number of solutions to the motion.

In large rotation algorithm, on the other hand, once we have determined the pure parameters a_i's, the unknown motion parameters can be computed by decomposing the singular values of a 3×3 matrix that consists of those parameters. Since these parameters are obtained from a unique mapping given four or more point correspondences, decomposing the singular values of this matrix means that we are looking for the physical characterization of the motion in the object space as given in Equation (3.31). As a result of this equation, and by using the rigidity constraint of the object under observation, the number of possible solutions are never more than two because a plane in 3-D object space can be oriented in at most two possible directions in order to cut a circle in an ellipsoid as described by Equation (3.31). The fact above is only true if and only if the singular values λ_i (i=1,2,3) are all distinct (Case 2). If two of the three singular values are equal, then there is only one possible orientation for the object surface before motion (Case 1). Finally, if the singular values are all identical, the motion consists of rotation around an axis through the origin only, i.e., $\Delta x = \Delta y = \Delta z = 0$ (Case 3). Hence the object surface can be anywhere, but the solution to the motion is only one. Note that, the results shown in Table (4.2) deviate up to 10% from the predefined motion and object parameters. This is because of the nonlinearity of the computation procedure and roundoff error generated by the computer.

4.3 Further Experimental Results

The effects of missing and extraneous landmarks in the second view are evaluated. In all experiments, we use the same motion and object parameters and the observed object follows only a large rigid motion (Case 2) as shown in Table (4.4). Figure (4.4) shows the landmarks of the object before and after following a rigid motion : (a) missing one point; (b) adding two extraneous points. Hence we consider only one missing and two extraneous from given six landmarks. The reason is because the matching error is undefined when the matched landmarks are less than three. Furthermore, in order to be able to use the motion estimation algorithm [21][23], we need at least four corresponding landmarks. We have also done, however, such experiments when two missing and three extraneous landmarks taken into account. The matching error here is undefined, i.e., the matching algorithm fails to detect the correct matches, and the motion and object parameters can not be estimated.

From Table (4.5) we see that the estimated and object parameters are not shaken by missing and extraneous points to the image coordinates in the second view. As described in Chapter 3, the pure parameters as shown in Equation (3.23) are a function of motion and object parameters only. These parameters are resulted from a unique mapping, Equation (3.14), of points $P_i = (X_i, Y_i)$ to $P_i' = (X_i', Y_i')$ where $i = 1...n$ is a landmark location of the object under observation. Therefore, regardless of which

