MODEL-BASED COMPUTER VISION:

Motion Analysis, Motion-based Segmentation, 3D Object Recognition.

By

Man-lee LIU

An Abstract

Submitted to the Division of Computer Science and Engineering
in partial fulfillment of the requirements for the
degree of Master of Philosophy
at the
Chinese University of Hong Kong

May 1998

Thesis supervisor: Prof. Kin-hong WONG

ABSTRACT

Model-based computer vision is an important problem and research topic, which can be divided into several areas such as motion analysis, motion-based segmentation and three-dimensional object recognition. In model-based computer vision, the model containing the 3D information of the object being investigated is assumed to be known in advance. In the past, many algorithms were proposed. Apart from the model of the object, other information such as image intensity values and image features were used in some of the other algorithms.

This thesis aims at providing analysis for the following three areas.

Firstly, model-based motion analysis is discussed. The objective of this area is to investigate the motion of a target object between two time instants. Two model-based motion analysis algorithms using the model and the intensity image of the object are proposed. One of these two approaches provides an iterative solution for the four-point case. Another approach, which is called the "Six-point Algorithm", gives a direct method for the general case.

Secondly, motion-based segmentation is discussed. The aim of this area is to find out all the motions in the scene if there is more than one rigid object undergoing several different motions. Two approaches using the incremental clustering technique are proposed. One of these two approaches considers the motion-based segmentation when the models and the reconstructed 3D information of the objects in the scene are provided. Another one considers the problem when only the

models and the intensity image containing the objects are provided.

Finally, model-based 3D object recognition is discussed. The objective of this research is to recognize an 3D object from a model database by using the technique of computer vision. A two-step approach using the intensity image is proposed. This new approach can recognize the object in the image and correspond it to a template in the 3D object database. Also, the position and orientation of the object can be revealed. This approach uses the "Six-point Algorithm" discussed above to calculate the motion of the object and the result is satisfactory.

Graduate College
The Chinese University of Hong Kong
Shatin, Hong Kong
China

CERTIFICATE OF APPROVAL

M.Phil. THESIS

This is to certify that the M.Phil. thesis of

Man-lee LIU

has been approved by the Examining Committee
for the thesis requirement for the Master of
Philosophy degree in Computer Science and Engineering
at the May 1998 graduation.

Thesis committee: _____
Thesis supervisor

_____
Member

_____
Member

_____
Member

_____
Member

MODEL-BASED COMPUTER VISION:

Motion Analysis, Motion-based Segmentation, 3D Object Recognition.

By

Man-lee LIU

Submitted to the Division of Computer Science and Engineering
in partial fulfillment of the requirements for the
degree of Master of Philosophy
at the
Chinese University of Hong Kong

May 1998

Thesis supervisor: Prof. Kin-hong WONG

ACKNOWLEDGMENTS

I wish to express my deepest gratitude and appreciation to Professor Kin-hong Wong for the guidance, encouragement and support which he provided throughout the course of this research.

I also want to extend my gratitude to Dr. Siu-hang Or for his great help in giving me a lot of useful information about this research.

Moreover, I wish to extend my gratitude to Mr. Lui-yuen Lai, Ming-yee Wong, Siu-tung Wu and Kim-wai Law for their fruitfull help in the preparation and installation of the equipment for the experiment.

In addition, I wish to express my gratitude to Mr. Man-ho Hui and Wilson Lai for great help in the programming of the real-time visual tracking system and 3D object recognition system in Visual C++.

TABLE OF CONTENTS

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

The main objective of this thesis is to discuss the three main areas in model-based computer vision. These three areas are:

1. Model-based motion analysis;
2. Motion-based segmentation;
3. 3D object recognition.

Both theoretical analysis and experiments using synthetic and real data are included in this work.

Model-based computer vision is different from image-based vision. Image-based vision is a form of bottom-up information flow [1][2]. In this case, the solution is computed based on the data implicit in the intensity image. Model-based vision is in a form of top-down information flow [1][2]. In this case, the prior knowledge of objects are used to improve the accuracy and the computation complexity. This kind of prior knowledge is called the "models" of the objects. The model contains features of the corresponding object, and these features are used to represent the object itself. Usually, the model contains features in 3D space. There are many different kinds of features chosen to represent the objects, for example, straight-lines, planes or points on the objects. In this thesis, we will

mainly consider the case using 3D points to represent the models of the objects in 3D space.

In this thesis, we mainly consider the cases with models containing 3D features. Also, all the contributions of this thesis are about the cases with models containing 3D features. However, cases with models containing 2D features will also be discussed for the completeness of this thesis.

In this thesis, we have proposed two algorithms, called the TL-algorithm and Six-point algorithm, for the problem of Model-based Motion Analysis. Also, a real-time visual tracking system is implemented. Moreover, two new algorithms are proposed for the problem of Motion-based Segmentation. In addition, a new algorithm for the problem of 3D Object Recognition is proposed, and a fast 3D object recognition system is implemented.

In this chapter, the three areas in model-based computer vision will be introduced. Section 1.1 gives a brief introduction to the model-based motion analysis. Section 1.2 gives the brief introduction to the motion-based segmentation. Also, the introduction to the 3D Object Recognition will be given in Section 1.3. The organization of this thesis is sketched in Section 1.4.

1.1 Model-based Motion Analysis

Model-based motion analysis is an important research topic in computer vision. In this problem, the motion of a selected object is investigated. In most cases, the position and orientation of the selected object can also be revealed after the motion is obtained. So, model-based motion analysis is sometimes considered to be "pose estimation" of the selected object. In this thesis, the objects used are assumed to be rigid bodies so that the shapes of these objects will not change (i.e. the objects used will not deform) during the process of motion analysis. Moreover,

the motion considered is also rigid so that all the feature points undergo the same motion.

Apart from the model of the selected object used as the input, the information describing the scene of the object is also required. The information can be the 3D coordinations of the feature points of the selected object in that scene. In this case, the 3D coordinations of the feature points in that scene are reconstructed by using stereo vision system or range-finding device. On the other hand, the information can also be the 2D projection of the feature points of the selected object by using a camera.

Since there are different kinds of input, different schemes are required. For the system using 3D coordinations of the feature points as the input, an algorithm of motion analysis with 3D-to-3D point correspondences [3] is required. For the system using 2D projection as the input, an algorithm with 2D-to-3D point correspondences [3] is required. These two kinds of motion analyses are called "*Motion from Structure*" (M.F.S) [4]. That means the motion parameters are calculated by using the 3D structure (i.e. the model) of the objects.

On the other hand, for the system using 2D projection of the feature points of the selected object as both the model and the information describing the scene of the object at an instant, an algorithm with 2D-to-2D point correspondences [3] is required.

The motion analysis in model-based computer vision can be applicable to many areas such as robot navigation and single camera calibration.

The problem of Motion analysis can be classified into 3 categories and will be described below:

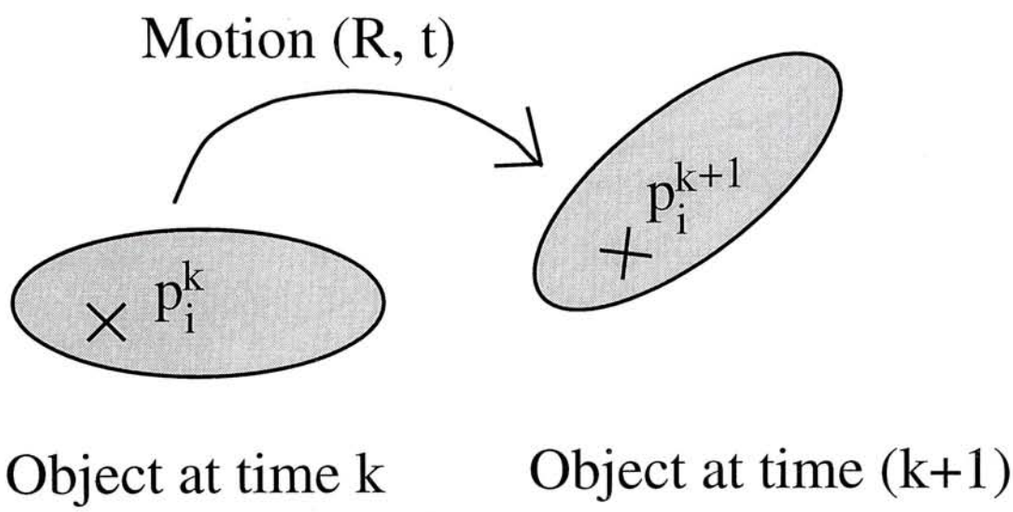


Figure 1: The i-th feature point at two instants when the object undergoes a motion representing by a rotation R and a translation t.

1.1.1 With 3D-to-3D Point Correspondences

Motion analysis with 3D-to-3D correspondences is used in the case when the motion of the selected object is estimated by using the 3D coordinates of feature points at two different instants. For example, N points in 3D space are chosen from the selected rigid object to be the feature points. This object undergoes a rigid motion between time k and $k+1$. The point correspondences of all the N feature points between these two instants are given. At time k, the coordinates of the $i-th$ feature point is $\mathbf{p}_i^k = [X_i^k, Y_i^k, Z_i^k]^T$ for $i = 1, ..., N$. At time $k+1$, the coordinates of the $i-th$ feature point is $\mathbf{p}_i^{k+1} = [X_i^{(k+1)}, Y_i^{(k+1)}, Z_i^{(k+1)}]^T$ for $i = 1, ..., N$. It is shown in Figure (1).

Since the object undergoes a rigid motion, the shape and size of this object remains unchanged. As a result, all the feature points undergo the same motion and the following relation can be obtained:

$$\mathbf{p}_i^{k+1} = \mathbf{R}\mathbf{p}_i^k + \mathbf{t}, \tag{1.1}$$

where $\mathbf{R}$ is a 3×3 orthogonal rotation matrix (i.e. $\mathbf{R}\mathbf{R}^T = \mathbf{R}^T\mathbf{R} = \mathbf{I}$ and

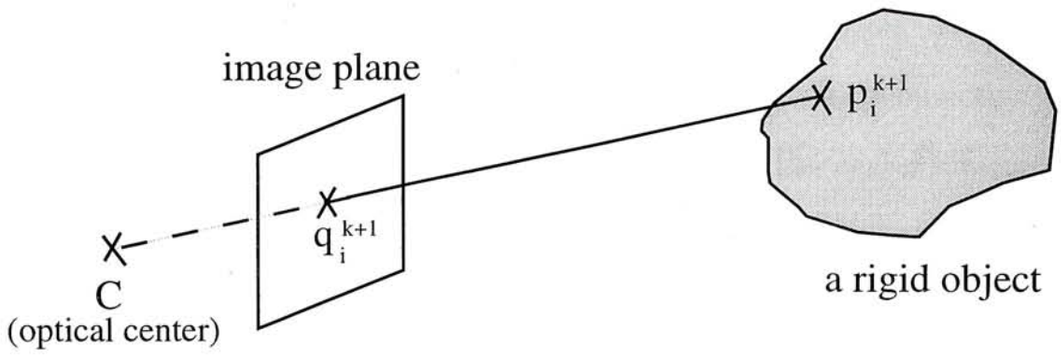


Figure 2: The 3D coordinates and the 2D projection of the i-th feature point at time (k+1) under the full perspective projection.

$det(\mathbf{R}) = 1$) and $\mathbf{t}$ is a 3×1 translation matrix. Therefore, $\mathbf{R}$ and $\mathbf{t}$ represent the rotational part and translational part of the motion respectively.

1.1.2 With 2D-to-3D Point Correspondences

Motion analysis with 2D-to-3D point correspondences is the case when the motion of the selected object is estimated by using the 3D model to represent the selected object and the 2D projection of the feature points of the object at an instant.

Supposing that there are N feature points on the selected object. At time k, the 3D coordinates of the $i-th$ feature point is $\mathbf{p}_i^k$. At time $k+1$, the object is captured by a camera and the 2D projection of the object is obtained. The 2D projection of the $i-th$ feature point at time $k+1$ is $\mathbf{q}_i^{k+1} = [x_i^{k+1}, y_i^{k+1}]^T$. Then we are going to compute the motion (i.e. $\mathbf{R}$ and $\mathbf{t}$) between time k and $k+1$ with a set of 3D coordinates at time k (i.e. model points) and a set of 2D projection (i.e. image points) at time k.

The relation between the coordinates of the $i-th$ feature point at time k ($\mathbf{p}_i^k$) and that at time $k+1$ ($\mathbf{p}_i^{k+1}$) is given in Equation (1.1). However, the 2D projection of the features is depending on what the camera model [5] is used.

In this thesis, we mainly used the pinhole camera model [5]. The projective geometry is full perspective projection [5]. It is because the perspective projection is the accurate projection, and it is widely used in the real situation. Some assumptions should be made when other projective geometries are used, but no assumption is required in the full perspective projection. By using this camera model, the relation between the 3D coordinates of the $i-th$ feature point and its 2D projection at time $k+1$ is shown in Figure (2) and the following equations can be obtained:

$$x_i^{k+1} = X_i^{k+1}\frac{f}{Z_i^{k+1}}, \quad \text{and} \quad y_i^{k+1} = Y_i^{k+1}\frac{f}{Z_i^{k+1}}. \tag{1.2}$$

By using Equations (1.1) and (1.2), the relations between the 3D coordinates, 2D projection and the motion are shown in the following formulas:

$$x_i^{k+1} = f\frac{r_{11}X_i^k + r_{12}Y_i^k + r_{13}Z_i^k + t_x}{r_{31}X_i^k + r_{32}Y_i^k + r_{33}Z_i^k + t_z}, \tag{1.3}$$

and

$$y_i^{k+1} = f\frac{r_{21}X_i^k + r_{22}Y_i^k + r_{23}Z_i^k + t_y}{r_{31}X_i^k + r_{32}Y_i^k + r_{33}Z_i^k + t_z}, \tag{1.4}$$

where r_{mn} is the element at the $m-th$ row and $n-th$ column of the rotation matrix $\mathbf{R}$, $[t_x, t_y, t_z]^T = \mathbf{t}$ is the translation matrix.

With these two equations (1.3) and (1.4), many algorithms are proposed to solve the motion of objects. Besides the approaches using full perspective projection, other algorithms using other projective geometries (such as orthographic projection, weak perspective projection and paraperspective projection [5]) will also be discussed in Chapter 2.

1.1.3 With 2D-to-2D Point Correspondences

Motion analysis with 2D-to-2D point correspondences is the case when the motion of the selected object is estimated by using the 2D projection of the feature

points of the object at two different instants. That means that different views (e.g. an image sequence) are used to calculate the motion.

In this case, the 3D structure of the selected object is unknown. Therefore, the motion and the 3D structure of the object will be calculated simultaneously. However, the depths of all the feature points and the translational parameters can only be determined up to a scale factor under the full perspective projection model [6]. Therefore, the motion analysis with 2D-to-2D point correspondences is different from the previous two cases (i.e. 3D-to-3D, 2D-to-3D). The motion is exactly found in the previous two cases while the translational parameters of the motion in the 2D-to-2D case can only be obtained up to a scale factor.

Since the motion and the structure are determined in the process, this process is called "Motion and Structure from Image Sequences" [7].

1.2 Motion-based Segmentation

In the previous section, the motion analysis in model-based computer vision is briefly introduced. In that section, an assumption is made that there is only one motion in the system. That means that between time k and $k+1$ there are only one rigid object or several objects undergoing one rigid motion. However, in the real world, multiple motions are commonplace (i.e. there are several different rigid objects in the system, and each of them has own motion).

In order to estimate all the motions in the scene with several different objects and motions, different objects with different motions should be separated first. The process of separating different objects according to their motions is called motion-based segmentation. The problem of motion-based segmentation is different from that of the traditional image segmentation [8]. In image segmentation, the partitions are done by considering the intensity, pattern or other image-based features

in an image. However, in the motion-based segmentation, the partitions are done by considering the motions of the rigid objects. Therefore, the techniques of the traditional image segmentation cannot be directly applied to the motion-based segmentation.

The motions of different rigid objects should be calculated after the motion segmentation. However, in the process of motion-based segmentation, the motions of all rigid objects are required as the input. So, it is a complicated problem.

Motion-based segmentation is applicable to solve the cases when there are multiple rigid motions. Also, it may be applied to the coding of video sequences.

1.3 3D Object Recognition

3D object recognition is another hardest problem in computer vision. In this problem, an object is recognized from a database containing many objects. In most cases, the 3D structure of the object to be recognized is considered. In the past, many scholars have proposed a variety of algorithms by using the reconstructed 3D information of the object to be recognized. In this thesis, we mainly consider the 3D object recognition technique by using an intensity image containing the object to be recognized. Therefore, only the 2D projection of the object is required.

On the other hand, Marr [9] claimed that enough 3D information about the objects in the model database is required during the 3D object recognition. Therefore, in this thesis, the models stored in the model database are mainly 3D structures of the objects. However, recognition system using database with different types of models will be presented in the chapter of literature review.

1.4 Organization of the Thesis

In Chapter 2, a literature survey on model-based motion analysis, motion-based segmentation and 3D object recognition will be presented. Also, a summary

of contributions of this thesis will also be presented.

Chapter 3 to 5 provide the main contribution of this thesis.

In Chapter 3, two approaches: TL-algorithm and Six-point algorithm, for the model-based motion analysis with 2D-to-3D correspondences are proposed. Besides the algorithms of these approaches, experiments using both synthetic and real data have been performed to verify these two approaches. Also, a comparison between the TL-algorithm and a famous algorithm proposed by Lowe [10][11][9] will be discussed. On the other hand, a comparison between the Six-point algorithm and another famous linear algorithm proposed by Faugeras [12] will also be presented. Moreover, a visual tracking system by using the Six-point algorithm will also be introduced.

In Chapter 4, two approaches for the motion-based segmentation are proposed. One is for the case with 3D-to-3D point correspondences. Another is for the case with 2D-to-3D point correspondences. Besides, experiments using both synthetic and real data have been done to verify these two approaches.

In Chapter 5, an approach for the 3D object recognition is proposed. Two 3D object recognition systems using the proposed algorithm will also be introduced. Moreover, experiments have been performed to verify the algorithm and the system.

Finally, a conclusion is drawn in Chapter 6 to summarize the work in this thesis.

CHAPTER 2

LITERATURE REVIEW AND SUMMARY OF CONTRIBUTIONS

In this chapter, related work by other scholars will be introduced. In Section 2.1, a review of model-based motion analysis will be introduced. In Section 2.2, a summary of the related work on motion-based segmentation will be presented. In Section 2.3, a survey about 3D object recognition will be given. Finally, Section 2.4 will summarize the contributions of this thesis.

2.1 Model-based Motion Analysis

In this section, the reviews of three divisions of motion analysis in model-based computer vision will be presented.

2.1.1 With 3D-to-3D Point Correspondences

In this problem, the point correspondences of a moving rigid object in two different instants are assumed to be provided. The aim is to determine the rotation $\mathbf{R}$ and translation $\mathbf{t}$. There are totally six unknowns (3 rotation angles and 3 translational parameters). It is known [13] that three noncollinear-point correspondences are sufficient to calculate $\mathbf{R}$ and $\mathbf{t}$ uniquely. By expanding Equation (1.1) with three point correspondences, nine nonlinear equations can be obtained. Iterative methods (e.g. [14]) can be applied to determine the "best" fits of the six unknowns. However, Huang et al. [3] claimed that the answer may be possible to

get stuck in the local minima when iterative methods are used. Linear algorithms are found to be more suitable.

Arun et al. [15] proposed a linear approach which is based on the singular value decomposition (SVD) [16] of a 3×3 matrix. A similar approach is proposed by Horn et al. [17]. It is based on exploiting the orthonormal properties of the rotation matrix. It computes the eigensystem of a different derived matrix. On the other hand, Faugeras et al. [18] also proposed an approach based on quaternions. Moreover, Walker et al. [19] suggested an approach by representing the rotational and translational components as dual quaternions. Also, a detailed survey on these four approaches is done by Eggert et al. [20].

In this thesis, Arun's algorithm [15] has been occupied as the core part of the motion analysis in our proposed solution for the motion-based segmentation with 3D-to-3D correspondences. So, a detailed description of Arun's algorithm will be presented in this thesis.