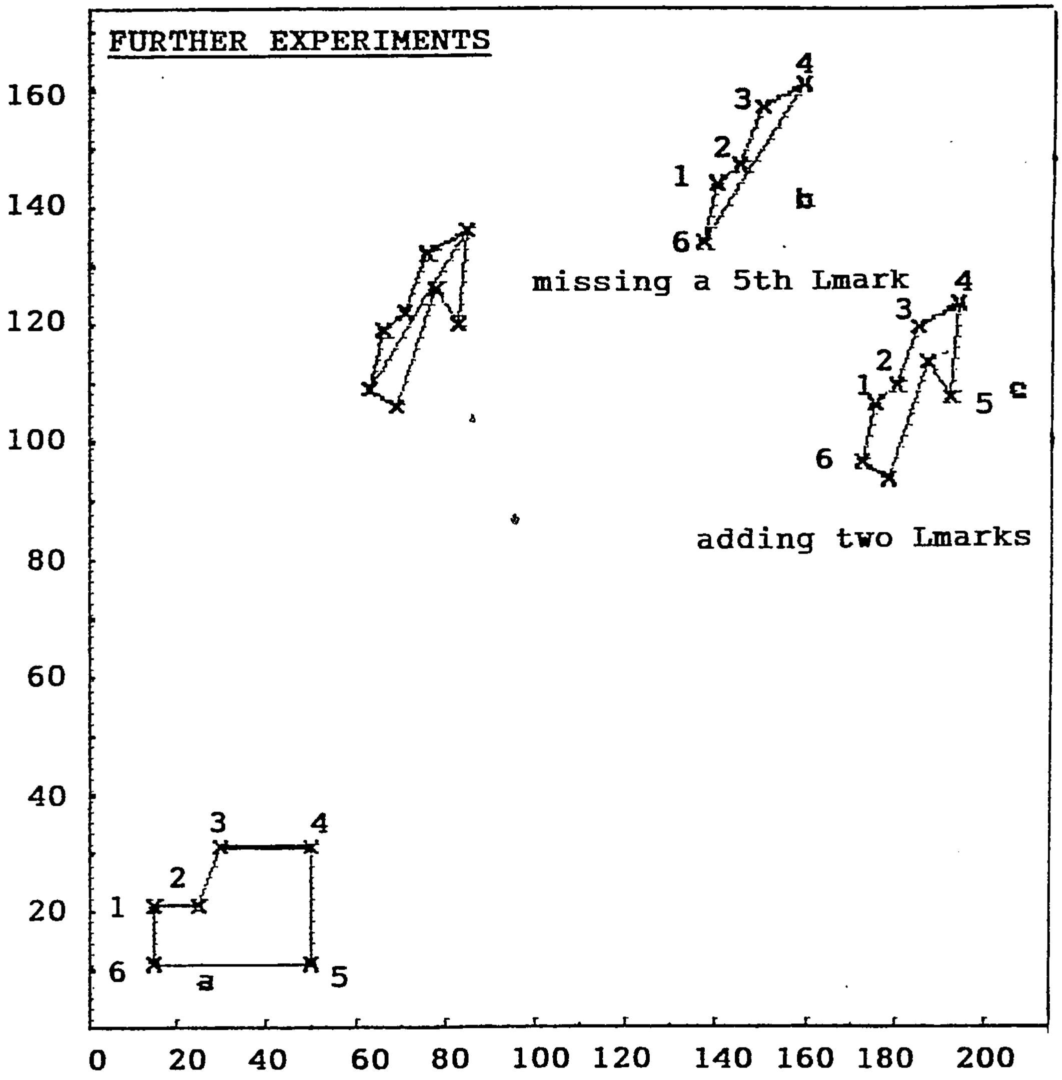


Figure 4.4: Landmarks of the Object: Before (a) and After a Large Rigid Motion - Case 2: (b) Missing One Point ; (c) Adding Two Extraneous Points.

Parameters	*Missing*	*Adding*
Rot.Angle	12^o	dito
Dir. Cosine	$(\cos 45^o, \cos 60^o, \cos 120^o)$	dito
Obj. Parameters	(1,2,1)	dito
Trans. Vector	$(\frac{2}{\sqrt{6}}, \frac{4}{\sqrt{6}}, \frac{2}{\sqrt{6}})$	dito
# of Lmarks @ t_1	6	dito
# of Lmarks @ t_2	5	8

Table 4.4: Predefined Motion and Object Parameters (Further Experiments)

Parameters	*Missing*	*Adding*
Rot.Angle	11.12^o	11.06^o
Dir. Cosine	(0.71,0.52,-0.52)	(0.69,0.48,-0.48)
Obj. Parameters	$\omega(3.84,9.25,3.75)$	$\omega(3.65,8.76,3.58)$
Trans. Vector	$\omega^{-1}(0.21,0.35,0.23)$	$\omega^{-1}(0.18,0.31,0.20)$
# of Matched Lmarks	5	4
Matching Error	2.67	3.49

Table 4.5: Estimated Motion and Object Parameters (Further Experiments)

corresponding points on the observed object being used, as long as there are more than three landmark corresponding pairs, the pure parameters and in fact the estimated and object parameters will remain relatively the same. Note that, because of a penalty for incomplete matching is added in the experiment with two extraneous points, the matching error could be different for both experiments, but the motion and object parameters are still relatively the same.

Chapter 5

Conclusion And Recommendation

The application of a landmark based approach in recognizing 3-D rigid planar objects[3] along with two different motion and structure estimation algorithm [21][23] have been studied to perform 3-D motion and structure estimation tasks of moving objects. The recognition algorithm, known as probe and block matching algorithm, is based on sphericity value, a shape measure, derived from affine transformation that maps feature points of the observed object from first to second view in the 2-D image space.

It is desired to have the correspondences between feature points in the first view, at time t_1, and those that in the second view, at time t_2. Once we have determined these correspondences, the reconstruction of the 3-D object, i.e., object parameters, along with the motion parameters, i.e., rotation matrix and translational vector, are then estimated using the two motion and structure estimation algorithm.

The two estimation algorithm above are based on a unique mapping given four or more feature point correspondences. By working with a set of so called pure parameters resulted from the above unique mapping, the structure and motion parameters of the moving object can be obtained by solving a sixth order polynomial equation if the motion is small, and decomposing the singular values of a 3×3 matrix consisting of those parameters if the motion is large. From all numerical experiments that have been tried, the following are the conclusion and recommendation.

1. The probe and block algorithm can perform the landmark matching tasks well if the object under observation follows a rigid motion such that the normal of the object surface is not almost perpendicular to the view axis (optical axis). The algorithm is also capable of detecting the correct matches of landmarks when missing and some extraneous landmarks in the second view are taken into account. If there are too many landmarks missing and mixing with some other extraneous landmarks such that the sequential order of the original landmarks is lost, the algorithm fails to determine the correct matches.

2. When the rigid motion is small, only a relative depth of the object we can get. The number of solutions, aside from the scale factor for the translational vector, depend on the number of real roots of the sixth order polynomial equation. If the motion is large, the number

of solutions is either one or two depending on the multiplicity of the singular values of the 3×3 real matrix of the pure parameters.

3. The motion estimation algorithm is convenient, inexpensive and not shaken by missing or extraneous landmarks to the object under observation as long as there are more than three landmark corresponding pairs. That is, regardless of which landmarks in the object being used, the estimated motion and object parameters will remain relatively the same.

4. Due to the nonlinearity of the computation procedure that has been used and roundoff error generated by the computer, the results presented in Chapter 4 deviate up to 10% from the predefined motion and object parameters. To increase the accuracy of the results is a matter of selecting programming package.

5. It is difficult to evaluate the performance of the whole procedure of the computation, since it involves two complex algorithm which is, each of them, not propotionally dependent. Although the large motion algorithm is relatively better than the small algorithm, the probe and block matching algorithm will fail to determine the correct matches of landmarks if the object under observation moves such that the normal of the object surface is almost perpendicular to the optical axis.

6. A challenging further study is to consider the effect of noise to the landmarks when the object undergoing *medium* movement. It is also necessary to determine the physical quantity for the term *medium* above.

Appendix A

Solution to Linear Equation $AX = B$

(Programming Code in C)

```
 /* This program computes linear equation ax=b
 used along with LUDCMP and LUBKSB both taken
 from Numerical Recipes in C good luck...*/


 #include <malloc.h>
 #include <stdio.h>
 #include <math.h>
 main(ac, av)
 int ac;
 char **av;
 {
 float **a, *b, d, jj;
 int n, *indx, i, j;
 n=atoi(av[1]);
 a=matrix(1,10,1,10);
 b=vector(1,10);
 indx=ivector(1,n);

for (i=1; i<=n; i++)
for (j=1; j<=n; j++)
        a[i][j]=atof(av[(i-1)*n+j+1]);
for (i=1; i<=n; i++)
        b[i]=atof(av[1+n*n+i]);

for (i=1; i<=n; i++) {
    for (j=1; j<=n; j++) {
        printf("  %5.2f\t",a[i][j]);
        }
        printf("  =  %5.2f\n", b[i]);
}
ludcmp(a,n,indx,&d);
lubksb(a,n,indx,b);
printf("  solution to Ax=b :\n");
  for (i=1; i<=n; i++)
      printf("%5.3f\n", b[i]);

}

/********************************************/

/* To perform LU decomposition
copied from Numerical Recipes in C */

#include <math.h>
#define TINY 1.0e-20
void ludcmp(a,n,indx,d)
int n, *indx;
float **a, *d;
{
int i, imax, j, k;
float big, dum, sum, temp;
float *vv, *vector();
void nrerror(), free_vector();
vv=vector(1,n);
*d=1.0;
for (i=1; i<=n;i++) {
     big=0.0;
     for (j=1;j<=n;j++)
         if ((temp=fabs(a[i][j]))>big) big=temp;
         if (big==0.0)
            nrerror("Singular matrix in routine LUDCMP");
     vv[i]=1.0/big;
}
for (j=1;j<=n;j++) {
    for (i=1;i<j;i++) {
     sum=a[i][j];
        for(k=1;k<i;k++) sum -= a[i][k]*a[k][j];
             a[i][j]=sum;
     }
big=0.0;
for(i=j;i<=n;i++) {
        sum=a[i][j];
        for (k=1;k<j;k++)
                sum -= a[i][k]*a[k][j];
        a[i][j]=sum;
        if((dum=vv[i]*fabs(sum))>=big) {
                big=dum;
                imax=i;
        }
}
if (j!=imax) {
        for (k=1; k<=n;k++) {
                dum=a[imax][k];
                a[imax][k]=a[j][k];
                a[j][k]=dum;
        }
        *d= -(*d);
        vv[imax]=vv[j];
}
indx[j]=imax;
if(a[j][j]==0.0) a[j][j]=TINY;
if (j!=n) {
  dum=1.0/(a[j][j]);
     for (i=j+1;i<=n;i++)
        a[i][j] *= dum;
     }
}
free_vector(vv,1,n);
}

/**************************************************/

/* To perform LU backsubstitution
Copied from Numerical Recipes in C */

void lubksb(a, n,indx, b)
float **a, b[];
int n, *indx;
{
int i, ii=0, ip, j;
float sum;
for (i=1; i<=n; i++)     {
        ip=indx[i];
        sum=b[ip];
        b[ip]=b[i];
if (ii)
   for (j=ii; j<=i-1; j++)
   sum -= a[i][j]*b[j];
   else if (sum) ii=i;
      b[i]=sum;
}
for (i=n; i>=1; i--)     {
  sum=b[i];
   for (j=i+1; j<=n; j++)
     sum -= a[i][j]*b[j];
        b[i]=sum/a[i][i];
    }
}
```