In their approach, noise is assumed to exist in the set of the 3D coordinates of the feature points. A noise vector $\mathbf{N}_i$, representing the noise of the 3D coordinates of the $i-th$ feature point after the motion, will be added to the Equation (1.1). Then the following equation can be obtained:

$$\mathbf{p}_i^{k+1} = \mathbf{R}\mathbf{p}_i^k + \mathbf{t} + \mathbf{N}_i. \tag{2.5}$$

Then finding the motion parameters becomes finding $\mathbf{R}$ and $\mathbf{t}$ to minimize the following cost function:

$$E = \sum_{i=1}^{N} \left\| \mathbf{p}_i^{k+1} - (\mathbf{R}\mathbf{p}_i^k + \mathbf{t}) \right\|^2. \tag{2.6}$$

In Arun's publication, it is claimed that the motion of the object will be purely rotation if the two centroids of the two sets of 3D coordinates of the feature points on the selected object are the same and equal to the origin of the coordinate

system (i.e. $[0,0,0]^T$). Therefore, the two centroids of the two sets of 3D points are "moved" to the origin by subtracting the corresponding centroid from each 3D point. So, the two centroids of the two sets of 3D coordinates of the feature points at time k and $k+1$ should be obtained as the followings first:

$$\mathbf{c}_k = \frac{1}{N}\sum_{i=1}^{N}\mathbf{p}_i^k, \tag{2.7}$$

$$\mathbf{c}_{k+1} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{p}_i^{k+1}, \tag{2.8}$$

where $\mathbf{c}_k$ and $\mathbf{c}_{k+1}$ are the centroids of the two sets of 3D points at time k and $k+1$ respectively.

Then the original least-squares problem in Equation (2.6) is reduced to the minimization problem as shown in the following cost function:

$$E = \sum_{i=1}^{N}\left\|\mathbf{q}_i^{k+1} - \mathbf{R}\mathbf{q}_i^k\right\|, \tag{2.9}$$

where

$$\mathbf{q}_i^k = \mathbf{p}_i^k - \mathbf{c}_k \tag{2.10}$$

$$\mathbf{q}_i^{k+1} = \mathbf{p}_i^{k+1} - \mathbf{c}_{k+1} \tag{2.11}$$

Then the translation is easily by obtained by the following equation:

$$\mathbf{t} = \mathbf{c}_{k+1} - \mathbf{R}\mathbf{c}_k. \tag{2.12}$$

Since $\mathbf{R}$ is an orthogonal matrix, the cost function in Equation (2.9) can be converted into the following form:

$$E = \sum_{i=1}^{N}\left\|(\mathbf{q}_i^k)^T - (\mathbf{q}_i^{k+1})^T\mathbf{R}\right\|. \tag{2.13}$$

Actually, the minimization problem of the cost function in Equation (2.13) is equivalent to the *orthogonal Procrustes problem* [16]. As shown in [16] that the rotation can be determined by calculating the SVD of $\mathbf{H} = \sum_{i=1}^{N} \mathbf{q}_i^k \left(\mathbf{q}_i^{k+1}\right)^T$ is given as follows:

$$\mathbf{R} = \mathbf{V}\mathbf{U}^T, \tag{2.14}$$

where

$$\mathbf{U}\mathbf{\Lambda}\mathbf{V}^T = \mathbf{H}, \tag{2.15}$$

in which $\mathbf{U}$ and $\mathbf{V}$ are the 3×3 orthogonal matrices, and $\mathbf{\Lambda}$ is the SVD of the $\mathbf{H}$.

As a result the motion of the object can be obtained.

However, the original version of the Arun's algorithm [15] fails to give a correct rotation matrix and returns a reflection instead (i.e. $\det(\mathbf{R}) = -1$) when the data is seriously corrupted. Umeyama [21] then made a modification to Arun's approach. The rotation is given as the followings:

$$\mathbf{R} = \begin{cases} \mathbf{V}\mathbf{U}^T & \text{if } \det(\mathbf{V}\mathbf{U}^T) = +1 \\ \mathbf{V} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \mathbf{U}^T & \text{if } \det(\mathbf{V}\mathbf{U}^T) = -1 \end{cases}.$$

Then the correct transformation parameters will always be obtained even the data is corrupted.

2.1.2 With 2D-to-3D Point Correspondences

The problem is to find the motion when correspondences between 2D image points and 3D model points of a rigid object are given.

Motion analysis with 2D-to-3D point correspondences has been investigated by many scholars for decades, and many methods have been proposed.

There are approaches which make use of the weak perspective projection model. Huttenlocher and Ullman [22] showed that the solution of the pose estimation problem under weak perspective projection using three non-collinear points is unique. Also, Alter [23] introduced a solution for 3D Pose estimation from 3 points under weak perspective projection. Moreover, Gee and Cipolla [24] developed a visual tracking system under weak perspective projection. However, the weak perspective assumption is valid only when the distance between the object and the camera is much larger than the relative distances between feature points on the object.

For the motion analysis with 2D-to-3D point correspondences under full perspective projection, Fischler and Bolles [25] assigned another name for this kind of problem. They called it Perspective-n-Point Problem (PnP), where n is number of point correspondences assigned to the system. In their paper [25], they presented that there will be infinite number of solutions for the PnP problems when n is less than 3. Also, they showed that the number of solutions may be more than one when less than 6 point correspondences are given to the system and these points are in the general 3D positions [25]. It means that multiple solutions may exist for a PnP problem when n is less than 6. For the P3P problem, Wolfe, Mathis, Sklair and Magee [26] also gave a detailed explanation of the formation of multiple solutions under full perspective projection.

For the techniques using full perspective projection, Fischler and Bolles [25] proposed a method by using three non-collinear points. By using this method, three quadratic equations with three unknowns can be obtained. For the four-point case (or so called Perspective-4-Point Problems in [25]), Horaud, Coino, Leboulleux and Lacolle [27] proposed a formulation which is a fourth-order polynomial equation with one unknown. However, the number of model points are restricted (3 in [25]

and 4 in [27]) and multiple solutions may exist in N-point problems for $N < 6$.

Fischler and Bolles [25] showed that using 4 points lying in a common plane will always provide an unique solution. Abidi and Chandra [28] introduced a direct solution using quadrangular targets. It is based on volume measurement of tetrahedra which is composed of target points and the lens center of the camera. Also, Hung, Yeh and Harwood [29] presented an algorithm using quadrangular targets. This method uses the decomposition technique proposed by Ganapathy [30] to compute the final pose of the object. However, the number of model points are still restricted to 4 in these algorithms.

On the other hand, Horn and Weldon [31] proposed a direct method. However, their approach can only handle the cases when:

1. the motion is pure rotation;
2. the motion is pure translation or the rotation component of the motion is known.

However, in the real situation, most motions are the combinations of the rotation and translation. Also, it is not practical to find out the rotation by other method before applying this approach to determine the translation. Therefore, this approach is not general enough.

On the other hand, Quan and Lan [32] proposed another linear algorithm for this problem with 4 or more point correspondences. Their algorithm is based on the Fischler's approach [25]. In their approach, the distance between each two feature points in 3D space. However, wrong solution may be get if only this constraint is being kept. In Figure (3), there are two objects (or models) with the same sides, but the shapes of these two objects are different. By using Quan's approach [32], the result may be the left object in Figure (3) although the original model should

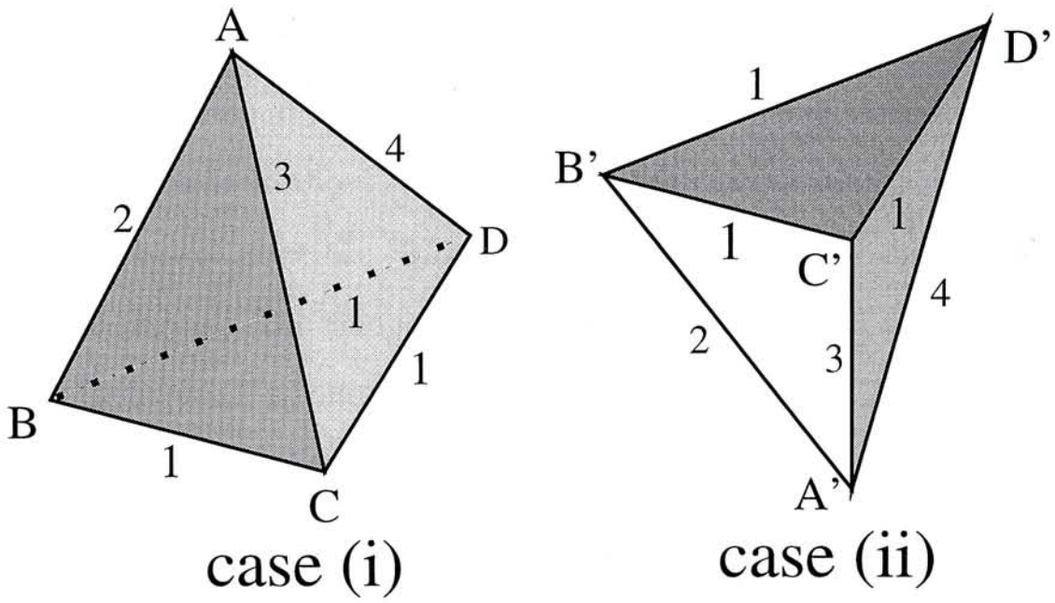


Figure 3: Two models with the same sides (AB=A'B', BC=B'C' and etc...), but with different orientations (i.e. different shapes). In case (i), A is above the plane formed by BCD, while in case (ii), it is under that plane.

be the right one in Figure (3). The same situation may also occur when 5 or more point correspondences are used.

Actually, our TL-algorithm, which will be discussed in Chapter 3, has added another constraint to eliminate the ambiguity for the case with 4 point correspondences. However, Quan's approach [32] did not handle this case. So, the result by using Quan's approach may be incorrect. Also, in their algorithm, the rigidity of the result cannot be ensured since their approach is going to minimize the error between the real distance and the estimated distance within each two feature points, but to get a result guided by the rigidity constraint of the object. Therefore, the result may be deformed (i.e. the shape of the result may not be the same as that of the original object). Moreover, in Quan's approach, the SVD of a matrix $\mathbf{\Pi}$ is required to be calculated. The dimension of this matrix is $\frac{(n-1)(n-2)}{2} \times 5$, where n is the number of point correspondences used. When n becomes larger, the dimension of the matrix $\mathbf{\Pi}$ will also become larger. For example, when 100

point correspondences are used, the dimension of of this matrix becomes 1176×5. The computation time of the SVD of this large matrix using Matlab 5.0 in a Sun Ultra 1/170 machine is about 0.067 second (This computation time is only the time required to calculate the SVD of a huge matrix, but not the time required in the whole process. The computation of SVD is only a small part in the whole process). Therefore, the computation time of this algorithm will increase rapidly with the increase of number of feature points used.

Faugeras also proposed another linear method [12]. This approach is proposed for the purpose of camera calibration [33][34]. Unlike the approach of Quan [32], the intrinsic parameters of the camera [12] can also be found. However, the translational component calculated from this algorithm is very sensitive to noise. This phenomenon is shown by the experiment done by Faugeras himself in his book [12]. In this thesis, an enhancement to the Faugeras's algorithm is proposed to improve the accuracy of the translational result. Also, another algorithm called "Six-Point Algorithm" and its second version are also proposed in this thesis. These two algorithms are similar to the Faugeras's algorithm. Therefore, Faugeras's algorithm [12] will be introduced in this thesis.

Besides, Lowe [11][10][35] proposed a more general solution in which the number of model points and the motion are not restricted. However, this method is nonlinear and requires prior initialization for all unknown parameters. The computation time is depending on the initial guess.

In this thesis, a new iterative approach for the P4P problem is proposed, and we have compared this new approach with Lowe's approach [10]. Therefore, we find it necessary to describe Lowe's approach [10] in this thesis.

2.1.2.1 An Iterative Approach: Lowe's Algorithm

In Lowe's algorithm, Equation (1.3) and (1.4) are converted into the following forms:

$$f\frac{r_{11}X_i^k + r_{12}Y_i^k + r_{13}Z_i^k + t_x}{r_{31}X_i^k + r_{32}Y_i^k + r_{33}Z_i^k + t_z} - x_i^{k+1} = 0, \tag{2.16}$$

$$f\frac{r_{21}X_i^k + r_{22}Y_i^k + r_{23}Z_i^k + t_y}{r_{31}X_i^k + r_{32}Y_i^k + r_{33}Z_i^k + t_z} - y_i^{k+1} = 0 \tag{2.17}$$

where $i = 1, ..., N$ in which N is the total number of feature points used.

Also, the 3×3 rotation matrix is represented by the the three rotation angles about the three axes and it is given as the following:

$$R = \begin{bmatrix} \cos\beta\cos\alpha & -\cos\beta\sin\alpha & \sin\beta \\ \cos\alpha\sin\gamma\sin\beta + \cos\gamma\sin\alpha & \cos\gamma\cos\alpha - \sin\gamma\sin\beta\sin\alpha & -\cos\beta\sin\gamma \\ -\cos\gamma\cos\alpha\sin\beta + \sin\gamma\sin\alpha & \cos\alpha\sin\gamma + \cos\gamma\sin\beta\sin\alpha & \cos\gamma\cos\beta \end{bmatrix}, \tag{2.18}$$

where α, β and γ are the rotation angles about z-axis, y-axis and x-axis respectively.

Since this system involves some trigonometric functions, such as sin, cos and tan. So, it is a non-linear system and too complicated to be solved in direct ways. Therefore the Gauss-Netwon method [36] is used.

Assuming that $\boldsymbol{\zeta}^s = [\gamma^s\,,\ \beta^s\,,\ \alpha^s\,,\ t_x^s\,,\ t_y^s\,,\ t_z^s]^T$ is the vector of parameters after the $s-th$ iteration. Then the error functions we want to minimize should become:

$$e_i = \left(\frac{r_{11}^s X_i^k + r_{12}^s Y_i^k + r_{13}^s Z_i^k + t_x^s}{r_{31}^s X_i^k + r_{32}^s Y_i^k + r_{33}^s Z_i^k + t_z^s} - x_i^{k+1}\right)^2 + \left(\frac{r_{21}^s X_i^k + r_{22}^s Y_i^k + r_{23}^s Z_i^k + t_y^s}{r_{31}^s X_i^k + r_{32}^s Y_i^k + r_{33}^s Z_i^k + t_z^s} - y_i^{k+1}\right)^2. \tag{2.19}$$

Supposing that $\mathbf{E}$ is the vector of all the error functions, e_i for $i = 1, ..., N$. Also, $\mathbf{J}$ is the Jacobian matrix where $\mathbf{J}_{mn} = \frac{\partial \mathbf{E}_m}{\partial \boldsymbol{\zeta}_n}$. Then by the Gauss-Netwon method [36], the following update rule of the parameters can be obtained:

$$\boldsymbol{\zeta}^{s+1} = \boldsymbol{\zeta}^s - \mathbf{h}_L, \tag{2.20}$$

where $\mathbf{h}_L$ is the vector of correction which can be solved by the following equation:

$$\mathbf{h}_L = (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{E}. \tag{2.21}$$

By applying the update rules (Equation (2.20) and (2.21)) iteratively with a suitable initial guess and threshold, the answer can be obtained.

Also, Lowe included the concept of stabilization in his algorithm. In [11], a stabilizer is added in order to solve the ill-conditioning problem such as the case where the object is projected approximately orthographically. After adding the stabilizer, Equation (2.21) is replaced by the following one:

$$\mathbf{h}_L = (\mathbf{J}^T\mathbf{J} + \mathbf{W}^T\mathbf{W})^{-1}\mathbf{J}^T\mathbf{E},$$

where $\mathbf{W}$ is a diagonal matrix in which each weight is inversely proportional to the standard deviation σ_j for the parameter j:

$$\mathbf{W}_{jj} = \frac{1}{\sigma_j}.$$

2.1.2.2 A Linear Approach: Faugeras's Algorithm

In Faugeras's approach [12], the Equation (2.16) and (2.17) are converted into the following homogeneous linear equations:

$$\boldsymbol{\chi}\mathbf{q} = \mathbf{0}, \tag{2.22}$$

where $\boldsymbol{\chi}$ is a $2N \times 12$ matrix and $\mathbf{q}$ is a 12×1 vector. The $(2i-1)-th$ row of $\boldsymbol{\chi}$ is

$$[fX_i^k\,,\ fY_i^k\,,\ fZ_i^k\,,\ f\,,\ 0\,,\ 0\,,\ 0\,,\ 0\,,\ -x_i^{k+1}X_i^k\,,\ -x_i^{k+1}Y_i^k\,,\ -x_i^{k+1}Z_i^k\,,\ -x_i^{k+1}], \tag{2.23}$$

and the $2i-th$ row is

$$[0\,,\ 0\,,\ 0\,,\ 0\,,\ fX_i^k\,,\ fY_i^k\,,\ fZ_i^k\,,\ f\,,\ -y_i^{k+1}X_i^k\,,\ -y_i^{k+1}Y_i^k\,,\ -y_i^{k+1}Z_i^k\,,\ -y_i^{k+1}], \tag{2.24}$$

where $i = 1, ..., N$, for N is the number of point correspondences used. Also, $\mathbf{q}$ is given as following:

$$\mathbf{q} = [\mathbf{r}_1\ ,\ t_x\ ,\ \mathbf{r}_2\ ,\ t_y\ ,\ \mathbf{r}_3\ ,\ t_z]^T, \tag{2.25}$$

in which $\mathbf{r}_j$ is the $j-th$ row vector of the rotation matrix (Also, $\mathbf{r}_3 = [r_{31}, r_{32}, r_{33}]^T$ etc.).

In his book [12], it is shown that the rank of $\boldsymbol{\chi}$ is usually equal to 11. Therefore there exists a non zero vector $\mathbf{q}$ of the 12 unknowns that is defined up to a scaling factor. Also, since the norm of the 3rd row vector of the rotation matrix is equal to 1 (i.e. $r_{31}^2 + r_{32}^2 + r_{33}^2 = 1$). Then the whole problem is converted to the following problem:

$$\underset{q}{\text{Min}} \|\boldsymbol{\chi}\mathbf{q}\|^2 \quad \text{subject to } \|\mathbf{r}_3\|^2 = 1. \tag{2.26}$$

This constrained optimization problem can be solved by using the technique described in Appendix B.

However, the rotation matrix obtained only after solving this constrained optimization problem may not be ensured to be orthogonal.

Then the result should be refined in order to fulfill the orthogonality of the rotation matrix. In this approach, five intrinsic parameters are introduced. They are:

1. u_0 is the horizontal shift (in x direction) of the optical center;
2. v_0 is the vertical shift (in y direction) of the optical center;
3. α_u is the magnifying ratio of the coordinates along horizontal direction (x direction);
4. α_v is the magnifying ratio of the coordinates along vertical direction (y direction);

5. θ is the actual angle between the two axes of the image coordinate system (i.e. the actual angle between the x-axis and y-axis of the image coordinate system).

Actually, the optical center may not be the same as the origin of the image coordinate system and θ may not be 90^oaccording to the error during the production of the camera (e.g. mistake in placing the CCD in the camera).

Supposing the result of the minimization in Equation (2.26) is $\mathbf{q}_e$=[$\mathbf{q}_1, q_{14}, \mathbf{q}_2, q_{24}, \mathbf{q}_3, q_{34}$ Then the correct motion parameters can be obtained as the following:

$$\begin{aligned} \mathbf{r}_3 &= \varepsilon \mathbf{q}_3, \\ u_0 &= \mathbf{q}_1 \mathbf{q}_3^T, \\ v_0 &= \mathbf{q}_2 \mathbf{q}_3^T, \\ t_z &= \varepsilon q_{34}. \end{aligned} \tag{2.27}$$

Also,

$$\cos\theta = -\varepsilon_u \varepsilon_v \frac{(\mathbf{q}_1 \times \mathbf{q}_3) \cdot (\mathbf{q}_2 \times \mathbf{q}_3)}{\|\mathbf{q}_1 \times \mathbf{q}_3\| \, \|\mathbf{q}_2 \times \mathbf{q}_3\|}, \tag{2.28}$$

where θ is restricted between 0 and π in order to keep the uniqueness of the result. Also, ε, ε_u and ε_v are either $+1$ or -1. Then the other parameters are determined as follows:

$$\begin{aligned} \alpha_u &= \varepsilon_u \|\mathbf{q}_1 \times \mathbf{q}_3\| \sin\theta, \\ \alpha_v &= \varepsilon_v \|\mathbf{q}_2 \times \mathbf{q}_3\| \sin\theta, \\ \mathbf{r}_2 &= \varepsilon(\mathbf{q}_2 - v_0 \mathbf{q}_3)\frac{\sin\theta}{\alpha_v}, \\ \mathbf{r}_1 &= \varepsilon(\mathbf{q}_1 + (\mathbf{q}_2 - v_0 \mathbf{q}_3)\frac{\alpha_u}{\alpha_v}\cos\theta - u_0 \mathbf{q}_3)\frac{1}{\alpha_u}, \\ t_y &= \varepsilon(q_{24} - v_0 q_{34})\frac{\sin\theta}{\alpha_v}, \\ t_x &= \varepsilon(q_{14} + (q_{24} - v_0 q_{34})\frac{\alpha_u}{\alpha_v}\cos\theta - u_0 q_{34})\frac{1}{\alpha_u}. \end{aligned} \tag{2.29}$$

The determinant of $\mathbf{R}$ is easily shown to be equal to $\varepsilon\varepsilon_u\varepsilon_v \text{sign}(\det((\mathbf{q}_1^T, \mathbf{q}_2^T, \mathbf{q}_3^T)^T))$. There are therefore four possibilities corresponding to the four choices of ε, ε_u and ε_v such that the determinant of the rotation matrix is equal to 1. Then the motion of the object is obtained.