Appendix B

SOLUTION TO THE SIXTH ORDER POLYNOMIAL EQUATION

(Programming Code in FORTRAN)

```
C THIS PROGRAM CALCULATES THE ROOTS OF A SIXTH ORDER
C POLYNOMIAL EQUATION USING BAIRSTOWS METHOD TAKEN FROM
C APPLIED NUMERICAL METHODS FOR DIGITAL COMPUTATIONS BY
C M.L. JAMES, AND J.M. SMITH

      DIMENSION A(30),B(30),C(30)
      WRITE(6,2)
    2 FORMAT('1',3X,'REAL PART',3X,'IMAGINARY PART',
     *3X,'ITERATIONS'//)
      READ(5,3) UI,VI,EPSI,N
    3 FORMAT(3F3.1,I2)

C ENTER POLYNOMIAL COEFFICIENTS:

      READ(5,4)  (A(I),I=1,N)
    4 FORMAT(5I3)

C SEE IF N=0,N=1,OR N IS GREATER THAN 1

   40 IF(N-1)100,5,7
    5 P=-A(1)
      Q=0.
      IT=1
      WRITE(6,6)N,P,Q,IT
    6 FORMAT(' ','X(',I2,')=',2X,F8.4,6X,F8.4,10X,I3)
      GO TO 100

C SEE IF N=2 OR IF N IS GREATER THAN 2

    7 IF(N.EQ.2) GO TO 8
      GO TO 13
    8 U=A(I)
      V=A(2)
      IT=1
    9 P=-U/2.
      RAD=U**2-4.*V

C CHECK THE SIGN OF U**2-4.*V

      IF(RAD.GT.0.) GO TO 12
      RAD=-RAD
      Q=SQRT(RAD)/2.
      WRITE(6,6)N,P,Q,IT
      N=N-1
      Q=-Q
   90 WRITE(6,6)N,P,Q,IT
   10 N=N-1

C CHECK TO SEE IF N IS GREATER THAN ZERO

      IF(N.LE.0) GO TO 100
      DO 11 I=1,N
   11 A(I)=B(I)
      GO TO 40
      12  Q=SQRT(RAD)/2.
      W=P
      Z=Q
      P=P+Q
      Q=0.
      WRITE(6,6)N,P,Q,IT
      N=N-1
      P=W-Z
      GO TO 90
   13 U=UI
      V=VI
      IT=1
```

```
C CALCULATE THE B VALUES
   50 B(1)=A(1)-U
      B(2)=A(2)-B(1)*U-V
      DO 14 K=3,N
   14 B(K)=A(K)-B(K-1)*U-B(K-2)*V

C CALCULATE THE C VALUES
      C(1)=B(1)-U
      C(2)=B(2)-C(1)*U-V
      M=N-1
      DO 15 K =3,M
   15 C(K)=B(K)-C(K-1)*U-C(K-2)*V

C CALCULATE DELU AND DELV
      IF(N.GT.3) GO TO 17
      DENOM=C(N-1)-C(N-2)**2
      IF(DENOM.EQ.0.)GO TO 30
      DELU=(B(N)-B(N-1)*C(N-2))/DENOM
   16 DELV=(C(N-1)*B(N-1)-C(N-2)*B(N))/DENOM
      GO TO 18
   17 DENOM=C(N-1)*C(N-3)-C(N-2)**2
      IF(DENOM.EQ.0.) GO TO 30
      DELU=(B(N)*C(N-3)-B(N-1)*C(N-2))/DENOM
       GO TO 16

C CALCULATE NEW U AND V VALUES

   18 U=U+DELU
      V=V+DELV
      SUM=ABS(DELU)+ABS(DELV)

C STORE THE FIRST SUM CALCULATED

      IF(IT.EQ.1.) GO TO 19
      GO TO 20
   19 STORE=SUM
      GO TO 21
   20 IF(IT.EQ.50)GO TO 28
      IF(IT.GE.200)GO TO 26
   21 IF(SUM.LE.EPSI)GO TO 9
      IF(IT.EQ.100)GO TO 23
   22 IT=IT+1
      GO TO 50
   23 WRITE(6,24)
   24 FORMAT(' ',10X,'CONVERGENCE IS SLOW')
      WRITE(6,25)U,V,DELU,DELV
   25 FORMAT(' ','U=',E14.7,3X,'V=',E14.7,3X,'DELU='
     *,E14.7,3X,'DELV=',E14.7)
      GO TO 22
   26 WRITE(6,27)
   27 FORMAT(' ',10X,'ITERATING STOPED AFTER 200
     *ITERARTIONS')
      WRITE(6,25)U,V,DELU,DELV
      GO TO 100

C SEE IF SUM AFTER 50 ITERATIONS
C EXCEEDS FIRST SUM STORED

   28 IF(SUM.LT.STORE)GO TO 21
      WRITE(6,29)
   29 FORMAT(' ',10X,'DIVERGENCE OCCURING')
      WRITE(6,25)U,V,DELU,DELV
      GO TO 100
   30 WRITE(6,31)
   31 FORMAT(' ',10X,'DENOM IS ZERO')
```

```
      GO TO 100
  100 STOP
      END
```