In [12], Faugeras showed that for N coplanar points ($N \geq 4$) in general position, the rank of $\boldsymbol{\chi}$ is equal to 8. Therefore, the 3-D reference points used for motion estimation should not be chosen to lie in the same plane since this approach for this case will not work.

2.1.3 With 2D-to-2D Point Correspondences

The problem is to determine the motion of a rigid object between two instants when 2D point correspondences of this rigid object are given.

In this problem, not only the motion parameters (i.e. $\mathbf{R}$ and $\mathbf{t}$) should be estimated, the 3D structure of the object is also required to be determined too. In the past, many methods were proposed. Some of them are based on the orthographic projective model and some others are based on the perspective model.

For the problem with the orthographic projective model, Ullman [4] showed that an unique solution to motion and 3D structure up to a reflection is sufficient to be obtained by using four point correspondences over three frames. Later, Huang et al. [37] proposed a linear algorithm to obtain the solution in this problem. Also, they proved that using three point correspondences over three frames, there may be up to 16 solutions for the motion and four for the 3D structure plus their reflections. Moreover, Aizawa et al. [38] proposed a simple two-step iterative method to reconstruct the motion and the 3D structure. This method was used as a part of their MBASIC algorithm for 3D model-based compression of video. However, Bozdagi [39] stated that this two-step approach is sensitive to random errors in the initial depth estimates, and he proposed an enhancement for this

two-step approach. The comparison of the performance of the originial method and the improved algorithm is given in [39].

For the problem with full perspective projective model, Longuet-Higgins [40] made a great break-through on the motion and structure reconstruction in computer vision. He proposed a two-step linear algorithm for the problem under full perspective projective model. In his algorithm, at least eight point correspondences are required to determine the essential matrix [5] which contains the rotation and translation (up to a scale for the translation). This algorithm is sometimes called the "Eight-point Algorithm" [41]. Later, Toscani and Faugeras [42] proposed to use quaternions in estimating the rotation matrix from the essential matrix in order to obtain an accurate result from noisy data. In this sub-section, the modified version of this two-step linear approach proposed by Toscani and Faugeras is introduced. This algorithm contains two steps:

1. Calculation of the essential matrix;
2. Extracting the motion parameters (i.e. rotation and translation) from the essential matrix.

Recalling Equation (1.1), $\mathbf{p}_i^k$ and $\mathbf{p}_i^{k+1}$ are the 3D coordinates of the $i-th$ feature point at time k and $k+1$ respectively. Also, $\mathbf{R}$ and $\mathbf{t}$ are respectively the rotational and translational parameters of the motion of the object between time k and $k+1$.

By observing Figure (4), the vectors $\mathbf{p}_i^{k+1}$, $\mathbf{t}$ and $\mathbf{R}\mathbf{p}_i^k$ are coplanar. Since $(\mathbf{t} \times \mathbf{R}\mathbf{p}_i^k)$ is orthogonal to this plane, the following relation can be obtained:

$$\mathbf{p}_i^{k+1} \cdot (\mathbf{t} \times \mathbf{R}\mathbf{p}_i^k) = 0, \tag{2.30}$$

where $\times$ and $\cdot$ stand for the vector product and dot product of vectors. Then by Equation (2.30), we can get:

$$\mathbf{p}_i^{k+1} \cdot \mathbf{E}\mathbf{p}_i^k = 0, \tag{2.31}$$

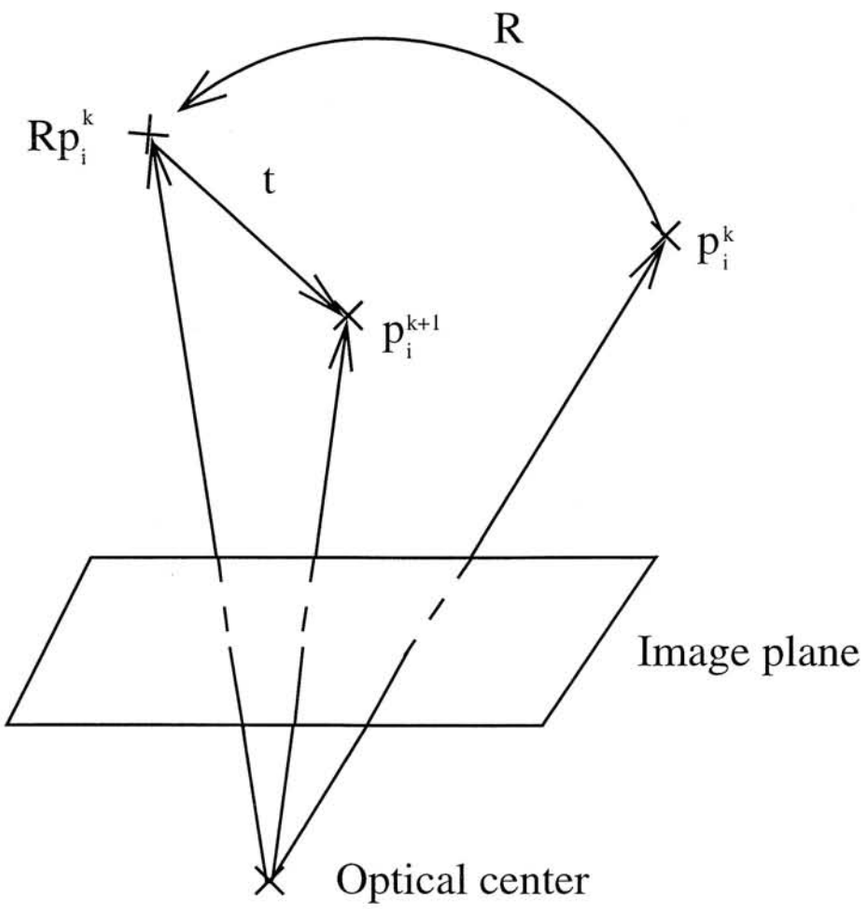


Figure 4: The change of the coordinates of the i-th feature point between time k and (k+1) after a motion (R and t).

where $\mathbf{E}$ is the essential matrix and it is given as the followings:

$$\mathbf{E} = \begin{bmatrix} e_1 & e_2 & e_3 \\ e_4 & e_5 & e_6 \\ e_7 & e_8 & e_9 \end{bmatrix} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \mathbf{R}. \tag{2.32}$$

Then we divide both sides of Equation (2.31) by Z_i^k and Z_i^{k+1} to obtain a relationship in terms of the image coordinates. By using Equation (1.2) and (1.3). We get:

$$\begin{bmatrix} \dot{x}_i^{k+1} & \dot{y}_i^{k+1} & 1 \end{bmatrix} \mathbf{E} \begin{bmatrix} \dot{x}_i^k \\ \dot{y}_i^k \\ 1 \end{bmatrix} = 0, \tag{2.33}$$

where

$$\dot{x}_i^j = \frac{x_i^j}{f} \quad \text{and} \quad \dot{y}_i^j = \frac{y_i^j}{f}, \tag{2.34}$$

for $j = k$ or $k+1$. and we assign $\dot{\mathbf{q}}_i^j = \begin{bmatrix} \dot{x}_i^j & \dot{y}_i^j & 1 \end{bmatrix}^T$.

Therefore, there are totally N equations similar to Equation (2.33) for N feature points. By stacking all these N equations together, we get:

$$\mathbf{Ge} = \mathbf{0}, \tag{2.35}$$

where $\mathbf{G}$ and $\mathbf{e}$ are respectively $N \times 9$ and 9×1 matrices and they are given as follows:

$$\mathbf{G} = \begin{bmatrix} \dot{x}_1^{k+1}\dot{x}_1^k & \dot{y}_1^{k+1}\dot{x}_1^k & \dot{x}_1^k & \dot{x}_1^{k+1}\dot{y}_1^k & \dot{y}_1^{k+1}\dot{y}_1^k & \dot{y}_1^k & \dot{x}_1^{k+1} & \dot{y}_1^{k+1} & 1 \\ \vdots & & & \vdots & & & & & \vdots \\ \dot{x}_N^{k+1}\dot{x}_N^k & \dot{y}_N^{k+1}\dot{x}_N^k & \dot{x}_N^k & \dot{x}_N^{k+1}\dot{y}_N^k & \dot{y}_N^{k+1}\dot{y}_N^k & \dot{y}_N^k & \dot{x}_N^{k+1} & \dot{y}_N^{k+1} & 1 \end{bmatrix}, \tag{2.36}$$

and $\mathbf{e}$ contains the 9 essential parameters which is:

$$\mathbf{e} = \begin{bmatrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 \end{bmatrix}^T. \tag{2.37}$$

Obviously, $\mathbf{e}$ can only be determined up to a scale factor. We can normalize translation vector to 1. It is reasonable and it is mentioned in Chapter 1 that the translation can only be found up to a scale factor. Then we have $\|\mathbf{E}\|^2 = \|\mathbf{e}\|^2 = 2$. Hence, we can obtain $\mathbf{e}$ by solving the following minimization problem:

$$\min_{\mathbf{e}} \|\mathbf{G}\mathbf{e}\|^2, \quad \text{subject to } \|\mathbf{e}\|^2 = 2. \tag{2.38}$$

The solution of $\mathbf{E}$ is the eigenvector of $\mathbf{G}^T\mathbf{G}$ of norm $\sqrt{2}$ corresponding to the smallest eigenvalue. Therefore the essential matrix $\mathbf{E}$ can be obtained. Afterwards, we can determine the unit vector of the translation $\mathbf{t}_s$ by solving the following minimization problem:

$$\min_{\mathbf{t}} \left\|\mathbf{E}^T\mathbf{t}_s\right\|, \quad \text{subject to } \|\mathbf{t}_s\| = 1. \tag{2.39}$$

The solution of $\mathbf{t}_s$ is an unit eigenvector of $\mathbf{E}\mathbf{E}^T$ corresponding to the smallest eigenvalue. If

$$\sum_{i=1}^{N} (\mathbf{t}_s \times \dot{\mathbf{q}}_i^k) \cdot (\mathbf{E}\dot{\mathbf{q}}_i^{k+1}) < 0,$$

then $\mathbf{t}_s \leftarrow -\mathbf{t}_s$.

Afterwards, we can determine the rotation matrix $\mathbf{R}_s$. As $\mathbf{E} = \mathbf{t}_{ms}\mathbf{R}_s$ by definition for

$$\mathbf{t}_{ms} = \begin{bmatrix} 0 & -t_{sz} & t_{sy} \\ t_{sz} & 0 & -t_{sx} \\ -t_{sy} & t_{sx} & 0 \end{bmatrix},$$

where $\begin{bmatrix} t_{sx} & t_{sy} & t_{sz} \end{bmatrix}^T = \mathbf{t}_s$.

Then we can find $\mathbf{R}_s$ by solving

$$\min_{\mathbf{R}_s} \|\mathbf{E} - \mathbf{t}_{ms}\mathbf{R}_s\|^2 \quad \text{subject to } \mathbf{R}_s^T\mathbf{R}_s = \mathbf{I} \text{ and } \det(\mathbf{R}_s)=1. \tag{2.40}$$

Then the above minimization problem can be solved by using the quaternion [43] representation of 3D rotations [26]. Therefore, the motion parameters can be

determined. For more details of the quaternion representation of 3D rotations, readers can refer to Appendix A.

On the other hand, Philip [44] proposed another approach to extract the motion parameters from the essential matrix by using the *singular value decomposition (SVD)* method [16].

However, this approach is quite sensitive to noise [7], for example, spatial quantization, feature detector errors, point mismatching and camera miscalibration. Weng, Huang and Ahuja [7] provided a systematic error estimation about this algorithm. Also, applications and properties are discussed in detailed in [5][41][45][7]. On the other hand, Zhang [46] proposed an interesting framework to reconstruct the motion and 3D structure by using four point correspondences from a motion of a stereo rig.

On the other hand, the fundamental matrix [5][41] is also a famous tool for this problem (especially, for establishing epipolar geometry [5] with uncalibrated cameras). The relation between a fundamental matrix $\mathbf{F}$ and an essential matrix $\mathbf{E}$ is given as follows:

$$\mathbf{F} = \mathbf{C}^{-T}\mathbf{E}\mathbf{D}^{-1},$$

where $\mathbf{C}$ and $\mathbf{D}$ are intrinsic matrices [5] of the cameras.

2.2 Motion-based Segmentation

Many algorithms for motion analysis in model-based computer vision are discussed in Section 2.1. All of them are based on the assumption that there is only one rigid motion in the scene. However, in the real situations, different rigid objects may undergo different motions. These algorithms can give significant result only if the features of the same motion are segmented first. Therefore, motion-based segmentation technique is required. In this section, a review for

motion-based segmentation is presented.

An algorithm of 3D motion computation and object segmentation is proposed by Zhang and Faugeras [47]. In their algorithm, 3D line-based features are used. The kinematic parameters are used to represent the motion of the objects, and a technique based on the Mahalanobis distance is used to perform the object segmentation. Also, a visual tracking system for tracking multiple moving objects was proposed by Hung et al. [48]. The 3D coordinates of the feature points are obtained with a stereo vision system. Then a RANSAC[1]-based clustering method is applied to perform the motion-based segmentation, and the motion is determined by using the Arun method [15]. Moreover, Wang and Duncan [49] proposed an iterative method which is based on the information about the velocities of the objects. Another method to solve the problem of multiple 3D rigid motions is proposed by Tao et al. [50]. It converts the estimation of multiple rigid motions into a maximum likelihood estimation procedure.

On the other hand, Jacobs et al. [51] proposed another RANSAC-based algorithm by making use of the weak perspective projection model. Xu et al. [5][52] proposed a solution by using epipolar geometry under the affine projection model. Besides, some other scholars [53][54] make use of optical flow to solve the motion segmentation problem.

2.3 3D Object Recognition

3D object recognition is one of the important problems and research topics in computer vision. In this thesis, we are going to deal with the problem that an object in an image is recognized by matching against the objects in the database according to their shape. In the past, many approaches are proposed by other

[1]RANSAC stands for "Random Sample Consensus" is a technique to fit a model to experimental data and it is proposed by Fischler and Bolles [25].

scholars.

Dorai et al. [55] presented a recognition system using images obtained from many different viewpoints as the models of the 3D objects in the database. However, in their approach, the object to be recognized is in the form of range image. So, a range-finding device is required. Schutz and Hugli [56] proposed another approach using range image with the cooperation of the ICP algorithm. Also, Gingins and Hugli [57] proposed a hybrid method by using the range image and intensity image. In this approach, intensity image is used to verify the hypothesis due to the range image. Therefore, in these algorithms, the three-dimensional structure of the object must be reconstructed first.

Silberberg et al. [58] presented a 2-stage algorithm that recognizes one or more 3D objects in an image by using the oriented model points. However, the orientation and the distance of the ground plane are assumed to be known in their approach. Piggio et al. [59] proposed to use a number of images form different viewpoint to train a neural network. The internal representation functions trained are used to represent the objects models. Then these models are used in the object recognition. Also, Kuno et al. [60] used a tree-like feature search strategy by using a number of images of the 3D objects in different viewpoints. Moreover, Xu and Zhang [5] proposed to perform 3D object recognition by using the technique of Epipolar geometry [5][46][45] to match model view with an input view. In addition, invariant feature is also used in 3D object recognition [61][62]. Although the 3D structure of the object need not to be reconstructed beforehand, the detailed pose information of the object cannot be obtained from them.

Also, Lowe [10][11] proposed an approach to recognize the 3D object from a single 2D intensity image, and the pose is also calculated as well. In his approach,

the iterative technique for the motion analysis with 2D-to-3D point correspondences proposed by Lowe [10] himself (which is introduced in Section 2.1.2) is applied as the core to determine the motion of the object in the scene.

2.4 Summary of Contributions

The contributions of this thesis are briefly summarised below:

1. An Enhancement of Faugeras's Algorithm:

   An enhancement to the Faugeras's algorithm [12] is proposed in order to improve the accuracy of the result of the translation matrix. Experiments have been done to compare the performance of this modified version and the original Faugeras's algorithm in terms of the accuracy of the translational parts of the result. From, the results, it is observed that the accuracy of this modified version is much better than that of the original Faugeras's algorithm.

   A publication about this work is prepared for a journal publication [63].

2. A new Iterative Method: TL-algorithm:

   A new iterative algorithm which is called the "TL-algorithm" is proposed for the P4P problems. Experiments have been done to compare the performance of the TL-algorithm and that of Lowe's algorithm [10] in terms of their computation times. From the results, it is observed that the computation speed of TL-algorithm is faster than that of Lowe's algorithm, especially for the case when the motion is pure rotation.

   Publication about this work is accepted by *3DCV'97* [64]. Also, another publication about this work is submitted to *Pattern Recognition Letters* and under revision [65].

3. A new Linear Method: Six-point Algorithm:

   A two-step linear algorithm which is called the "Six-point Algorithm" is proposed for PnP problems with $n \geq 6$. There are totally two versions of the "Six-point Algorithm". These two versions of "Six-point Algorithm" and Faugeras's algorithm [12] are quite similar. They mainly differ by using different constraints in the constrained optimization. Also, we propose to use the technique of representation of rotations by quaternion and the technique of SVD [16] to estimate the rotation of the object. It can be applicable to any PnP problems for $n \geq 6$ when *not all the feature points (in 3D space) lie in the same plane.*

   Also, the Six-point algorithm is much more general than Horn's approach [31] since the Six-point algorithm can handle any kind of rigid motion. It means that the Six-point algorithm can be applicable to the case even if the motion is neither pure rotation nor pure translation. Also, the technique of SVD is applied in the estimation of rotation matrix. From the experimental result, the use of the technique of SVD can improve the computation speed. Moreover, experiments have done to verified our proposed approach. Besides, there is a comparison between Faugeras's algorithm and the two versions of the "Six-point Algorithm". In addition, a visual tracking system is implemented by using the Six-point algorithm as the core to estimate the motion of the moving object. Unlike the visual tracking system proposed by Gee [24], our system makes use of the Six-point algorithm and the camera model is full perspective instead of weak perspective in that of Gee. Besides, Six-point algorithm is used as a part of a motion-based segmentation algorithm and a 3D object recognition algorithm proposed by us. These two algorithms will also be discussed in this thesis.

   A publication about this work is prepared for a journal publication [63].

4. A new Algorithm for Motion-based Segmentation with 3D-to-3D Point Correspondences:

   Moreover, a motion-based segmentation algorithm for the case using 3D-to-3D point correspondences is proposed. This new approach makes use of incremental clustering to segment all the feature points into a number of clusters. The feature points in each cluster have the same motion (same $\mathbf{R}$ and $\mathbf{t}$). Since, the incremental clustering technique is applied, the number of clusters can also be determined. Therefore, prior knowledge of the number of motions is not required. Afterwards, the motion for each cluster will be calculated. Arun's method [15] is occupied as the core of motion analysis.
   Publications about this work are accepted by *ICPR'98* [66] and *NOLTA'98* [67].

5. A new Algorithm for Motion-based Segmentation with 2D-to-3D Point Correspondences:

   On the other hand, a motion-based segmentation algorithm for the case using 2D-to-3D point correspondences is proposed too. It is a similar approach to our another proposed approach for the case of 3D-to-3D. The incremental clustering method is also used. However, the Six-point algorithm is occupied as the core of motion analysis, such that this algorithm is suitable to solve the motion-based segmentation problem for the case of 2D-to-3D. Experiments have been performed to verify these two new approaches.
   A publication about this work is accepted by *ICIPS'98* [68].

6. A new Algorithm for Motion-based 3D Object Recognition from an Intensity Image:

   Finally, an algorithm for 3D object recognition is proposed. It is a two-step approach to recognize 3D objects from a single intensity image with an

unknown viewpoint. In this approach, the object in the image is recognized by matching against the objects in the database according to their shape. This new approach is unlike others since the 3D information of the object to be recognized need not be reconstructed and an image is sufficient to recognize the object according to its 3D shape. The most similar approach is proposed by Lowe [10]. However, our approach uses the Six-point algorithm as the core of the motion analysis instead of the iterative method in Lowe's approach [10]. Therefore, the time for recognition is shortened.

A publication about this work is submitted to *98 Workshop on Computer Vision* [69].