Appendix C

SINGULAR VALUE DECOMPOSITION

(Programming Code in C)

```
/* this program computes singular values of
a 3x3 real matrix used along with svdcmp
and svbksb both taken from Numerical Recipes in C */


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

main (ac,av)
int ac;
char **av;
{
double **a,*b,d,jj,wmax,wmin,**u,*w,**v,*x;
int i,j,n,*indx,k;
n=atoi(av[1]);
a=matrix(1,n,1,n);
b=vector(1,n);
u=matrix(1,n,1,n);
w=vector(1,n);
v=matrix(1,n,1,n);
x=vector(1,n);
indx=ivector(i,n);

for(i=1;i<=n;i++)
   for(j=1;j<=n;j++)
        a[i][j]=atof(av[(i-1)*n+j+1]);

/*calculate A = U W V*/

printf("singular decomposition of matrix A :\n\n");
for(i=1;i<=n;i++)  /*copy a[i][j] into u[i][j]*/
    for(j=1;j<=n;j++)
         u[i][j] = a[i][j];

     svdcmp(u,n,n,w,v);

     for(i=1;i<=n;i++){
        for(j=1;j<=n;j++){
            printf("%2.2f\t",u[i][j]);
        }
     printf("%2.3f\n",w[i]);
     }
     printf("\n");

     for(i=1;i<=n;i++){
          for(j=1;j<=n;j++){
              printf("%2.2f\t",v[i][j]);
          }
            printf("\n");
        }
      }



    /* This is a routine for SV decomposition
    used along with svbksb both copied from
    Numerical Recipes in C */

    #include <math.h>
    static double at,bt,ct;
    #define PYTHAG(a,b) ((at=fabs(a)) > (bt=fabs(b)) ?\
    (ct=bt/at,at*sqrt(1.0+ct*ct)) :\
    (bt ? (ct=at/bt,bt*sqrt(1.0+ct*ct)): 0.0))

    static double maxarg1,maxarg2;
```

```
#define MAX(a,b) (maxarg1=(a),maxarg2=(b),(maxarg1) > (maxarg2) ?\
        (maxarg1) : (maxarg2))
#define SIGN(a,b) ((b) >= 0.0 ? fabs(a) : -fabs(a))

void svdcmp(a,m,n,w,v)
double **a,*w,**v;
int m,n;

{
 int flag,i,its,j,jj,k,l,nm;
 double c,f,h,s,x,y,z;
 double anorm=0.0,g=0.0,scale=0.0;
 double *rv1,*vector();
 void nrerror(),free_vector();

 if (m<n)
 nrerror("SVDCMP: You must augment A with extra zero rows");
 rv1=vector(1,n);

 /*Housholder reduction to bidiagonal form*/
 for (i=1;i<=n;i++) {
   l=i+1;
        rv1[i]=scale*g;
        g = s = scale = 0.0;
 if(i<=m) {
   for (k=i;k<=m;k++) scale += fabs(a[k][i]);
   if (scale) {
     for (k=i;k<=m;k++) {
        a[k][i] /= scale;
          s += a[k][i]*a[k][i];
    }
 f=a[i][i];
 g= (-SIGN(sqrt(s),f));
 h = f*g-s;
 a[i][i] = f-g;
 if(i != n) {
      for (j=l;j<=n;j++) {
          for(s=0.0,k=i;k<=m;k++) s += a[k][i]*a[k][j];
          f = s/h;
          for(k=i;k<=m;k++) a[k][j] += f*a[k][i];
      }
 }
 for (k=i;k<=m;k++) a[k][i] *= scale;
 }
 }
 w[i] = scale*g;
   g = s = scale = 0.0;
 if(i <= m && i != n) {
 for (k=l;k<=n;k++) scale += fabs(a[i][k]);
    if (scale) {
      for (k=l;k<=n;k++) {
 a[i][k] /= scale;
 s  += a[i][k]*a[i][k];
 }
 f = a[i][l];
 g = (-SIGN(sqrt(s),f));
 h = f*g-s;
 a [i][l] = f-g;
 for (k=l;k<=n;k++) rv1[k] = a[i][k]/h;
 if (i != m) {
 for (j=l;j<=m;j++) {
   for(s=0.0,k=l;k<=n;k++) s += a[j][k]*a[i][k];
   for(k=l;k<=n;k++) a[j][k] += s*rv1[k];
 }
 }
 for (k=l;k<=n;k++) a[i][k] *= scale;
```

```
  }
  }
  anorm=MAX(anorm,(fabs(w[i])+fabs(rv1[i])));
  }