CHAPTER 3

MODEL-BASED MOTION ANALYSIS WITH 2D-TO-3D POINT CORRESPONDENCES

In this chapter two approaches for model-based motion analysis with 2D-to-3D point correspondences are proposed. The first one is an iterative method specific for the P4P problem. It is called the TL-algorithm. Another is a linear algorithm for any PnP (for $n \geq 6$) problems with general motion (i.e. the motion need not be pure rotation nor pure translation). This linear algorithm is called the Six-point algorithm. Actually, there are two versions of the Six-point algorithm and they will be described in this chapter. Also, a visual tracking system by using the Six-point algorithm will also be described.

In addition, an enhancement of Faugeras's algorithm [12] is proposed to improve the accuracy of the result of the translation matrix. Moreover, a comparison between the two versions of Six-point algorithm, the original Faugeras's algorithm and the modified Faugeras's algorithm will be presented.

Also, Figure 5 describes a brief summary of different methods in Model-based Motion Analysis.

3.1 A new Iterative Algorithm for the Perspective-4-point Problem: TL-algorithm

In this section, the TL-algorithm is described. In Figure (6), the motion of this object is being investigated. On this object, 4 feature points (in 3D space)

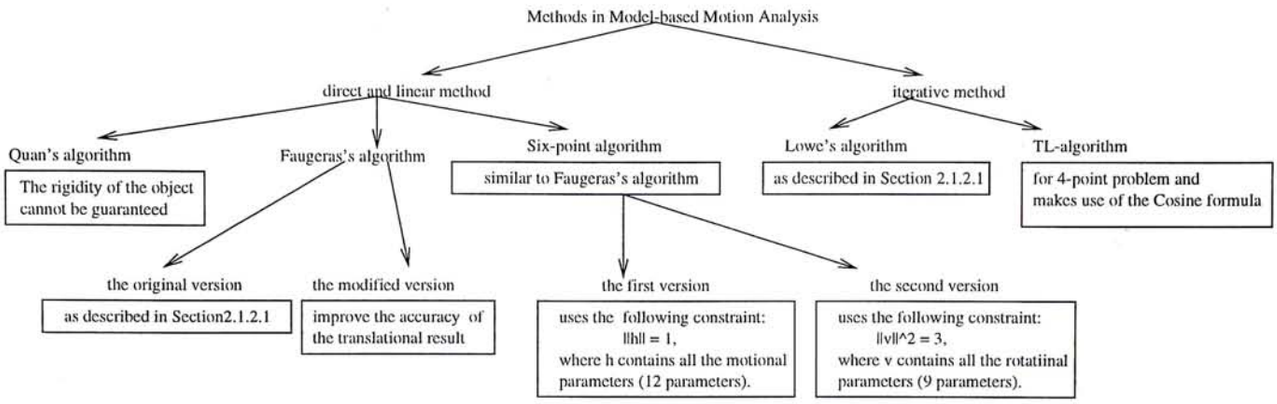


Figure 5: A diagram illustrates the relation between different methods in Model-based Motion Analysis.

are extracted (which are called model points). Since it is a model-based problem, the distances between each extracted point should be measured before motion tracking. The distances measured will form a model to represent the object. Then in the process of motion tracking, a camera is used to capture the images of that rigid object. The 4 model points will project to the image plane (i.e. screen of the camera) to form 4 image points (i.e. projection of the feature points) as shown in Figure (6).

3.1.1 Algorithm

Referring to Figure (6), the relations between the model points at time k and corresponding image points can be expressed in the following way:

$$\overrightarrow{PA_i} = l_i \overrightarrow{u_i}, \tag{3.41}$$

where

$$\overrightarrow{u_i} = \frac{\overrightarrow{Pa_i}}{\left\| \overrightarrow{Pa_i} \right\|}, \tag{3.42}$$

in which A_i and a_i are respectively the $i-th$ model point and $i-th$ image point. Also, i=1,2,3 or 4, l_i=distances between the model points A_i and the optical center. Moreover, $\overrightarrow{Pa_i} = [x_i^k\ ,\ y_i^k\ ,\ f]^T$ where $[x_i^k, y_i^k]^T$ is the coordinates of projection $i-th$

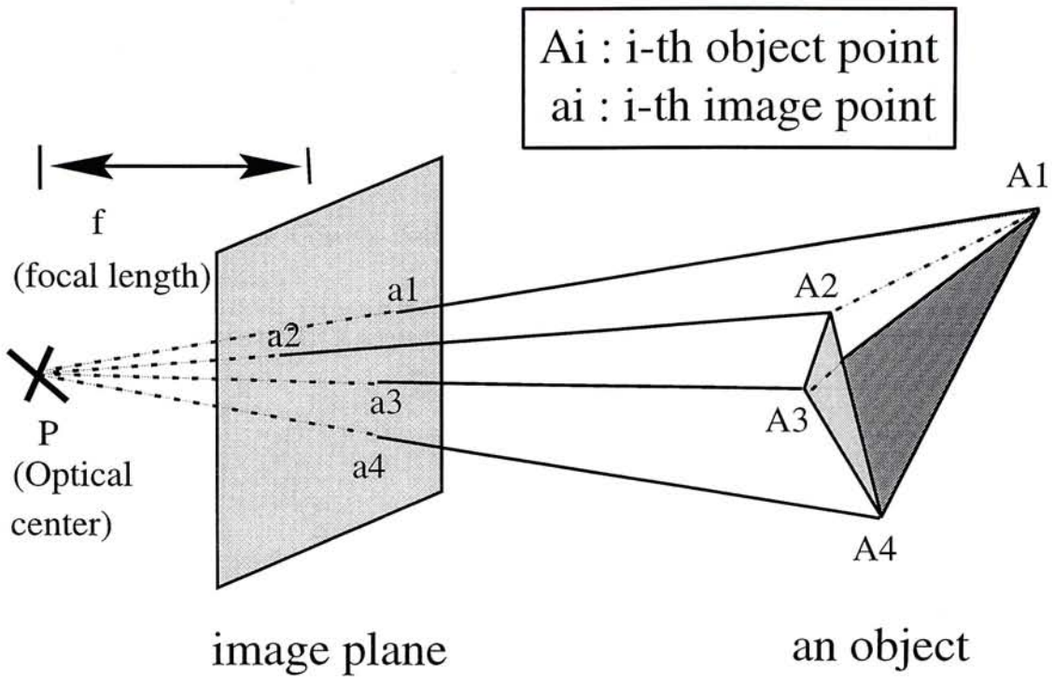


Figure 6: a model with 4 model points and their corresponding image points

feature point and f is the focal length. Also, applying the constraints of the distances between each model point can obtain the following cost function:

$$e^s(i,j) = ((l_i^s)^2 + (l_j^s)^2 - 2l_i^s l_j^s(\overrightarrow{u_i} \cdot \overrightarrow{u_j})) - \left|\overrightarrow{A_i A_j}\right|^2, \tag{3.43}$$

where $i, j = 1, 2, 3$ or 4 with $i < j$.

Therefore, there are totally six cost functions (in the form of equation (3.43) with different combinations of i and j). However, it is not enough to reconstruct the model by these six constraints only. For example, Figure (3) shows there are two sets of model points that can also satisfy the same set of the six constraints. In order to perserve the shape of the model so that the model can be reconstructed uniquely, another constraint is required and it is given as the following cost function:

$$\begin{aligned} g^s &= [(l_2^s\overrightarrow{u_2} - l_3^s\overrightarrow{u_3}) \times (l_4^s\overrightarrow{u_4} - l_3^s\overrightarrow{u_3})] \cdot (l_1^s\overrightarrow{u_1} - l_3^s\overrightarrow{u_3}) \\ &\quad -(\overrightarrow{A_3A_2} \times \overrightarrow{A_3A_4}) \cdot \overrightarrow{A_3A_1} \end{aligned} \tag{3.44}$$

Assuming that $\mathbf{E}^s$ is the vector of all $e^s(i,j)$ and g^s after $s-th$ iteration (i.e. $\mathbf{E}^s$=$[e^s(1,2), e^s(1,3), e^s(1,4), e^s(2,3), e^s(2,4), e^s(3,4), g^s]^T$). $\mathbf{J}$ is the Jacobian matrix, $\mathbf{J}_{mn} = \frac{\partial \mathbf{E}_m^s}{\partial l_n^s}$. Also, $\mathbf{L}^s$ is the column vector of all l_i^s(i.e. $\mathbf{L}_i^s = l_i^s$ for $i = 1, ..., 4$). Then by the Gauss-Netwon method [36], we have the following update rule to minimize the cost function:

$$\mathbf{L}^{s+1} = \mathbf{L}^s - \boldsymbol{\kappa}, \tag{3.45}$$

where

$$\mathbf{J}\boldsymbol{\kappa} = \mathbf{E}, \tag{3.46}$$

in which

$$\boldsymbol{\kappa} = (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{E}. \tag{3.47}$$

By applying equations (3.45) and (3.47) iteratively with a suitable initial guess (which should be closed to the answer), the result can be obtained until the stopping criteria is reached (e.g. $\boldsymbol{\kappa}$ less than a threshold or $\mathbf{E}$ less than a threshold).

The selection of initial guess not only affect the computation time, but the accuracy of the result may also be influenced. As mentioned in Section 2.1.2 that multiple solutions may exist for a P4P problem [25]. If the initial guess is not close enough to the correct answer, the system may converge to another local minimum.

From the experiment, we found that a suitable initial guess of the lengths at time $k+1$ should be the estimated lengths at time k (i.e. result at time k). It is reasonable since the motion of the object may not be so large between time k and $k+1$ if the time interval is short enough.

3.1.2 Experiment

Both synthetic and real data are used to examine our model-based pose estimation algorithm. Both experiments are done on a SGI-INDY computer. Also,

all the programs are implemented in MATLAB 4.2c.

3.1.2.1 Experiment using Synthetic Data

In the experiment using synthetic data, both TL-algorithm and Lowe's algorithm iterates until one of the following criteria is met:-

1. Each element in the error vector is less than 10^{-3} units.

2. The number of iterations exceeds an upper bound of 300.

In the experiment, a series of images is generated and input to the system. In this experiment, 600 images are created. A new model will be used and created randomly in every 20 images. A model consists of four 3D feature points and they are generated randomly from an uniform distribution within a cube of size $20 \times 20 \times 20$ units with the center at $[0, 0, 120]$. Also, the images between the $20n - th$ image and the $20(n + 1) - th$ image will be generated by applying a transformation matrix (generated randomly) to the object in the previous image.

Except for the first image, the system chooses the result of the previous image as the initial guess. It is reasonable because the translation and rotation between two consecutive frames in a realistic movie should be small. Also, there are upper bounds to govern the variations of the rotation and the translation. The experimental result is shown in Table (1)-(6).

By comparing the result shown in Table (2) with that in Table (4), it is shown that the TL-algorithm has a better performance than Lowe's algorithm since the mean value of the iteration required in the TL-algorithm is much less than that in the Lowe's algorithm. The reason for this phenomenon is that, the distance between the model points and the center of perspective remains unchanged between two consecutive images or frames if the motion between them is pure rotation. This

algorithm	min. iter.	max. iter.	Mean (iter.)	S.D. (iter.)	Time (s)
Lowe's	3	11	3.700	0.735	32
TL	2	10	3.252	0.928	13

***Note:- the term "iter." indicates the number of iterations for the systems to converge to a stable answer. "s" stands for second.

Table 1: Result for synthetic data:- bound of rotational angle = 9^o and bound of translation in each axis = 10 units

algorithm	min. iter.	max. iter.	Mean (iter.)	S.D. (iter.)	Time (s)
Lowe's	3	22	4.383	1.120	36
TL	2	42	3.220	1.787	13

Table 2: Result for synthetic data:- bound of rotational angle = 18^o and bound of translation in each axis = 10 units

effect can also be found by comparing the result shown in Table (1) and Table (3).

Also, by Table (2), (4) and (6), it is shown that, the performance of the TL-algorithm declines as the bound of the translation (in each axis) increases. It is because the motion of the object becomes more "translational" as such a bound increases. Therefore, the translational motion becomes dominant. Relatively, the importance of the rotational motion becomes smaller. In the extreme case, the result in Table (5) shows that the performance of the TL-algorithm is much worse than Lowe's algorithm (in terms of the mean values of the iterations) when the motion is pure translational.

As a result, we can conclude that the performance of the TL-algorithm is better in the cases that the transformation of the object is dominated by the rotational motion. Therefore, TL-algorithm is suitable for the environment where the motion of the rigid object (which position is required to find) is mainly rotation.

algorithm	min. iter.	max. iter.	Mean (iter.)	S.D. (iter.)	Time (s)
Lowe's	3	6	3.780	0.579	31
TL	1	9	1.8167	0.929	11

Table 3: Result for synthetic data:- bound of rotational angle = 9^o and bound of translation in each axis = 0 unit

algorithm	min. iter.	max. iter.	Mean (iter.)	S.D. (iter.)	Time (s)
Lowe's	3	51	4.518	2.546	37
TL	1	11	1.747	0.996	11

Table 4: Result for synthetic data:- bound of rotational angle = 18^o and bound of translation in each axis = 0 unit

algorithm	min. iter.	max. iter.	Mean (iter.)	S.D. (iter.)	Time (s)
Lowe's	2	7	2.075	0.326	19
TL	3	84	4.025	4.939	15

Table 5: Result for synthetic data:- bound of rotational angle = 0^o and bound of translation in each axis = 40 units

algorithm	min. iter.	max. iter.	Mean (iter.)	S.D. (iter.)	Time (s)
Lowe's	3	13	4.310	0.8469	35
TL	3	120	4.172	5.359	15

Table 6: Result for synthetic data:- bound of rotational angle = 18^o and bound of translation in each axis = 40 units

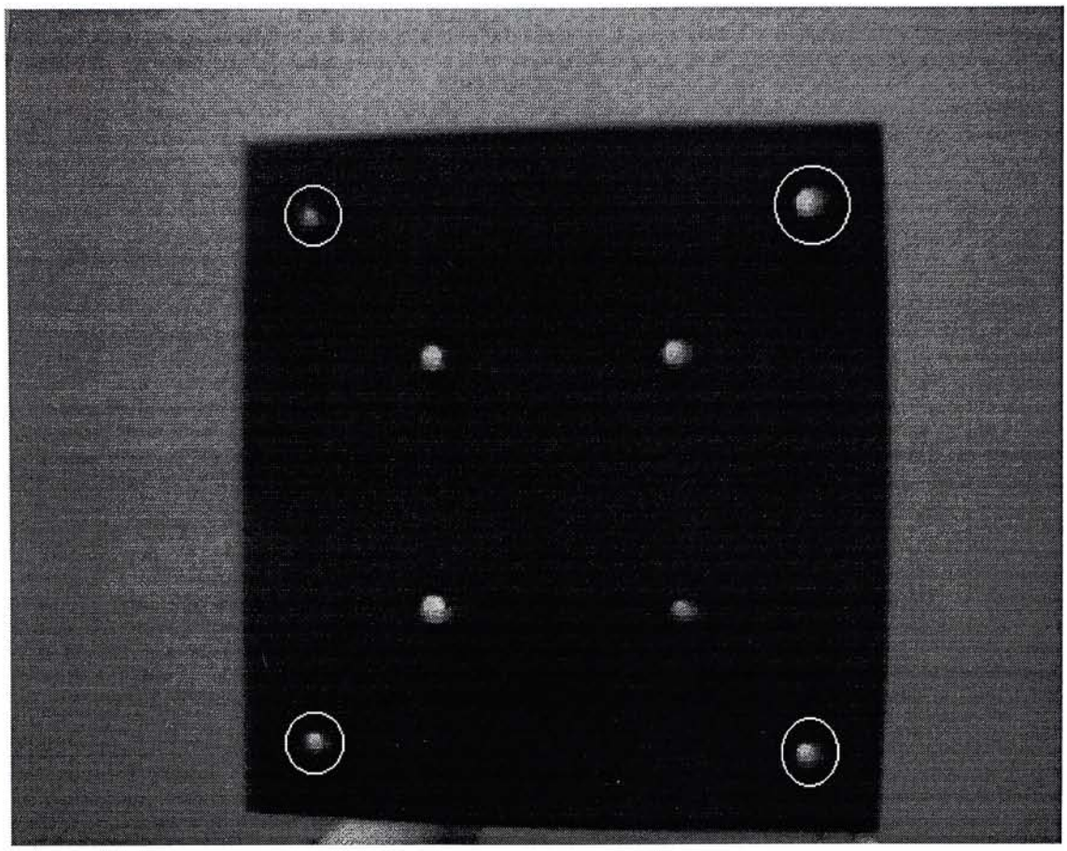


Figure 7: The input device. The four LEDs inside circles are used as the four feature points.

On the other hand, the computation time for the TL-algorithm is much less than that for Lowe's algorithm. It is because the number of the parameters (or unknown to be found) in Lowe's algorithm is 6 which is more than that required by the TL-algorithm. Both these two algorithms use the Gauss-Newton method [36]. In the Gauss-Newton method, there are some calculations of multiplication of the matrices and the inverse of a matrix. Both these calculations are very expensive. Therefore, the smaller the size of the matrices (i.e. the error vector and especially for the Jacobian matrix), the shorter the computation time. It is equivalent to the minimization of the number of the parameters (or unknowns) used in the system. As a result, the computational time for the TL-algorithm is shorter.

3.1.2.2 Experiment using Real Data

In the experiment using real data, images of an input device is taken by a calibrated camera [33]. This device is made of a black card and eight LEDs as shown in Figure (7). The four LEDs at the outermost layer (i.e. the LEDs marked by the circles in Figure (7)) are used as the four model points. A short movie is created by capturing the random motion of this device. Then the correspondences between the four model points (LEDs) and the image points in each frame of the movie are found by using the cross-correlation technique. This movie contains 113 frames. By using the TL-algorithm, about 28 frames can be handled in each second. In this experiment, the limit for the stopping criterion is set to be 10^{-10} units (i.e. each element in the correction vector $\boldsymbol{\kappa}$ should be less than 10^{-10}units when the stopping criterion is reached). Therefore, the TL-algorithm is very fast.

3.2 An Enhancement of Faugeras's Algorithm

The aim of proposing this enhancement is to improve the accuracy of the translational part of the motion of the object calculated by using Faugeras's algorithm [12].

In the original Faugeras's algorithm [12], the rotational and translational components are calculated as shown in Equation (2.27)-(2.29). The orthogonality of the rotation matrix is ensured. However, the cost function in Equation (2.26) cannot be sure to be minimized with the resultant translation matrix and rotation matrix especially when the system is seriously corrupted by noise.

According to the argument by Faugeras, the resultant rotation matrix is the optimal solution. In this thesis, we suggested to use another method to find a suitable translation matrix to minimize the cost function in Equation (2.26).

In order to minimize the cost function in Equation (2.26) and at the same

time keeping the resultant rotational components given from Equation (2.27)-(2.29) to be the solution, the resultant rotational components are back-substituted into Equation (2.26). Then the translational components are computed as follows:

$$\mathbf{t} = -(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{D}\mathbf{q}_R, \tag{3.48}$$

where $\mathbf{C}$ is a $N \times 3$ matrix (which contains the elements in the $10-th$ to $12-th$ columns in $\boldsymbol{\chi}$), $\mathbf{D}$ is a $N \times 9$ matrix (which contains the elements in the $1-st$ to $9-th$ columns in $\boldsymbol{\chi}$), and

$$\mathbf{q}_R = \mathbf{B}\mathbf{q}, \tag{3.49}$$

in which $\mathbf{q}$ contains all the motional components as given in Equation(2.25), and

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

Therefore, $\mathbf{q}_R$ is actually a 9×1 vector and containing all the rotational components of the system. Also,

$$\boldsymbol{\chi}\mathbf{q} = \mathbf{C}\mathbf{t} + \mathbf{D}\mathbf{q}_R. \tag{3.50}$$

By using the above equations, the translation matrix, which can minimize the cost function in Equation (2.26), can be obtained. Actually, it is a kind of constrained optimization problem and readers can refer to Appendix B for more details.

along or about	x-axis	y-axis	z-axis
Rotational Angles (in radian)	0.19783057	-1.04168364	0.39935766
Translational components	-427.4820	-26.6806	450.2650

Table 7: The motion parameters used in Experiment One.

3.2.1 Experimental Comparison between the Original Faugeras's Algorithm and the Modified One

In this section, experiments based on the synthetic data are performed to compare the performances of the original and the modified Faugeras's algorithm. There are two experiments.

3.2.1.1 Experiment One: Fixed Motion

In this experiment, the performances of these two algorithms are measured by using different numbers of feature points with a fixed rotation matrix and a fixed translation matrix as shown in Table (7). The number of feature points ranges from 6 to 100. Also, in order to estimate their performances under noisy environment, noise is added to the image points (i.e. the pixel coordinates) of the projection. The noise is Gaussian and independent, and its standard deviation varies between 0 (for no noise) and 3 pixels. For each test, 100 sets of different feature points are generated randomly from an uniform distribution within a cube of size $2 \times 2 \times 2$ with center at $[0, 0, 300]$. Then the root mean square (RMS) error of the translational components will be recorded over 100 trials for the same number of feature points and noise level. The focal length f is set to be 0.6 units. Also, the distance between two adjacent pixels in the image is 0.0025 units.