 /*accumulation of right-hand transformations*/

  for (i=n;i>=1;i--) {
  if (i <n) {
     if (g) {
  for (j=1;j<=n;j++)
      v[j][i] = (a[i][j]/a[i][l])/g;
  for (j=1;j<=n;j++) {
      for (s=0.0,k=l;k<=n;k++) s += a[i][k]*v[k][j];
      for (k=l;k<=n;k++) v[k][j] += s*v[k][i];
   }
  }
  for (j=l; j<=n;j++) v[i][j] = v[j][i] = 0.0;
  }
  v[i][i] = 1.0;
  g = rv1[i];
  l = i;
 }
/*Accumulation of left-hand transformation*/
  for (i=n;i>=1;i--) {
  l = i+1;
  g = w[i];
  if(i < n)
  for (j=l;j<=n;j++) a[i][j] = 0.0;
  if(g) {
  g = 1.0/g;
  if(i != n){
  for (j=l;j<=n;j++) {
  for (s=0.0,k=l;k<=m;k++) s += a[k][i]*a[k][j];
  f=(s/a[i][i])*g;
  for (k=i;k<=m;k++) a[k][j] += f*a[k][i];
 }
}
  for (j=i;j<=m;j++) a[j][i] *= g;
 } else {
  for (j=i;j<=m;j++) a[j][i]=0.0;
 }
 ++a[i][i];
}

/*diagonalization of the bidiagonal form*/
 for (k=n;k>=1;k--) {
 for (its=1;its<=30;its++) {
 flag = 1;
 for(l=k;l>=1;l--){
 nm = l-1;
   if(fabs(rv1[l])+anorm==anorm) {
      flag = 0;
         break;
    }
   if(fabs(w[nm])+anorm==anorm) break;
 }
if(flag) {
c = 0.0;
s = 1.0;
for(i=l;i<=k;i++){
 f = s*rv1[i];
 if(fabs(f)+anorm != anorm) {
      g = w[i];
      h = PYTHAG(f,g);
      w[i] = h;
```

```
       h = 1.0/h;
       c = g*h;
       s = (-f*h);
       for(j=1;j<=m;j++){
          y = a[j][nm];
          z = a[j][i];
          a[j][nm] = y*c+z*s;
          a[j][i] = z*c-y*s;
        }
      }
 }
 }
    z = w[k];
   if(l==k) {
       if (z <0.0 ) {
  w[k] = (-z);
  for (j=1;j<=n;j++) v[j][k]=(-v[j][k]);
 }
 break;
 }
if (its==30) nrerror("No converegence in 30 SVDCMP iteration");
x = w[l];
nm = k-1;
y = w[nm];
g = rv1[nm];
h = rv1[k];
f = ((y-z)*(y+z)+(g-h)*(g+h))/(2.0*h*y);
g = PYTHAG(f,1.0);
f = ((x-z)*(x+z)+h*((y/(f+SIGN(g,f)))-h))/x;
c = s = 1.0;
for(j=l;j<=nm;j++) {
i = j+1;
g = rv1[i];
y = w[i];
h = s*g;
g = c*g;
z = PYTHAG(f,h);
rv1[j] = z;
c = f/z;
s = h/z;
f = x*c+g*s;
g = g*c-x*s;
h = y*s;
y = y*c;
for(jj=1;jj<=n;jj++) {
 x = v[jj][j];
 z = v[jj][i];
 v[jj][j] = x*c+z*s;
 v[jj][i] = z*c-x*s;
}
z = PYTHAG(f,h);
w[j] = z;
if(z){
 z = 1.0/z;
 c = f*z;
 s = h*z;
}
f = (c*g)+(s*y);
x = (c*y)-(s*g);
for(jj=1;jj<=m;jj++) {
 y = a[jj][j];
 z = a[jj][i];
 a[jj][j] = y*c+z*s;
 a[jj][i] = z*c-y*s;
}
}
```

```
        rvl[l] = 0.0;
        rvl[k] = f;
          w[k] = x;
        }
        }
        free_vector(rvl,1,n);
        }

    void svbksb(u,w,v,m,n,b,x)
    float **u,w[],**v,b[],x[];
    int m,n;

    /* Copied from Numerical Recipes in C */

    {
     int jj,j,i;
     float s,*tmp,*vector();
     void free_vector();
     tmp=vector(1,n);
     for(j=1;j<=n;j++) {     /*calculate U transpose B*/
          s=0.0;
           if (w[j]) {          /*Nonzero result only if wj is nonzero*/
     for (i=1;i<=m;i++) s += u[i][j]*b[i];
     s /= w[j];       /*This is the divide by wj*/
     }
     tmp[j]=s;
     }
     for (j=1; j<=n;j++) {    /*Matrix multiply by V to get answer*/
     s=0.0;
     for (jj=1;jj<=n;jj++) s +=v[j][jj]*tmp[jj];
     x[j]=s;
      }
     free_vector(tmp,1,n);
  }
```