Since only the calculations of the translation are different in these two algorithms, so only the RMS error of the translational components are considered in this experiment. The RMS error of the translational components of these two

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0093	0.0318	3.5464	3282.7669	519.7047
20	0.0000	0.0019	0.1889	19.0184	935.9055
45	0.0000	0.0010	0.1086	9.3912	391.2966
70	0.0000	0.0006	0.0748	8.5571	339.1891
100	0.0000	0.0006	0.0621	7.5802	360.2608
150	0.0000	0.0004	0.0459	6.7451	328.8169
200	0.0000	0.0004	0.0409	6.1644	326.4095

Table 8: RMS error of the translational component of the original Faugeras's algorithm along the x-direction with different numbers of feature points (between 6 and 200) and noise level (between 0 and 2).

algorithms is shown in tables (8)-(13). It is shown that the *RMS error of the translational components of the modified Faugeras's algorithm is smaller than that of the original one especially when the noise level is high. Also, the error in the modified version becomes smaller when number of feature points is increasing.*

On the other hand, the computation time of the original version and that of the modified version are shown in Table (14) and (15). From these two tables, it is shown that the *computation time is only depending on the number of feature points used, but not depending on the noise level.* Therefore, the comparison between these two algorithms in terms of computation time can be easily observed from Figure (8). From this figure, it is shown that *the computation time of the original version is shorter than that of the modified version.* It is the result of longer computation time required to compute the translation matrix in the modified version.

As a result, there is a trade-off between computation speed and accuracy in the result.

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0006	0.0094	0.4566	161.7890	462.4204
20	0.0000	0.0002	0.0176	1.1720	186.7725
45	0.0000	0.0000	0.0056	0.4055	166.3094
70	0.0000	0.0000	0.0034	0.3569	38.2557
100	0.0000	0.0000	0.0022	0.2322	75.6689
150	0.0000	0.0000	0.0017	0.1736	9.7463
200	0.0000	0.0000	0.0015	0.1621	10.9373

Table 9: RMS error of the translational component of the modified Faugeras's algorithm along the x-direction with different numbers of feature points (between 6 and 200) and noise level (between 0 and 2).

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0036	0.0142	1.1398	404.5382	145.1875
20	0.0000	0.0009	0.0961	8.3243	53.9644
45	0.0000	0.0005	0.0510	4.1238	44.1712
70	0.0000	0.0003	0.0350	3.3586	39.9494
100	0.0000	0.0003	0.0293	3.0023	31.8459
150	0.0000	0.0002	0.0219	2.2463	29.4750
200	0.0000	0.0002	0.0187	2.1168	31.5101

Table 10: RMS error of the translational component of the original Faugeras's algorithm along the y-direction with different numbers of feature points (between 6 and 200) and noise level (between 0 and 2).

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0000	0.0008	0.0307	11.2862	29.8981
20	0.0000	0.0000	0.0022	0.1642	11.3544
45	0.0000	0.0000	0.0006	0.0544	10.7242
70	0.0000	0.0000	0.0004	0.0423	2.3837
100	0.0000	0.0000	0.0003	0.0345	4.7293
150	0.0000	0.0000	0.0002	0.0222	0.6523
200	0.0000	0.0000	0.0002	0.0194	0.7478

Table 11: RMS error of the translational component of the modified Faugeras's algorithm along the y-direction with different numbers of feature points (between 6 and 200) and noise level (between 0 and 2).

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0062	0.0195	1.2024	219.4376	437.2376
20	0.0000	0.0010	0.0862	10.0583	372.9977
45	0.0000	0.0005	0.0523	6.2899	347.5923
70	0.0000	0.0003	0.0390	5.3925	329.9595
100	0.0000	0.0004	0.0284	5.6945	328.7777
150	0.0000	0.0003	0.0292	4.6892	322.6897
200	0.0000	0.0002	0.0219	4.4855	322.1139

Table 12: RMS error of the translational component of the original Faugeras's algorithm along the z-direction with different numbers of feature points (between 6 and 200) and noise level (between 0 and 2).

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0009	0.0099	0.5088	170.9325	485.3186
20	0.0000	0.0002	0.0176	1.1680	196.4831
45	0.0000	0.0001	0.0057	0.4329	175.0443
70	0.0000	0.0000	0.0036	0.3770	40.7451
100	0.0000	0.0000	0.0023	0.2443	79.7322
150	0.0000	0.0000	0.0018	0.1846	10.4328
200	0.0000	0.0000	0.0015	0.1648	11.6332

Table 13: RMS error of the translational component of the modified Faugeras's algorithm along the z-direction with different numbers of feature points (between 6 and 200) and noise level (between 0 and 2).

noise level	0	0.005	0.05	0.5	2
no. of pt.=6	0.0080	0.0079	0.0079	0.0079	0.0080
no. of pt.=20	0.0093	0.0093	0.0093	0.0093	0.0093
no. of pt.=45	0.0106	0.0106	0.0107	0.0106	0.0106
no. of pt.=70	0.0111	0.0112	0.0112	0.0118	0.0112
no. of pt.=100	0.0121	0.0120	0.0120	0.0124	0.0120
no. of pt.=150	0.0138	0.0136	0.0137	0.0137	0.0137
no. of pt.=200	0.0151	0.0150	0.0152	0.0150	0.0150

Table 14: In Experiment One: The computation time (in terms of second) of the original Faugeras's algorithm with different number of feature points (between 6 and 200) and noise level (between 0 and 2).

noise level	0	0.005	0.05	0.5	2
no. of pt.=6	0.0089	0.0089	0.0088	0.0088	0.0088
no. of pt.=20	0.0103	0.0103	0.0103	0.0103	0.0103
no. of pt.=45	0.0121	0.0121	0.0121	0.0121	0.0121
no. of pt.=70	0.0128	0.0128	0.0128	0.0135	0.0128
no. of pt.=100	0.0139	0.0139	0.0138	0.0142	0.0138
no. of pt.=150	0.0160	0.0158	0.0159	0.0158	0.0158
no. of pt.=200	0.0175	0.0175	0.0179	0.0175	0.0175

Table 15: In Experiment One: The computation time (in terms of second) of modified Faugeras's algorithm with different number of feature points (between 6 and 200) and noise level (between 0 and 2).

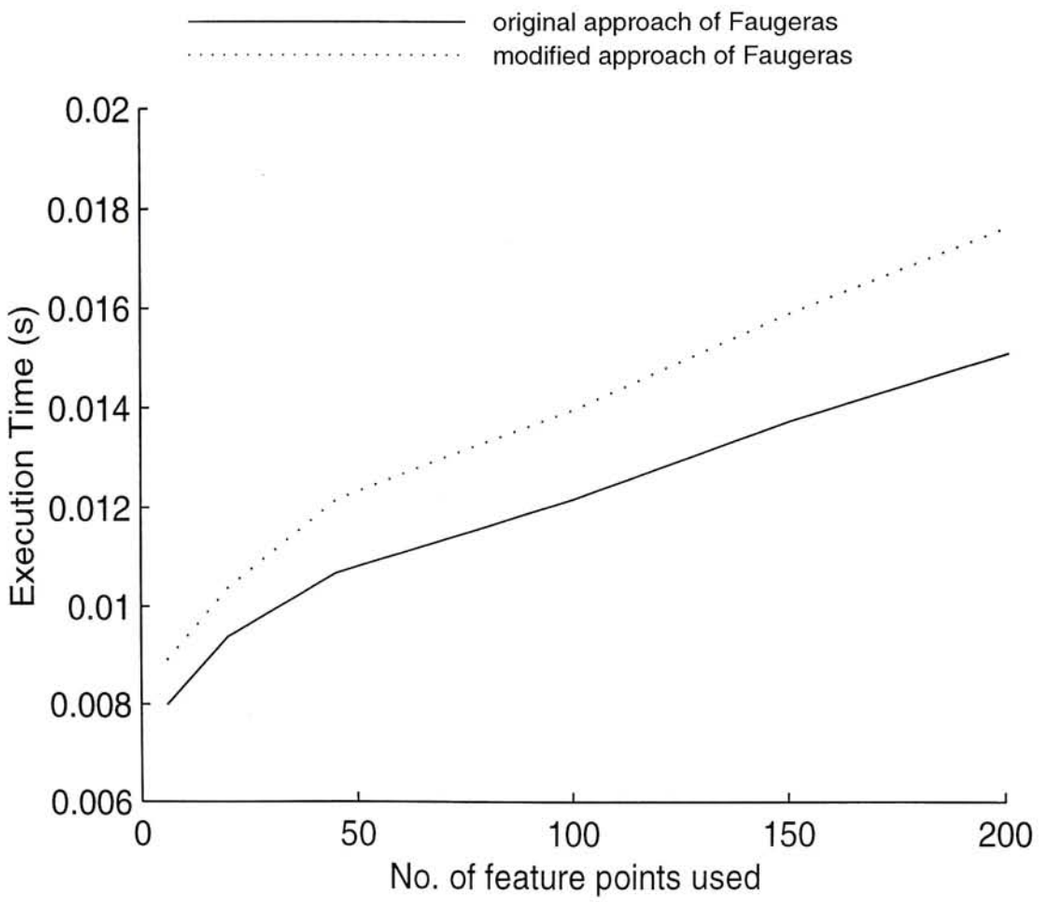


Figure 8: Computation times of the original Faugeras's algorithm and the modified Faugeras's algorithm with fixed motion

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0	0.4	2.5	793.4	8933.8
20	0.0	0.0	0.3	115.7	795.4
45	0.0	0.0	4.3	359.1	3461.0
70	0.0	0.0	2.1	17.6	2061.6
100	0.0	0.0	1.0	136.8	1674.9
150	0.0	0.0	0.2	95.5	136.1
200	0.0	0.0	0.6	71.6	2296.6

Table 16: RMS of percentage error of the translational component of the original Faugeras's algorithm along the x-direction with different numbers of feature points (between 6 and 200) and noise level (between 0 and 2).

3.2.1.2 Experiment Two: Using Motion Generated Randomly

This experiment is similar to the first experiment, however, the motions used are randomly generated each time. The rotation is generated randomly from an uniform distribution such that each of the three rotation angles about the three axes (x, y and z axes) are varying between -180^o and $+180^o$, while the translation vector is generated randomly from an uniform distribution in the range $[-2, ..., 2]$. The focal length f is set to be 0.6 units. Also, the distance between two adjacent pixels in the image is 0.0025 units.

In this experiment, the root mean square (RMS) of the percentage error of the translational components will be recorded over 100 trials for the same number of feature points and noise level.

The RMS percentage error of the translational components are shown in tables (16)-(21). Also, the speeds of these two approaches are shown in Figure (9). From the result, the same phenomena are obtained as mentioned in Experiment

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0	0.0	0.3	95.6	101.6
20	0.0	0.0	0.1	4.4	171.3
45	0.0	0.0	0.3	18.2	281.7
70	0.0	0.0	0.0	1.7	53.1
100	0.0	0.0	0.1	2.7	77.1
150	0.0	0.0	0.0	1.3	54.5
200	0.0	0.0	0.0	2.8	62.3

Table 17: RMS of percentage error of the translational component of the modified Faugeras's algorithm along the x-direction with different numbers of feature points (between 6 and 200) and noise level (between 0 and 2).

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0	0.0	8.1	400.4	905.7
20	0.0	0.0	0.4	99.6	424.5
45	0.0	0.0	0.2	61.1	1213.6
70	0.0	0.0	0.3	30.5	75.8
100	0.0	0.0	0.1	54.5	238.7
150	0.0	0.0	0.1	6.4	48.2
200	0.0	0.0	0.1	15.8	312.2

Table 18: RMS of percentage error of the translational component of the original Faugeras's algorithm along the y-direction with different numbers of feature points (between 6 and 200) and noise level (between 0 and 2).

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0	0.0	1.1	25.1	467.4
20	0.0	0.0	0.0	4.3	38.2
45	0.0	0.0	0.0	6.2	88.0
70	0.0	0.0	0.0	1.7	24.0
100	0.0	0.0	0.0	3.4	48.9
150	0.0	0.0	0.0	0.9	5.5
200	0.0	0.0	0.0	1.2	8.6

Table 19: RMS of percentage error of the translational component of the modified Faugeras's algorithm along the y-direction with different numbers of feature points (between 6 and 200) and noise level (between 0 and 2).

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0	0.0	0.1	25.5	74.4
20	0.0	0.0	0.0	0.5	22.7
45	0.0	0.0	0.0	0.3	18.3
70	0.0	0.0	0.0	0.2	17.1
100	0.0	0.0	0.0	0.2	16.2
150	0.0	0.0	0.0	0.2	16.2
200	0.0	0.0	0.0	0.1	16.3

Table 20: RMS of percentage error of the translational component of the original Faugeras's algorithm along the z-direction with different numbers of feature points (between 6 and 200) and noise level (between 0 and 2).

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0	0.0	0.1	23.3	77.1
20	0.0	0.0	0.0	0.1	12.7
45	0.0	0.0	0.0	0.0	2.0
70	0.0	0.0	0.0	0.0	1.2
100	0.0	0.0	0.0	0.0	1.0
150	0.0	0.0	0.0	0.0	1.0
200	0.0	0.0	0.0	0.0	0.9

Table 21: RMS of percentage error of the translational component of the modified Faugeras's algorithm along the z-direction with different numbers of feature points (between 6 and 200) and noise level (between 0 and 2).

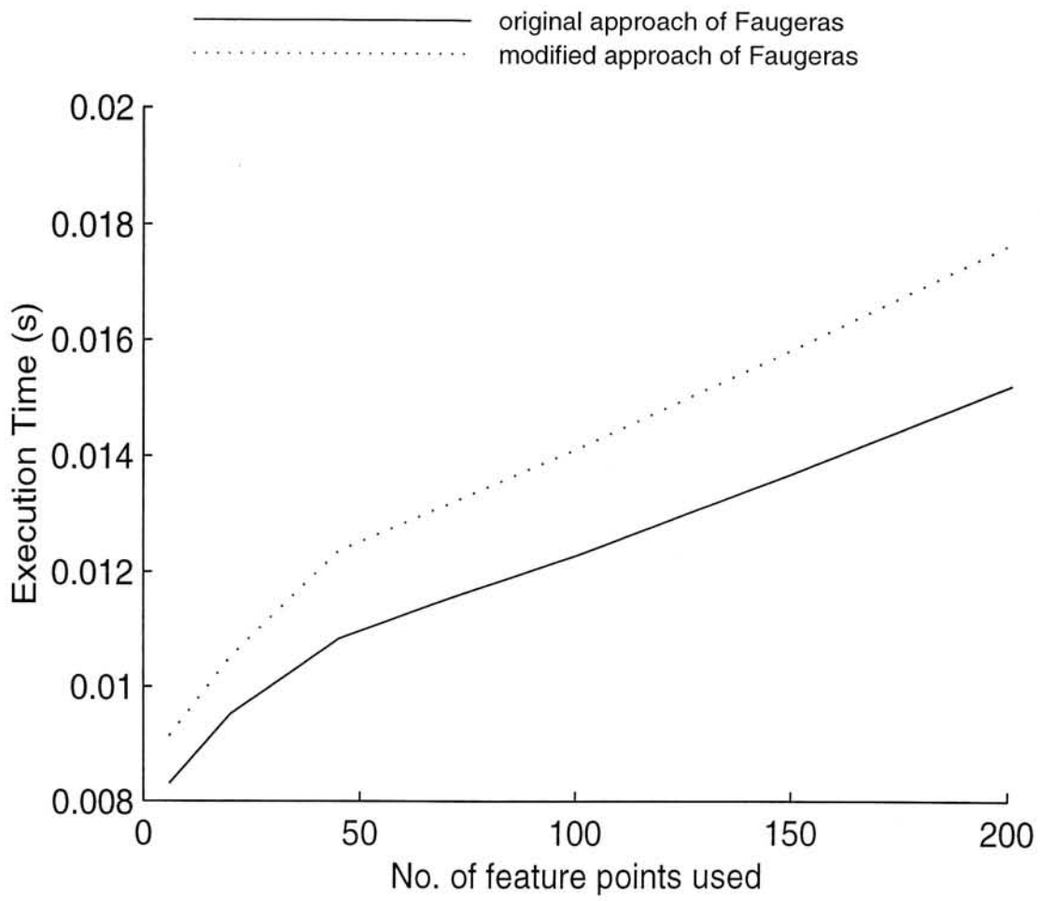


Figure 9: Computation time of the original Faugeras's algorithm and the modified Faugeras's algorithm with random motion

One using fixed motion parameters.

3.2.2 Discussion

From the experiments, we can draw the following conclusions about the modified version of Faugeras's algorithm:

1. The accuracy of the result about the translational components of the modified Faugeras's algorithm is improved especially when the noise level is high;

2. The accuracy in the modified version becomes higher when number of feature points is increasing.

3. The computation time is only depending on number of feature points used, but not depending on noise level. However, the computation speed of the modified version is slower than that of the original version.

Actually, the enhancement in the accuracy is the result of minimizing the cost function in Equation (2.26) by back-substituting the resultant rotational parameters. In the original version, there is no guarantee that the cost function in Equation (2.26) can be minimized by the estimated motion parameters.

However, more computation time is required to back-substitute the resultant rotational parameters into the system to obtain the translation. Therefore, the computation speed of the modified version is lower than that of the original version. As a result, there is a trade-off between computation speed and accuracy.

3.3 A new Linear Algorithm for the Model-based Motion Analysis: Six-point Algorithm

In this paper, a direct and linear approach is proposed to solve the model-based pose estimation problem under full perspective projection. Unlike iterative methods, initial guesses are not required in this approach, so the computation speed of the system is better than that of other iterative algorithms (e.g. [10]). Also, the technique of the singular value decomposition (SVD) [16] is used to maintain the orthogonality of the rotational matrix.

Actually, this algorithm is similar to Faugeras's algorithm [12] that a similar system of $2N$ homogeneous linear equations (as shown in equations (2.22)-(2.25)) for N feature points by using equations (1.3) and (1.4). However, different constraints are applied in the constrained optimization step in order to see whether there is any improvement. In this thesis, two versions of the Six-point Algorithm are proposed and discussed. These two versions also differ by using different constraints in the step of constrained optimization of the linear system.

The experiments using both synthetic and real data have been done to verify this approach.

3.3.1 General Information of the Six-point Algorithm

Both versions of the Six-point algorithm also contain two parts: 1) Linear solution, 2) constraint satisfactory part.

The first part is used to obtain a rough estimate of the motion of the object. The second part is used to refine the solution in order to maintain the orthogonal property of the rotation matrix so that the rigidity of the object can be preserved.

The two versions will use the same system of homogeneous linear equation

in 3.51 and they are only different in the Linear solution part. The constraint satisfactory part is the same in both versions.

In the Six-point algorithm, the intrinsic parameters [12] of the camera can also be obtained by using equations (2.27)-(2.29) just the same as Faugeras's algorithm. However, intrinsic parameters [12] and camera calibration [70][71] are out of the scope of this thesis. So, the details of calculating the intrinsic parameters will not be described in this thesis. Since, the aim of this chapter is motion analysis with 2D-to-3D point correspondences, the camera used to capture the image is well-calibrated before doing the experiments. Actually, an experiment has been done to investigate the ability of the two versions of Six-point algorithm in finding the intrinsic parameters. From the experimental result, the performances of all these three approaches (two versions of the Six-point algorithm and original Faugeras's algorithm) in estimating the intrinsic parameters are comparative. However, the details of this experiment will not be mentioned in this thesis.

3.3.2 Original Version of the Six-point Algorithm

3.3.2.1 Linear Solution Part

In the original version of Six-point algorithm, the system of homogeneous linear equation is obtained by rearranging Equation (1.3) and (1.4) as the following:

$$\mathbf{A}\mathbf{h} = \mathbf{0}, \tag{3.51}$$

where $\mathbf{h} = [r_{11}, r_{12}, r_{13}, r_{21}, r_{22}, r_{23}, r_{31}, r_{32}, r_{33}, t_x, t_y, t_z]^T$, and

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1, & \mathbf{b}_1, & \mathbf{a}_2, & \mathbf{b}_2, & \cdots & \mathbf{a}_N, & \mathbf{b}_N \end{bmatrix}^T, \tag{3.52}$$

in which

$\mathbf{a}_i = [fX_i^k,\ fY_i^k,\ fZ_i^k,\ 0,\ 0,\ 0,\ -x_i^{k+1}X_i^k,\ -x_i^{k+1}Y_i^k,\ -x_i^{k+1}Z_i^k,\ f,\ 0,\ -x_i^{k+1}]$,

also,

$\mathbf{b}_i = [0,\ 0,\ 0,\ fX_i^k,\ fY_i^k,\ fZ_i^k,\ -y_i^{k+1}X_i^k,\ -y_i^{k+1}Y_i^k,\ -y_i^{k+1}Z_i^k,\ 0,\ f,\ -y_i^{k+1}]$,

and N is the number of model points on the rigid object.