Bibliography

[1] J.K. Aggarwal, N. Nandhakumar, "On the Computation of Motion from Sequence of Images - A Review," *Proc. IEEE,* Vol. 76, No. 8, pp. 917-935, August 1988.

[2] J. Aloimonos and C.M. Brown, "Perception of Structure from Motion," *Proc. IEEE Conf. Comput. Vision and Pattern Recognition,* Miami Beach, FL, June 1986, pp. 510-517.

[3] N. Ansari, E.J. Delp, "Recognizing Planar Objects in 3-D Space," *Proc. SPIE Visual Communications and Image Processing, Nov. 8-10, 1989, Philadelphia, PA*, Vol. 1197, pp. 127-138.

[4] N. Ansari, "Shape Recognition : A Landmark-Based Approach," *Ph.D. Thesis, Purdue University, School of Engineering,* West Lafayette, IN, August 1988.

[5] F. Attneave, "Some Informational Aspects of Visual Perception," *Psychological Review,* Vol. 61, No. 3, pp. 183-1954, 1954.

[6] F.L. Bookstein, *The Measurement of Biological Shape and Shape Change,* New York : Springer-Verlag, 1978.

[7] C. Cafforio and F. Rocca, "Methods for Measuring Small Displacement of Television Images," *IEEE Trans. Inform. Theory,* Vol. IT-22, No. 5, pp. 573-579, 1976.

[8] D. Gans, *Transformation And Geometrics,* Newyork : Appleton-Century- Crofts, 1969.

[9] J.E. Hall, *Analytic Geometry*, Wadsworth Publishing Company, inc.

[10] B.K.P. Horn and E.J. Weldon, Jr., "Computationally-Efficient Methods for Recovering Translational Motion," in *Proc. IEEE First Int. Conf. Computer Vision,* 1987, pp. 2-11.

[11] T.S. Huang and R.Y. Tsai, "Image Sequence Analysis : Motion Estimation," in *Image Sequence Analysis*, T.S. Huang, Ed. New York : Springer-Verlag, 1981.

[12] T.S. Huang, C.H. Lee, "Motion And Structure from Orthographic Projections," *IEEE Trans. Pattern Analysis and Machine Intelligence,* Vol. II, No. 5, pp. 536-541, May1989.

[13] M.L. James, J.M. Smith, J.C. Wolgord, *Applied Numerical Methods for Digital Computations*, 3rd. Edt. Harper and Row, Publishers, Newtork.

[14] Johansson, G., "Visual Perception of Biological Motion and A Model for Its Analysis," *Perception and Psychophysics* Vol. 14, No. 2, pp. 201-211, 1973.

[15] J.O. Limb and J.A. Murphy, "Estimation of Velocity of Moving Images in Television," *Computer Graphics Image Processing,* Vol. 4, pp. 311-327, 1975.

[16] —, "Measuring the Speed of Moving Objects from Television," *IEEE Trans. Commun.,* Vol. COM-23, pp. 474-478, 1975.

[17] H.C. Longuet-Higgins, "A Computer Algorithm for Reconstructing A Scene from Two Projections," *Nature,* Vol. 293, pp. 133-135, 1981.

[18] D.F. Rogers, J.A. Adams, *Mathematical Elements for Computer Graphics,* New York : Mc Graw-Hill, 1976.

[19] R.J. Scalkoff and E.S. Mc Vey, "A Model and Tracking Algorithm for A Class of Video Targets," *IEEE Trans. Pattern Analysis and Machine Intelligeence,* Vol. PAMI-4 No. 1, pp. 2-10, 1982.

[20] M. Subbarao and A.M. Waxman, "Closed Form Solutions to Image Flow Equations for Planar Surfaces in Motion," *Computer Vision Graphics Image Processing,* Vol. 36, pp. 208-228, 1986.

[21] R.Y. Tsai, T.S. Huang, "Estimating 3-D Motion Parameters of A Rigid Planar Patch," *IEEE Trans. Acoustics, Speech and Signal Processing,* Vol. ASSP-29, No. 6, pp. 1147-1152, 1981.

[22] ——, "Uniqueness and Estimation of 3-D Motion Parameters of Rigid Objects with Curved Surfaces," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 6, No. 1, pp. 13-26, 1984.

[23] ——, W. Zhu, "Estimating 3-D Motion Parameters of A Rigid Planar Patch II : Singular Value Decomposition," *IEEE Trans. Acoustics, Speech and Signal Processing,* Vol. ASSP-30 No. 4, pp. 525-534, 1982.

[24] ——, "Estimating 3-D Motion Parameters of A Rigid Planar Patch III : Finite Point Correspondences and 3-View Problem," *IEEE Trans.Acoustics, Speech, Signal Processing,* Vol. ASSP-32, No. 2, pp. 213-219 April 1984.

[25] S. Ullman, *The Interpretation of Visual Motion,* Cambridge , MA : M.I.T. Press. 1979.

[26] Wallach, H. And D.N. O'Connell, "Kinetic Depth Effect," *J. Exp. Psychol.* Vol. 45, No. 4, pp. 205-17, 1953.

[27] M. Yamamoto, "A General Aperture Problem for Direct Estimation of 3-D Motion Parameters," *IEEE Trans. Pattern Analysis and Machine Intell.,* Vol. II, No. 5,

pp. 528-536, May 1989.