Finding the answer of the motion parameters $\mathbf{h}$ is equivalent to finding a $\mathbf{h}$ to minimize the following equation:

$$\min_{\mathbf{h}} \|\mathbf{Ah}\|^2 . \tag{3.53}$$

Supposing that

$$\begin{aligned} \mathbf{Ah} &= \mathbf{E} \\ \Rightarrow \quad \|\mathbf{Ah}\|_F^2 &= \|\mathbf{E}\|_F^2 \end{aligned} \tag{3.54}$$

where $\mathbf{E}$ is a 12×1 error vector, and $\|\circ\|_F$ stands for the Frobenius norm [16] and this notation will be used throughout this section.

To minimize the error is equivalent to minimizing $\|\mathbf{E}\|_F^2$ in the Equation (3.54). Supposing that

$$\mathbf{h} = s\mathbf{h}_a, \tag{3.55}$$

where $\|\mathbf{h}_a\| = 1$, s is scalar and $s = \|\mathbf{h}\|$. Then letting $\mathbf{E} = s\mathbf{e}$, the equation (3.54) becomes:

$$\begin{aligned} \mathbf{A}(s\mathbf{h}_a) &= s\mathbf{e} \\ \mathbf{Ah}_a &= \mathbf{e} \\ \Rightarrow \quad \|\mathbf{Ah}_a\|_F^2 &= \|\mathbf{e}\|_F^2 \end{aligned} \tag{3.56}$$

According to equations (3.54) and (3.56), the following conclusion can be drawn:

> if $\mathbf{h}_a$ gives a minimum of $\|\mathbf{e}\|_F^2$, then $s\mathbf{h}_a$ gives a minimum of $\|s\mathbf{e}\|_F^2$ for all scalar s.

Also, Faugeras showed that the rank of $\mathbf{A}$ is equal to 11 in general [12]. Then the nullspace of the system is usually dimension 1 and there is usually an unique solution to Equation (3.53).

Therefore, the minimization problem in Equation (3.53) becomes the following problem:

$$\underset{\mathbf{h}_a}{\text{Min}} \|\mathbf{A}\mathbf{h}_a\|_F^2 \quad \text{subject to } \|\mathbf{h}_a\|_F^2 = 1. \tag{3.57}$$

By applying the technique of Lagrange multiplier, the whole problem can be further transformed into the following:

$$\underset{\mathbf{h}_a}{MinF}(\mathbf{h}_a\,,\,\lambda), \tag{3.58}$$

where

$$F(\mathbf{h}_a, \lambda) = \|\mathbf{A}\mathbf{h}_a\|_F^2 + \lambda(1 - \|\mathbf{h}_a\|_F^2). \tag{3.59}$$

In order to minimize the cost function in Equation (3.59), the first derivative of Equation (3.59) with respect to $\mathbf{h}_a$ is assigned to zero, then we have

$$\mathbf{A}^T\mathbf{A}\mathbf{h}_a = \lambda\mathbf{h}_a. \tag{3.60}$$

Therefore, $\mathbf{h}_a$ is an eigenvector of $\mathbf{A}^T\mathbf{A}$ and λ is the corresponding eigenvalue, and λ should be the smallest eigenvalue in order to minimize the cost function in equation (3.59).

3.3.2.2 Constraint Satisfaction

However, the motion parameters found in the "Linear Solution" part is only the rough approximation of the solution of the motion parameters because the rotation matrix represented in $\mathbf{h}_a$ may not be orthogonal. Therefore one more step should be applied to find out a correct answer (or the best approximation in the noisy environment).

Supposing that $\mathbf{h}_e$ is the correct solution. Let $\mathbf{h}_u$ be the normalized vector of $\mathbf{h}_e$. Then we have:

$$\mathbf{h}_e = \|\mathbf{h}_e\|\,\mathbf{h}_u.$$

Then for $s = \|\mathbf{h}_e\|$, the error function of the whole system will become $\|\mathbf{h}_e\|\,\mathbf{e}$ (i.e. $\mathbf{E}$). However, for this error function, the minimum error of Equation (3.53) should be the error when $\mathbf{h}$ equals to $\|\mathbf{h}_e\|\,\mathbf{h}_a$.

On the other hand, if the noise in the system is small, then $\|\mathbf{h}_e\|\,\mathbf{h}_a$ is near to $\mathbf{h}_e$ (for the system is free of noise, $\|\mathbf{h}_e\|\,\mathbf{h}_a$ should be exactly equal to $\mathbf{h}_e$, i.e., $\mathbf{h}_u = \mathbf{h}_a$. It is because the rank of $\mathbf{A}$ is usually equal to 11). Therefore it is confident that:

If we have :

1. $\mathbf{h}_o$ is a parameter vector that can preserve the constraint about the orthogonality of the rotation;
2. $\|\mathbf{h}_o\| = \|\mathbf{h}_e\|$, and
3. for all parameter vectors that obey the constraint and have length equal to $\|\mathbf{h}_e\|$, $\mathbf{h}_o$ is the nearest parameter vector to $\|\mathbf{h}_e\|\,\mathbf{h}_a$ (i.e. $\|(\mathbf{h}_o - \|\mathbf{h}_e\|\,\mathbf{h}_a)\|$ is minimized among all the other vectors that obey the constraint).

then it is probably that $\mathbf{h}_o$ is equal to (or the best approximation of) the correct answer $\mathbf{h}_e$.

However, we do not know any information about z (where $z = \|\mathbf{h}_e\|$) after the calculation in the "Linear Solution" part.

Actually, we need not find out the value of z explicitly, but we can still go ahead to find out the rotation and translation of the system. Therefore, the above constraint problem can be converted to the following minimization problem:

$$\mathbf{R}_o = \underset{\mathbf{R}}{\mathrm{Min}} \|\mathbf{R} - z\mathbf{R}_e\|_F^2 \tag{3.61}$$

for all orthogonal matrices $\mathbf{R}$ and

$$\mathbf{R}_e = \sigma \mathbf{R}_h, \tag{3.62}$$

where

$$\begin{cases} \sigma = 1 & \text{if } \det(\mathbf{R}_h) \geq 0 \\ \sigma = -1 & \text{otherwise} \end{cases},$$

in which

$$\mathbf{R}_h = \begin{bmatrix} h_{a1} & h_{a2} & h_{a3} \\ h_{a4} & h_{a5} & h_{a6} \\ h_{a7} & h_{a8} & h_{a9} \end{bmatrix}. \tag{3.63}$$

Note: We can assign $(-\mathbf{R}_h)$ to be the estimated rotation matrix if $\det(\mathbf{R}_h) < 0$, because $(-\mathbf{h}_a)$ can also minimize the cost function in (3.59).

Actually, $\mathbf{R}_o$ is also the answer of Equation (3.61) for $z = 1, 2$ or any real numbers (i.e. for all real number z, the answer of Equation (3.61) is also $\mathbf{R}_o$ if there is no change for $\mathbf{R}_e$). As a result, $\mathbf{R}_o$ is only depending on $\mathbf{R}_e$, but not the scalar z. We can prove it simply by making use the technique of SVD.

Supposing that we want to find an orthogonal matrix $\mathbf{B}$ to minimize the following cost function:

$$\|\mathbf{B} - \mathbf{C}\|_F^2 .$$

where $\mathbf{C}$ is any square matrix given.

It is actually the Orthogonal Procrustes problem [16]. Golub and Van Loan gave out an answer by using the technique of SVD in their book [16]. According to their answer,

$$\mathbf{B} = \mathbf{U}\mathbf{V}^T,$$

where $\mathbf{U}\mathbf{\Lambda}\mathbf{V}^T = \mathbf{C}$, in which $\mathbf{\Lambda}$ is the singular value matrix and $\mathbf{U}$ and $\mathbf{V}$ are the orthogonal matrices.

Then, supposing that we are going to find out the orthogonal matrix $\mathbf{B}_o$ to minimize the cost function $\|\mathbf{B}_o - \mathbf{D}\|_F^2$, in which $\mathbf{D} = k\mathbf{C}$ for any real number k. According to Golub's statement [16], we should first apply SVD to $\mathbf{D}$ to get the two orthogonal matrices. However, the scalar k will not affect the calculation of SVD. The singular value matrix of $\mathbf{D}$ is equal to the k times of the singular value matrix of $\mathbf{C}$. Therefore we can have the following result,

$$\mathbf{D} = \mathbf{U}\mathbf{\Gamma}\mathbf{V}^T,$$

where $\mathbf{\Gamma} = k\mathbf{\Lambda}$. Then $\mathbf{B}_o = \mathbf{U}\mathbf{V}^T = \mathbf{B}$.

As a result, it is shown that the answer of an Orthogonal Procrustes problem is independent of the scalar factor of the given matrix (i.e. $\mathbf{C}$ or $\mathbf{D}$).

Therefore, we can conclude that if:

$$\mathbf{R}_o = \underset{\mathbf{R}}{\text{Min}} \|\mathbf{R} - \mathbf{R}_e\|_F^2, \tag{3.64}$$

then $\mathbf{R}_o$ is also an answer of the following

$$\underset{\mathbf{R}}{\text{Min}} \|\mathbf{R} - z\mathbf{R}_e\|_F^2,$$

for all scalar z. So, it is no need to find out the value of z explicitly.

On the other hand, there is also another important meaning about Equation (3.61). It actually finds the first nine elements in the parameter vector $\mathbf{h}_{o2}$ such that,

1. $\|(\mathbf{h}_{o2}^* - \|\mathbf{h}_e\|\,\mathbf{h}_a^*)\|_F^2$ is minimum, where $\mathbf{h}_a^*$ only contains the first nine elements of $\mathbf{h}_a$ and $\mathbf{h}_{o2}^*$ only contains the first nine elements of $\mathbf{h}_{o2}$;

2. $\mathbf{h}_{o2}$ obeys the orthogonality of the rotation matrix.

According to the above argument, it can be confident that the first nine elements in $\mathbf{h}_{o2}$ should be the best (or very very good) approximation of the first nine elements

of the correct answer $\mathbf{h}_e$ (or they are exactly the correct answer if there is no noise in the system).

As a result, the orthogonal matrix $\mathbf{R}_o$ obtained by minimising the cost function given in Equation (3.64) is the best approximation (or even the correct solution for the noise free environment) of the rotation parameter.

In order to maintain the orthogonality and to minimize the cost function in Equation (3.64), two methods are discussed in this thesis. The first one makes use of the representation of rotations by quaternions [18][72][43]. Another is based on the technique of SVD.

Use of Representation of Rotations by Quaternion

Actually the detailed description of representation of rotation by quaternions is presented in Appendix A.

In order to apply the technique described in Appendix A to find out the answer of the rotation matrix, the matrices $\mathbf{C}$ and $\mathbf{D}$ in Equation (1.101) should be replaced by $\mathbf{I}$ and $\mathbf{R}_e$ respectively (where $\mathbf{I}$ is a 3×3 identity matrix).

Then the quaternion $\mathbf{q}$ (i.e. the eigenvector of $\mathbf{B}$ in Equation (1.102)) is the required quaternion representation of the rotation parameters. By applying Equation (1.105), the rotation matrix can be obtained.

Use of Singular Value Decomposition

In this thesis, another technique is applied to determine the rotation matrix based on the technique of singular value decomposition (SVD).

Actually, Equation (3.64) is a standard orthogonal Procrustes problem [16]. Therefore the rotation matrix $\mathbf{R}_o$ is given as follows:

$$\mathbf{R}_o = \mathbf{U}\mathbf{V}^T, \tag{3.65}$$

where $\mathbf{U}$ and $\mathbf{V}$ are 3×3 orthogonal matrices which is given by the SVD of $\mathbf{R}_e$ as follows:

$$\mathbf{R}_e = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T. \tag{3.66}$$

Actually, the use of the technique of SVD to determine the motion between two sets of 3D points (i.e. 3D-to-3D correspondence problem) was proposed by Arun, Huang and Blostein [15]. And their algorithm was enhanced by Umeyama [21]. However, our proposed approach to determine the rotation matrix is different from the algorithm using SVD in 3D-to-3D correspondence problem in the way that our proposed algorithm will not give out a rotation matrix which is a reflection (i.e. $\det(\mathbf{R}_o)$=-1). Our approach will always give out a rotation matrix with +1 as the determinant. It is because the determinant of $\mathbf{R}_e$ is ensured to be positive in equation (3.62). In addition, according to the properties of SVD, all the singular values must be larger than or equal to zero. $\det(\mathbf{\Lambda})$ must be positive. Therefore, $\det(\mathbf{U}\mathbf{V}^T)$ must also be positive and it is equal to +1.

Determination of the translational matrix

After the rotation matrix $\mathbf{R}_o$ is obtained, we would like to find the translation matrix $\mathbf{t}$. From (3.51), we can derive the following linear system:

$$\mathbf{\Gamma t} = \mathbf{\Theta}, \tag{3.67}$$

where

$$\mathbf{\Gamma} = \begin{bmatrix} \boldsymbol{\alpha}_1, & \boldsymbol{\beta}_1, & \cdots & \boldsymbol{\alpha}_N, & \boldsymbol{\beta}_N \end{bmatrix}^T, \tag{3.68}$$

and

$$\mathbf{\Theta} = \begin{bmatrix} c_1, & d_2, & \cdots & c_N, & d_N \end{bmatrix}^T, \tag{3.69}$$

in which

$$\boldsymbol{\alpha}_i = \left[\begin{array}{ccc} -f, & 0, & x_i^{k+1} \end{array} \right], \tag{3.70}$$

$$\boldsymbol{\beta}_i = \left[\begin{array}{ccc} 0, & -f, & y_i^{k+1} \end{array} \right], \tag{3.71}$$

and

$$\begin{aligned} c_i =& R_{o_{11}} f X_i^k + R_{o_{12}} f Y_i^k + R_{o_{13}} f Z_i^k - R_{o_{31}} x_i^{k+1} X_i^k \\ & - R_{o_{32}} x_i^{k+1} Y_i^k - R_{o_{33}} x_i^{k+1} Z_i^k, \end{aligned} \tag{3.72}$$

$$\begin{aligned} d_i =& R_{o_{21}} f X_i^k + R_{o_{22}} f Y_i^k + R_{o_{23}} f Z_i^k - R_{o_{31}} y_i^{k+1} X_i^k \\ & - R_{o_{32}} y_i^{k+1} Y_i^k - R_{o_{33}} y_i^{k+1} Z_i^k. \end{aligned} \tag{3.73}$$

Also $R_{o_{mn}}$ is the element of $\mathbf{R}_o$ in the $m-th$ row and $n-th$ column.

By using the least-squares fitting technique, the estimated translation matrix $\mathbf{t}_o$ is given as follows:

$$\mathbf{t}_o = \left(\boldsymbol{\Gamma}^T \boldsymbol{\Gamma} \right)^{-1} \boldsymbol{\Gamma}^T \boldsymbol{\Theta}. \tag{3.74}$$

As a result, the rotation $\mathbf{R}_o$ and translation $\mathbf{t}_o$ of the rigid object between time k and $k+1$ are obtained by the procedures described above.

3.3.3 Second Version of the Six-point Algorithm

In the second version of Six-point algorithm, the same homogeneous linear equation system shown in Equation (3.51) is used. The difference of this approach from the original version is to use different constraints in the constrained optimization in the "Linear Solution Part". In the second version, the following constraint is used instead of $\|\mathbf{h}\|_F^2 = 1$ in the original version:

$$\|\mathbf{r}_v\|_F^2 = 3, \tag{3.75}$$

where $\mathbf{r}_v = [r_{11}, r_{12}, r_{13}, r_{21}, r_{22}, r_{23}, r_{31}, r_{32}, r_{33}]^T$.

Actually, the constraint in Equation (3.75) is the fact that $\|\mathbf{R}\|_F^2 = 3$ for any rotation matrix $\mathbf{R}$ (since a rotation matrix is orthogonal). As shown in Appendix B that $\mathbf{Ah}$ can be rewritten into the following form:

$$\mathbf{Ah} = \mathbf{Ct} + \mathbf{Dr}_v, \tag{3.76}$$

where $\mathbf{t}$ is the 3×1 translation matrix of the motion, and the dimensions of $\mathbf{C}$ and $\mathbf{D}$ are respectively $(N \times 3)$ and $(N \times 9)$.

As a result, the minimization problem in Equation (3.53) can be transformed into the following problem by applying the technique of Lagrange multiplier:

$$\underset{\mathbf{r}_v, \mathbf{t}}{\text{Min}} \Psi(\mathbf{r}_v\,,\, \mathbf{t}\,,\, \tau), \tag{3.77}$$

where

$$\Psi(\mathbf{r}_v\,,\, \mathbf{t}\,,\, \tau) = \|\mathbf{Ct} + \mathbf{Dr}_v\|^2 + \tau(3 - \|\mathbf{r}_v\|^2). \tag{3.78}$$

Minimizing the cost function in Equation (3.77) can be achieved by using the results described in Appendix B. Then the result of $\mathbf{r}_v$ is an eigenvector of the matrix $\mathbf{K}$ of norm $\sqrt{3}$ corresponding to the smallest eigenvalue τ, where

$$\mathbf{K} = \mathbf{D}^T(\mathbf{I} - \mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T)\mathbf{D}.$$

Then, the "Constraint Satisfactory Part" described in Section 3.3.2.2 is applied to refine the solution of the rotational parameters in order to preserve the orthogonality of the rotation matrix. Afterwards, the translation matrix can also be obtained as described in Section 3.3.2.2.

3.3.4 Experiment

In this research, we used both synthetic and real data to verify our theory. The algorithm was coded in Matlab 5.0. All the experiments were conducted on

a Sun Ultra 1/170E Creator3D machine (which uses Solaris 2.5.1 as the operating system, and 256MB ram as the primary memory).

3.3.4.1 With Synthetic Data

Two types of experiments have been done to compare the performances of the following algorithms:

1. the original Faugeras's algorithm;
2. the modified Faugeras's algorithm;
3. the original Six-point algorithm;
4. the second version of the Six-point algorithm.

The first experiment will use a fixed motion while the second experiment will use random generated motions.

Experiment One: With Fixed Motion

In this experiment, the performances of these two algorithms are measured by using different numbers of feature points with a fixed rotation matrix and a fixed translation matrix as shown in Table (7). The number of feature points ranges from 6 to 100. Also, in order to estimate their performances under noisy environment, noise is added to the image points (i.e. the pixel coordinates) of the projection. The noise is Gaussian and independent, and its standard deviation varies between 0 (for no noise) and 3 pixels. For each test, 100 sets of different feature points are generated randomly from an uniform distribution within a cube of size $2 \times 2 \times 2$ with center at $[0, 0, 300]$. Then the root mean square (RMS) error of the translational components and rotational angles will be recorded over 100 trials for the same number of feature points and noise level. The focal length f is

set to be 0.6 units. Also, the distance between two adjacent pixels in the image is 0.0025 units. When considering the RMS error of rotation angles, the unit is in radian.

Actually, the result shown in Section 3.2.1.1 is part of the result in this experiment.

From the experiment, it is observed that the accuracies of using the technique of quaternions and SVD in "Constraint Satisfactory Part" of the Six-point algorithm are almost identical. This phenomenon occurs because both the techniques are designed to obtain optimal solution [16][18][72][43] in standard Orthogonal Procrustes problem [16] in the "Constraint Satisfactory Part". Therefore, the rotation matrices obtained by these two techniques are almost identical. The Frobenius norm of the difference of the rotation matrices obtained by these two approaches is about 10^{-15}. And the difference of the translation matrices is also so small that can be neglected. As a result, the comparison of these two techniques in terms of accuracy is omitted. However, the computation speeds of these two techniques are quite different. Therefore, when we are discussing about the accuracies of the two versions of Six-point algorithm, only the results using the technique of SVD will be considered.

As mentioned in Section 3.2, the results of the rotation in the original version and modified version of Faugeras's approach are the same, so only the rotational angles of the original version of Faugeras's algorithm are discussed in this experiment.

Tables (22)-(30) indicate the RMS error of the rotational angles of the two versions of the Six-point algorithm and the original Faugeras's algorithm. Tables (22)-(24) illustrate the result of the rotational angles about the x-axis. Tables (25)-(27) illustrate the result of the rotational angles about the y-axis. Tables

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0000	0.0000	0.0027	0.5227	1.7333
20	0.0000	0.0000	0.0002	0.0194	0.3822
45	0.0000	0.0000	0.0001	0.0092	0.2327
70	0.0000	0.0000	0.0001	0.0074	0.1241
100	0.0000	0.0000	0.0001	0.0062	0.1004
150	0.0000	0.0000	0.0001	0.0046	0.0830
200	0.0000	0.0000	0.0000	0.0043	0.0723

Table 22: RMS error of rotational angle of the original Six-point algorithm about the x-axis

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0000	0.0000	0.0027	0.5740	1.6879
20	0.0000	0.0000	0.0002	0.0193	1.3678
45	0.0000	0.0000	0.0001	0.0093	1.0048
70	0.0000	0.0000	0.0001	0.0074	0.7162
100	0.0000	0.0000	0.0001	0.0062	0.6665
150	0.0000	0.0000	0.0001	0.0047	0.4463
200	0.0000	0.0000	0.0000	0.0043	0.3753

Table 23: RMS error of the rotational angle of the second version of the Six-point algorithm about the x-axis

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0000	0.0000	0.0023	0.6661	1.7129
20	0.0000	0.0000	0.0001	0.0132	0.9614
45	0.0000	0.0000	0.0001	0.0058	0.6477
70	0.0000	0.0000	0.0000	0.0050	0.3105
100	0.0000	0.0000	0.0000	0.0034	0.2797
150	0.0000	0.0000	0.0000	0.0036	0.0690
200	0.0000	0.0000	0.0000	0.0030	0.0715

Table 24: RMS error of the rotational angle of the original Faugeras's algorithm about the x-axis

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0000	0.0000	0.0016	0.2624	0.7452
20	0.0000	0.0000	0.0001	0.0119	0.2293
45	0.0000	0.0000	0.0001	0.0063	0.1228
70	0.0000	0.0000	0.0000	0.0045	0.0869
100	0.0000	0.0000	0.0000	0.0039	0.0701
150	0.0000	0.0000	0.0000	0.0040	0.0552
200	0.0000	0.0000	0.0000	0.0032	0.0511

Table 25: RMS error of the rotational angle of the original Six-point algorithm about the y-axis

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0000	0.0000	0.0016	0.2731	0.8576
20	0.0000	0.0000	0.0001	0.0122	0.6398
45	0.0000	0.0000	0.0001	0.0064	0.3714
70	0.0000	0.0000	0.0000	0.0046	0.2912
100	0.0000	0.0000	0.0000	0.0039	0.2530
150	0.0000	0.0000	0.0000	0.0040	0.2050
200	0.0000	0.0000	0.0000	0.0032	0.2130

Table 26: RMS error of the rotational angle of the second version of the Six-point algorithm about the y-axis

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0000	0.0000	0.0032	0.3999	1.2327
20	0.0000	0.0000	0.0002	0.0180	0.6152
45	0.0000	0.0000	0.0001	0.0082	0.3555
70	0.0000	0.0000	0.0001	0.0067	0.2545
100	0.0000	0.0000	0.0001	0.0047	0.1391
150	0.0000	0.0000	0.0000	0.0049	0.1181
200	0.0000	0.0000	0.0000	0.0042	0.1179

Table 27: RMS error of the rotational angle of the original Faugeras's algorithm about the y-axis

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0000	0.0000	0.0034	0.3891	1.1153
20	0.0000	0.0000	0.0003	0.0260	0.4142
45	0.0000	0.0000	0.0001	0.0120	0.2880
70	0.0000	0.0000	0.0001	0.0098	0.1650
100	0.0000	0.0000	0.0001	0.0085	0.1305
150	0.0000	0.0000	0.0001	0.0062	0.1088
200	0.0000	0.0000	0.0001	0.0058	0.0966

Table 28: RMS error of the rotational angle of the original Six-point algorithm about the z-axis

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0000	0.0000	0.0034	0.4638	1.2891
20	0.0000	0.0000	0.0003	0.0258	1.2652
45	0.0000	0.0000	0.0001	0.0121	1.1059
70	0.0000	0.0000	0.0001	0.0098	0.8209
100	0.0000	0.0000	0.0001	0.0085	0.7597
150	0.0000	0.0000	0.0001	0.0062	0.5026
200	0.0000	0.0000	0.0001	0.0058	0.4228

Table 29: RMS error of the rotational angle of the second version of the Six-point algorithm about the z-axis

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0000	0.0000	0.0039	0.5321	1.1921
20	0.0000	0.0000	0.0002	0.0205	1.0066
45	0.0000	0.0000	0.0001	0.0105	0.4766
70	0.0000	0.0000	0.0001	0.0084	0.3290
100	0.0000	0.0000	0.0001	0.0073	0.2326
150	0.0000	0.0000	0.0001	0.0054	0.1704
200	0.0000	0.0000	0.0000	0.0050	0.1576

Table 30: RMS error of the rotational angle of the original Faugeras's algorithm about the z-axis

(28)-(30) illustrate the result of the rotational angles about the z-axis.

From these 9 tables, it is shown that the accuracies of the result of the rotation angles in these three approaches are comparative. Also, it is shown that the accuracy increases when the number of feature points used is increasing.

Moreover, tables (31)-(36) illustrate the RMS error of translational components of the original version and the second version of the Six-point algorithm. Also, tables (8)-(13) show the RMS error of translational components of the original Faugeras's algorithm and the modified Faugeras's algorithm. From these 12 tables, it is shown that the accuracies of the two versions of Six-point algorithm and the modified Faugeras's algorithm are comparable. However, the RMS error of translational components of the original Faugeras's algorithm is much larger than that of the other three approaches.

On the other hand, the computation time of the following approaches are recorded and compared:

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0000	0.0067	0.2791	142.4146	517.8505
20	0.0000	0.0001	0.0125	0.8372	85.6345
45	0.0000	0.0000	0.0043	0.3314	10.7527
70	0.0000	0.0000	0.0028	0.2929	6.2274
100	0.0000	0.0000	0.0019	0.2206	5.6366
150	0.0000	0.0000	0.0016	0.1524	4.5665
200	0.0000	0.0000	0.0014	0.1359	4.4229

Table 31: RMS error of the translational component of the original Six-point algorithm along the x-direction

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0000	0.0067	0.2788	166.5805	486.7313
20	0.0000	0.0001	0.0125	0.8471	285.8782
45	0.0000	0.0000	0.0042	0.3420	181.1637
70	0.0000	0.0000	0.0028	0.2986	29.0298
100	0.0000	0.0000	0.0019	0.2181	20.2719
150	0.0000	0.0000	0.0016	0.1516	13.1828
200	0.0000	0.0000	0.0014	0.1348	13.0214

Table 32: RMS error of the translational component of the second version of the Six-point algorithm along the x-direction

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0000	0.0005	0.0283	9.2338	32.7777
20	0.0000	0.0000	0.0014	0.1132	5.7759
45	0.0000	0.0000	0.0005	0.0429	0.9138
70	0.0000	0.0000	0.0003	0.0369	0.5680
100	0.0000	0.0000	0.0003	0.0304	0.4843
150	0.0000	0.0000	0.0002	0.0196	0.4155
200	0.0000	0.0000	0.0001	0.0169	0.3799

Table 33: RMS error of the translational component of the original Six-point algorithm along the y-direction

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0000	0.0005	0.0282	10.9421	30.9757
20	0.0000	0.0000	0.0014	0.1108	18.0250
45	0.0000	0.0000	0.0005	0.0433	11.3955
70	0.0000	0.0000	0.0003	0.0376	2.1024
100	0.0000	0.0000	0.0003	0.0303	1.3726
150	0.0000	0.0000	0.0002	0.0196	0.9173
200	0.0000	0.0000	0.0001	0.0168	0.9574

Table 34: RMS error of the translational component of the second version of the Six-point algorithm along the y-direction

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0000	0.0073	0.2981	150.4151	544.6467
20	0.0000	0.0001	0.0129	0.8316	90.3496
45	0.0000	0.0000	0.0044	0.3456	11.2944
70	0.0000	0.0000	0.0030	0.3088	6.3936
100	0.0000	0.0000	0.0020	0.2316	5.9335
150	0.0000	0.0000	0.0017	0.1594	4.7898
200	0.0000	0.0000	0.0014	0.1413	4.6554

Table 35: RMS error of the translational component of the original Six-point algorithm along the z-direction

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0000	0.0073	0.2978	176.3084	512.2487
20	0.0000	0.0001	0.0129	0.8332	299.8390
45	0.0000	0.0000	0.0044	0.3574	190.5932
70	0.0000	0.0000	0.0030	0.3147	30.0544
100	0.0000	0.0000	0.0020	0.2289	21.0561
150	0.0000	0.0000	0.0017	0.1582	13.1952
200	0.0000	0.0000	0.0014	0.1402	13.2802

Table 36: RMS error of the translational component of the second version of the Six-point algorithm along the z-direction

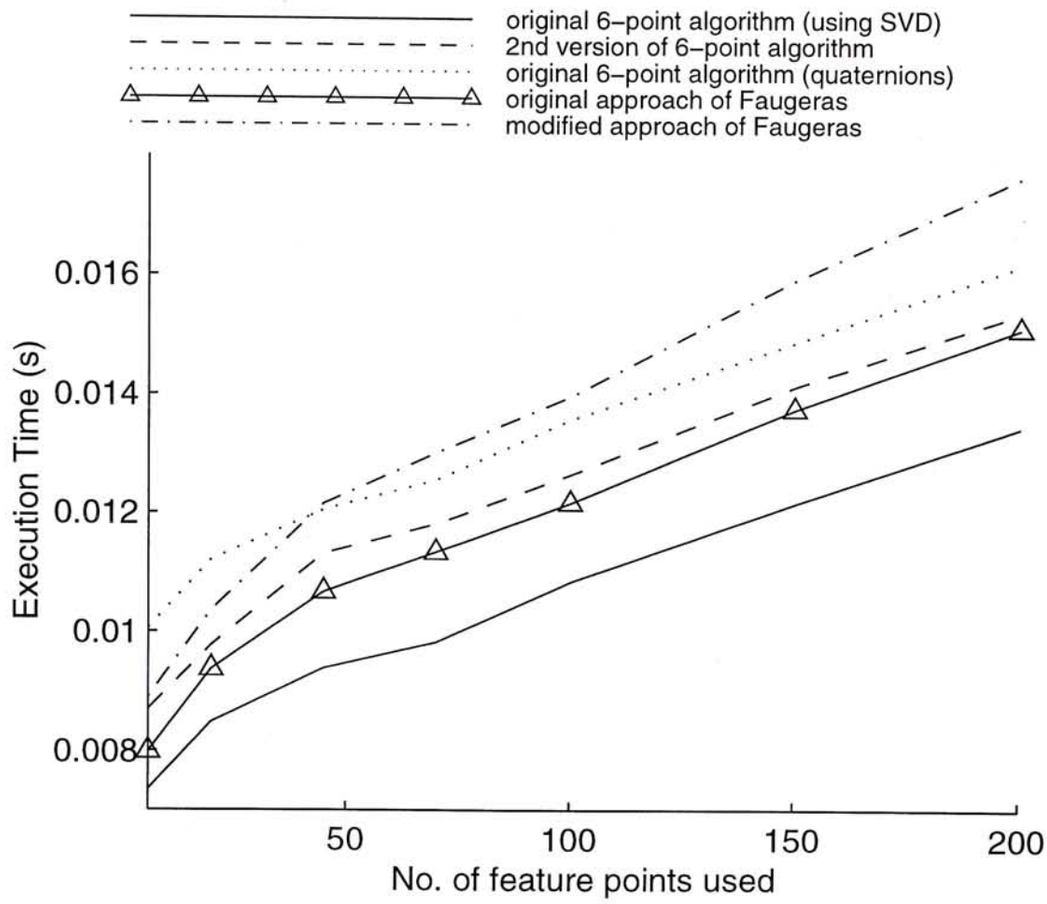


Figure 10: The computation time of the five approaches with fixed motion

1. the original Faugeras's algorithm;

2. the modified Faugeras's algorithm;

3. the original Six-point algorithm using the technique of SVD;

4. the original Six-point algorithm using the technique of quaternions;

5. the second version of the Six-point algorithm using the technique of SVD.

In each approach, the intrinsic parameters are calculated in order to provide a fair environment to compare the computation speed of these five approaches.

From the experiment, the computation time of these five approaches only depends on the number of feature points used, but not depends on the noise level.

So, Figure (10) is used to compare the computation speeds of these five approaches. In Figure (10), the computation time of these five approaches is shown. From this figure, it is observed that the original Six-point algorithm using the technique of SVD in the "Constraint Satisfactory Part" has the fastest computation speed. The original Faugeras's algorithm is the second fastest approach. The second version of the Six-point algorithm is the third fastest approach. The original Six-point algorithm using the technique of quaternion and the modified Faugeras's algorithm are the slowest approaches. On the other hand, it is quite interesting that when the number of feature points used is small (about < 40), the speed of the modified Faugeras's algorithm is faster than that of the original Six-point algorithm using quaternion. However, it is reversed when the number of feature points used is larger than 40.

As a result, the original Six-point algorithm using the technique of SVD has a better performance than the other approaches in terms of accuracy and computation speed.

Experiment Two: With Motion Generated Randomly

This experiment is similar to the first experiment. However, the motions used are randomly generated each time. The rotation is generated randomly from an uniform distribution such that each of the three rotation angles about the three axes (x, y and z axes) are varying between -180^o and $+180^o$, while the translation vector is generated randomly from an uniform distribution in the range $[-2, ..., 2]$. The focal length f is set to be 0.6 units. Also, the distance between two adjacent pixels in the image is 0.0025 units.

In this experiment, the root mean square (RMS) of the percentage error of the translational components and rotational angles will be recorded over 100 trials for the same number of feature points and noise level.

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0	0.0	1.1	85.8	809.7
20	0.0	0.0	0.1	15.0	130.9
45	0.0	0.0	0.1	5.0	75.5
70	0.0	0.0	0.0	3.3	78.0
100	0.0	0.0	0.0	9.3	38.1
150	0.0	0.0	0.0	2.3	61.4
200	0.0	0.0	0.0	1.6	43.8

Table 37: RMS of percentage error of the rotational angle of the original Six-point algorithm about the x-axis

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0	0.0	1.1	86.5	822.2
20	0.0	0.0	0.1	15.0	154.9
45	0.0	0.0	0.1	5.0	63.5
70	0.0	0.0	0.0	3.2	88.5
100	0.0	0.0	0.0	9.3	33.3
150	0.0	0.0	0.0	2.3	84.4
200	0.0	0.0	0.0	1.6	47.5

Table 38: RMS of percentage error of the rotational angle of the second version of the Six-point algorithm about the x-axis

Actually, the result shown in Section 3.2.1.2 is part of the result in this experiment.

As mentioned in Section 3.2, the results of the rotation in the original version

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0	0.0	1.2	130.1	901.2
20	0.0	0.0	0.1	18.9	249.8
45	0.0	0.0	0.1	14.4	215.5
70	0.0	0.0	0.1	4.9	146.2
100	0.0	0.0	0.1	9.7	150.9
150	0.0	0.0	0.0	3.6	141.3
200	0.0	0.0	0.0	9.1	140.4

Table 39: RMS of percentage error of the rotational angle of the original Faugeras's algorithm about the x-axis

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0	0.0	0.3	97.4	362.6
20	0.0	0.0	0.0	1.8	29.4
45	0.0	0.0	0.0	1.6	24.9
70	0.0	0.0	0.0	1.0	37.1
100	0.0	0.0	0.0	2.8	21.1
150	0.0	0.0	0.0	1.4	15.1
200	0.0	0.0	0.0	1.5	7.8

Table 40: RMS of percentage error of the rotational angle of the original Six-point algorithm about the y-axis

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0	0.0	0.3	97.1	322.9
20	0.0	0.0	0.0	1.8	61.2
45	0.0	0.0	0.0	1.6	27.8
70	0.0	0.0	0.0	1.0	37.9
100	0.0	0.0	0.0	2.8	22.1
150	0.0	0.0	0.0	1.4	15.6
200	0.0	0.0	0.0	1.5	9.8

Table 41: RMS of percentage error of the rotational angle of the second version of the Six-point algorithm about the y-axis

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0	0.0	0.4	191.5	723.4
20	0.0	0.0	0.1	2.7	175.4
45	0.0	0.0	0.0	2.3	42.9
70	0.0	0.0	0.0	2.7	33.2
100	0.0	0.0	0.0	1.5	34.3
150	0.0	0.0	0.0	0.7	33.3
200	0.0	0.0	0.0	1.0	42.5

Table 42: RMS of percentage error of the rotational angle of the original Faugeras's algorithm about the y-axis

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0	0.2	1.0	88.7	507.4
20	0.0	0.0	0.1	20.8	65.1
45	0.0	0.0	0.1	9.5	69.0
70	0.0	0.0	0.0	9.7	77.7
100	0.0	0.0	0.0	9.4	60.8
150	0.0	0.0	0.0	9.5	41.2
200	0.0	0.0	0.0	3.8	28.5

Table 43: RMS of percentage error of the rotational angle of the original Six-point algorithm about the z-axis

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0	0.2	1.0	91.6	516.2
20	0.0	0.0	0.1	21.0	121.0
45	0.0	0.0	0.1	9.5	57.7
70	0.0	0.0	0.0	9.7	72.9
100	0.0	0.0	0.0	9.4	43.2
150	0.0	0.0	0.0	9.5	63.3
200	0.0	0.0	0.0	3.7	47.1

Table 44: RMS of percentage error of the rotational angle of the second version of the Six-point algorithm about the z-axis

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0	0.1	9.1	135.7	618.2
20	0.0	0.0	0.2	34.3	306.9
45	0.0	0.0	0.0	16.4	134.1
70	0.0	0.0	0.1	12.1	364.0
100	0.0	0.0	0.0	10.0	82.4
150	0.0	0.0	0.0	19.4	118.1
200	0.0	0.0	0.0	3.5	52.0

Table 45: RMS of percentage error of the rotational angle of the original Faugeras's algorithm about the z-axis

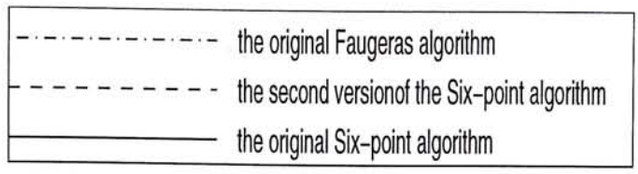
and modified version of Faugeras's approach are the same, so only the rotational angles of the original version of Faugeras's algorithm is discussed in this experiment.

Tables (37)-(45) indicate the RMS of the percentage error of the rotational angles of the two versions of the Six-point algorithm and the original Faugeras's algorithm. Tables (37)-(39) illustrate the result of the rotational angles about the x-axis. Tables (40)-(42) illustrate the result of the rotational angles about the y-axis. Tables (43)-(45) illustrate the result of the rotational angles about the z-axis.

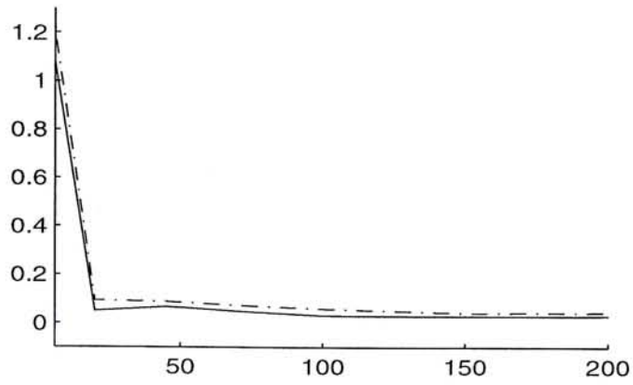
Also, the results, when noise level is equal to 0.05, is shown in Figure 11 too.

From the tables, it is shown that the accuracy of all these three algorithms in calculating the rotational parameters are comparative.

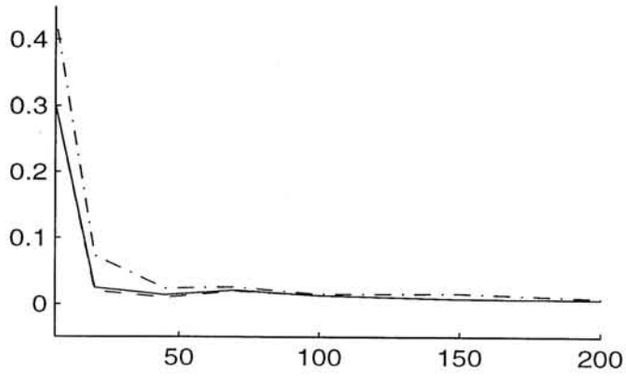
Moreover, tables (46)-(51) illustrate the RMS percentage error of translational components of the original version and the second version of the Six-point



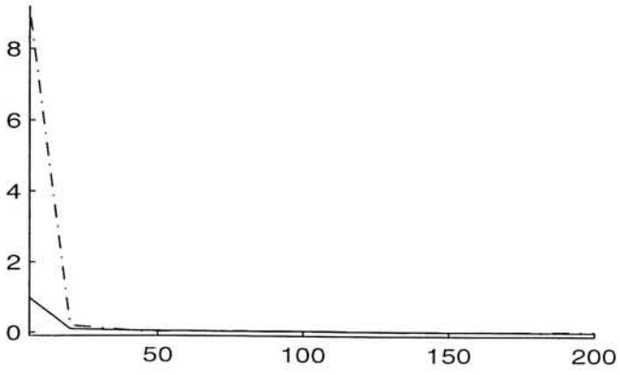
(a)



(b) About x-axis



(c) About y-axis.



(d) About z-axis.

Figure 11: Percentage Error of the rotational angles vs no. of points used when the noise level = 0.05.

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0	0.0	0.4	61.6	489.2
20	0.0	0.0	0.1	2.0	151.8
45	0.0	0.0	0.2	18.1	232.7
70	0.0	0.0	0.0	2.4	46.1
100	0.0	0.0	0.1	1.6	55.8
150	0.0	0.0	0.0	1.7	60.8
200	0.0	0.0	0.0	2.7	51.2

Table 46: RMS of percentage error of the translational component of the original Six-point algorithm long the x-direction

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0	0.0	0.4	75.3	741.1
20	0.0	0.0	0.1	2.0	155.6
45	0.0	0.0	0.2	18.1	232.0
70	0.0	0.0	0.0	2.3	44.6
100	0.0	0.0	0.1	1.6	57.9
150	0.0	0.0	0.0	1.7	59.3
200	0.0	0.0	0.0	2.7	49.5

Table 47: RMS of percentage error of the translational component of the second version of the Six-point algorithm long the x-direction

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0	0.0	0.2	26.0	374.1
20	0.0	0.0	0.0	1.5	29.6
45	0.0	0.0	0.0	4.5	68.4
70	0.0	0.0	0.0	2.0	24.8
100	0.0	0.0	0.0	3.0	46.4
150	0.0	0.0	0.0	0.8	12.4
200	0.0	0.0	0.0	1.3	7.7

Table 48: RMS of percentage error of the translational component of the original Six-point algorithm long the y-direction

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0	0.0	0.2	27.6	108.3
20	0.0	0.0	0.0	1.5	30.9
45	0.0	0.0	0.0	4.6	56.7
70	0.0	0.0	0.0	2.0	21.0
100	0.0	0.0	0.0	3.0	50.8
150	0.0	0.0	0.0	0.7	17.0
200	0.0	0.0	0.0	1.3	6.0

Table 49: RMS of percentage error of the translational component of the second version of the Six-point algorithm long the y-direction

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0	0.0	0.1	24.8	79.7
20	0.0	0.0	0.0	0.1	1.6
45	0.0	0.0	0.0	0.0	1.0
70	0.0	0.0	0.0	0.0	0.8
100	0.0	0.0	0.0	0.0	0.8
150	0.0	0.0	0.0	0.0	0.7
200	0.0	0.0	0.0	0.0	0.7

Table 50: RMS of percentage error of the translational component of the original Six-point algorithm long the z-direction

No. of pt.	Noise Level				
	0	0.005	0.05	0.5	2
6	0.0	0.0	0.1	26.4	82.2
20	0.0	0.0	0.0	0.1	1.7
45	0.0	0.0	0.0	0.0	1.0
70	0.0	0.0	0.0	0.0	0.8
100	0.0	0.0	0.0	0.0	0.8
150	0.0	0.0	0.0	0.0	0.7
200	0.0	0.0	0.0	0.0	0.7

Table 51: RMS of percentage error of the translational component of the second version of the Six-point algorithm long the z-direction

algorithm. Also, tables (16)-(21) show the RMS percentage error of translational components of the original Faugeras's algorithm and the modified Faugeras's algorithm. From these 12 tables, it is shown that the accuracies of the two versions of the Six-point algorithm and the modified Faugeras's algorithm are comparative. However, the percentage error of translational components of the original Faugeras's algorithm is much larger than that of the other three approaches. This phenomenon agrees with the result in the first experiment using fixed motion.

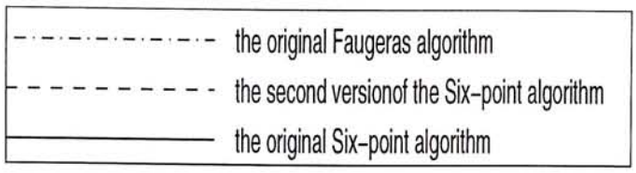
Also, the results, when noise level is equal to 0.05, is shown in Figure 12 too.

On the other hand, the computation time of the following approaches are recorded and compared:

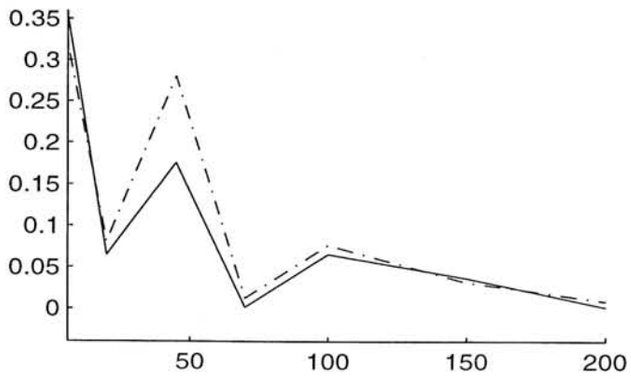
1. the original Faugeras's algorithm;
2. the modified Faugeras's algorithm;
3. the original Six-point algorithm using the technique of SVD;
4. the original Six-point algorithm using the technique of quaternions;
5. the second version of Six-point algorithm using the technique of SVD.

In each approach, the intrinsic parameters are calculated in order to provide a fair environment to compare the computation speed of these five approaches.

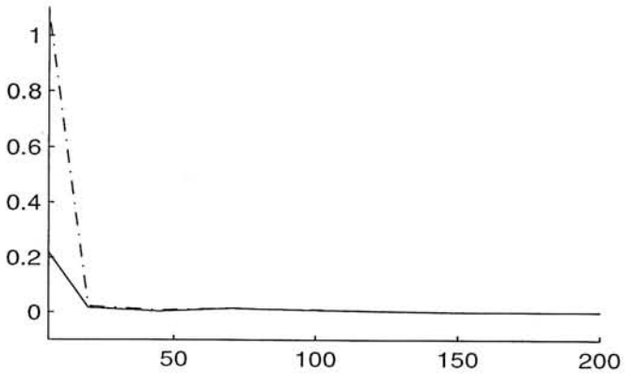
Figure (13) illustrates the computation time of all the five approaches. From this figure, the result is similar to that in the first experiment when fixed motion is used. The original Six-point algorithm using SVD in the "Constraint Satisfactory Part" has the fastest computation speed.



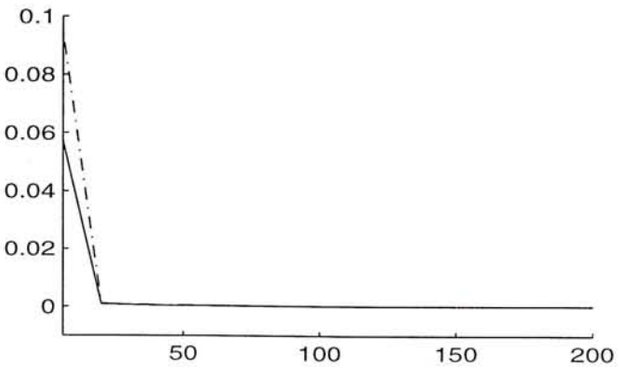
(a)



(b) Along x-axis



(c) Along y-axis.



(d) Along z-axis.

Figure 12: Percentage Error of the translational components vs no. of points used when the noise level = 0.05.

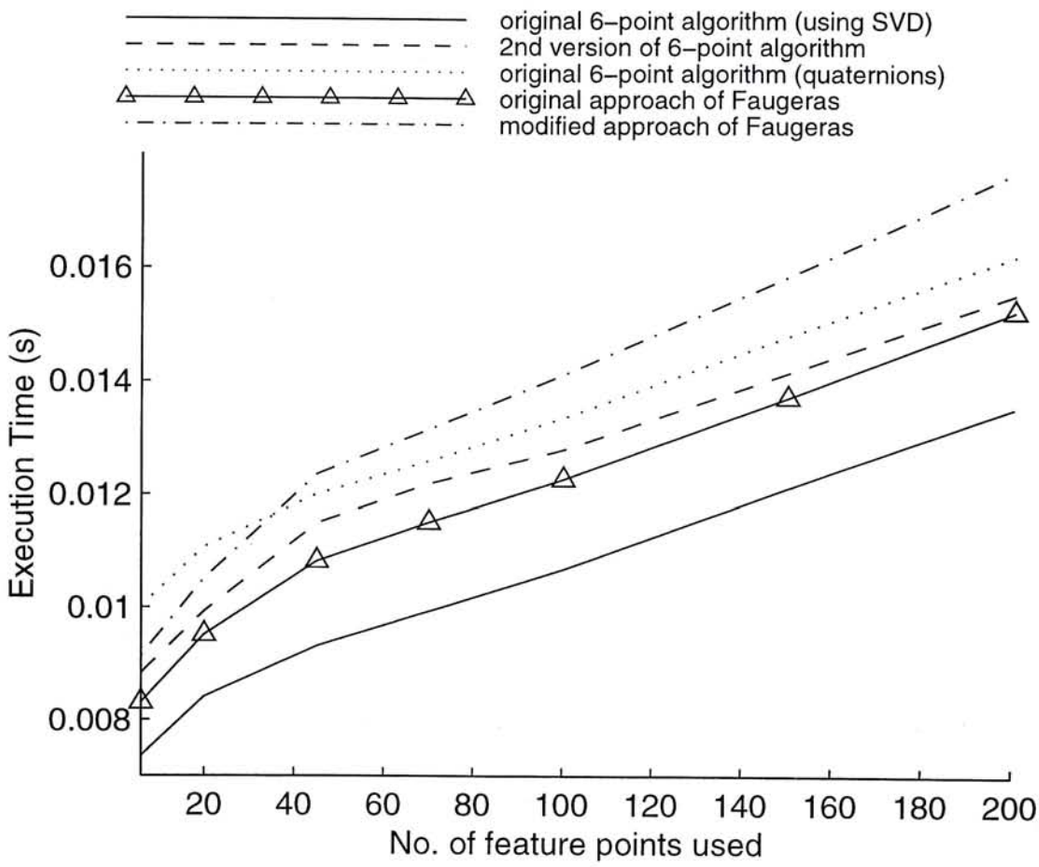
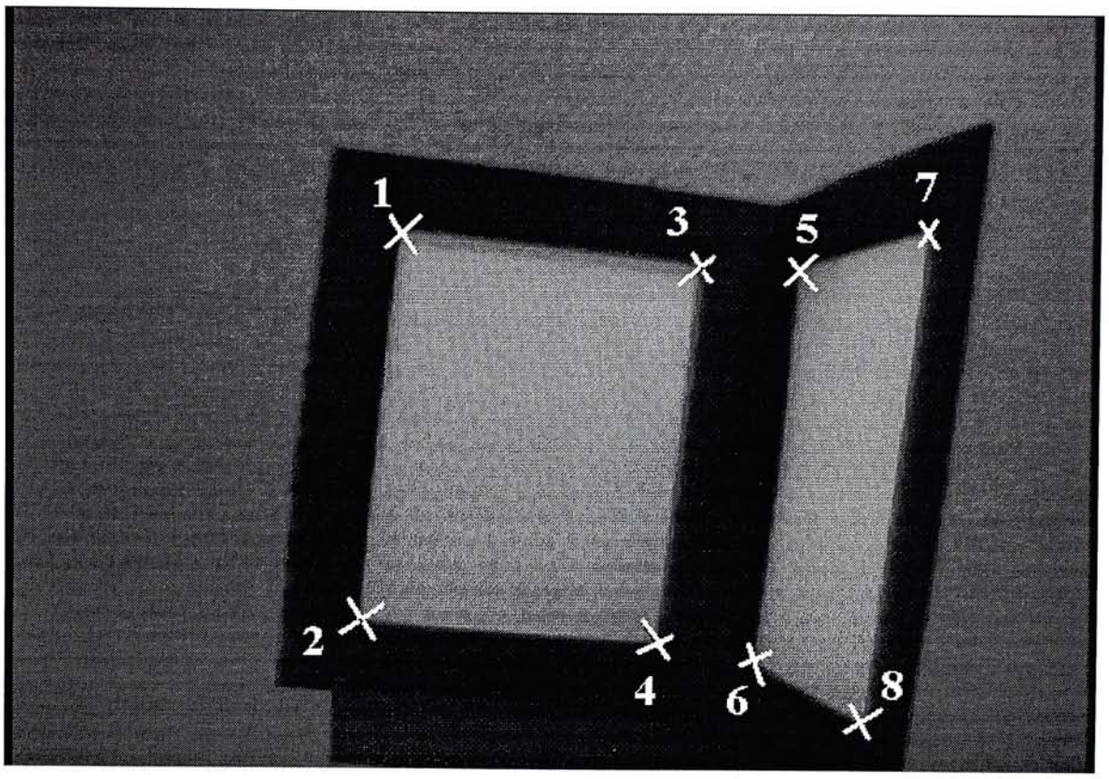
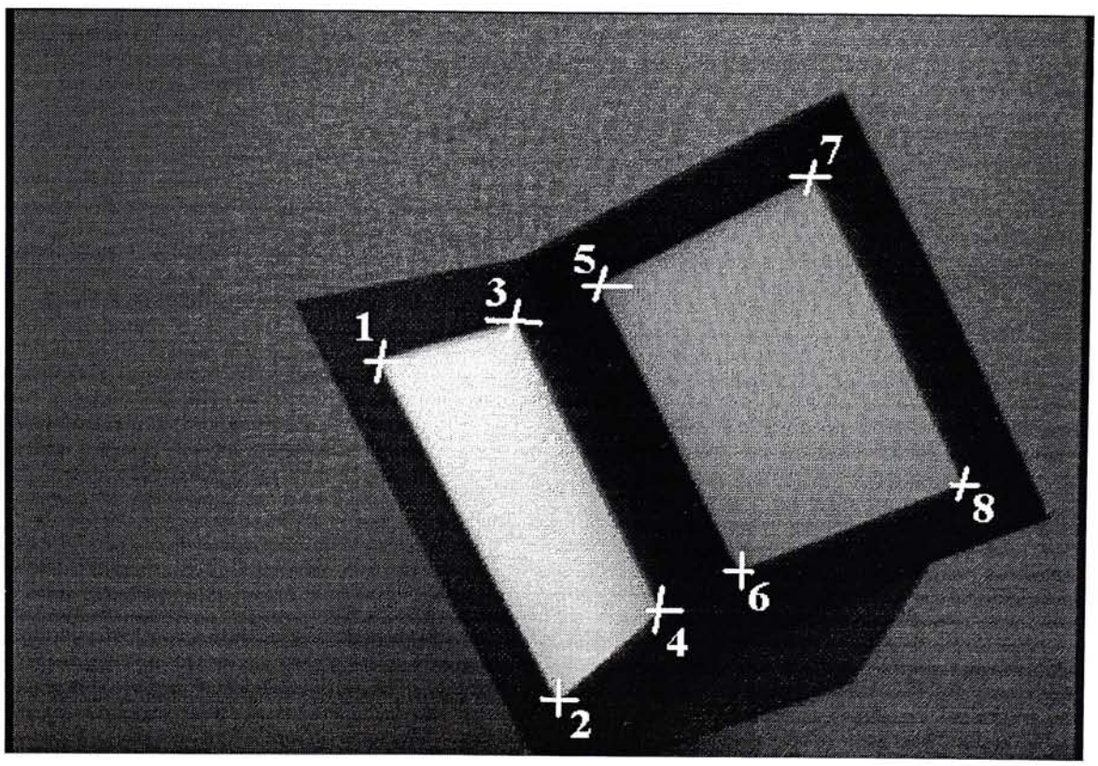


Figure 13: The computation time of the five approaches with motion generated randomly.

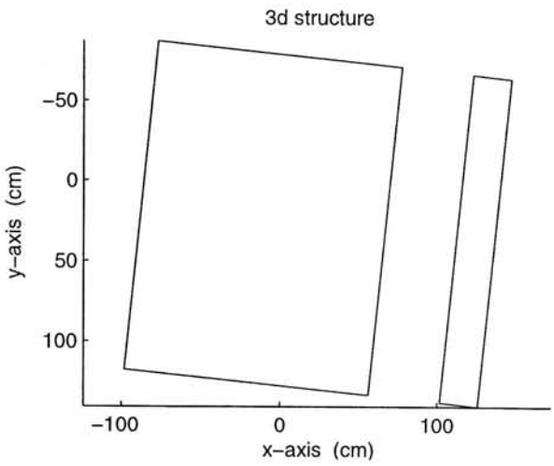


(a) at time k

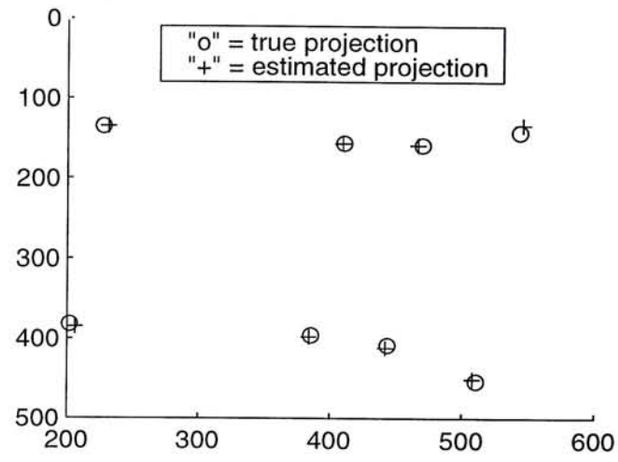


(b) at time k+1

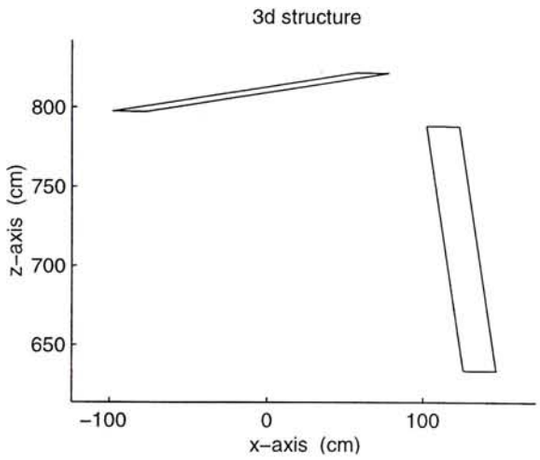
Figure 14: Object used in real data experiment



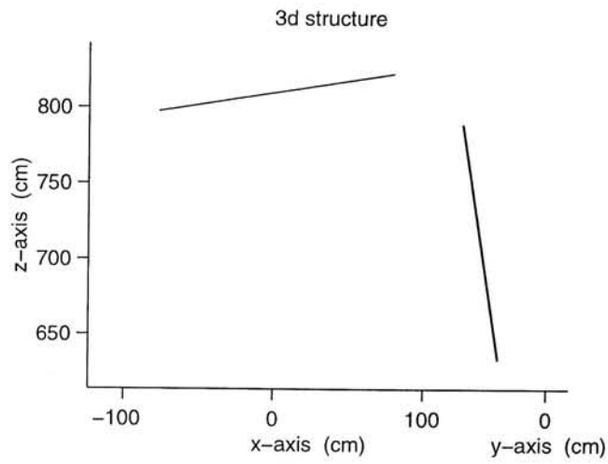
(a) front view



(b) the 2D image

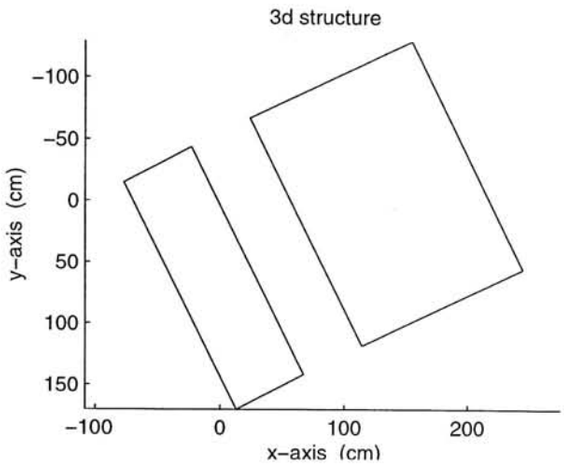


(c) top view

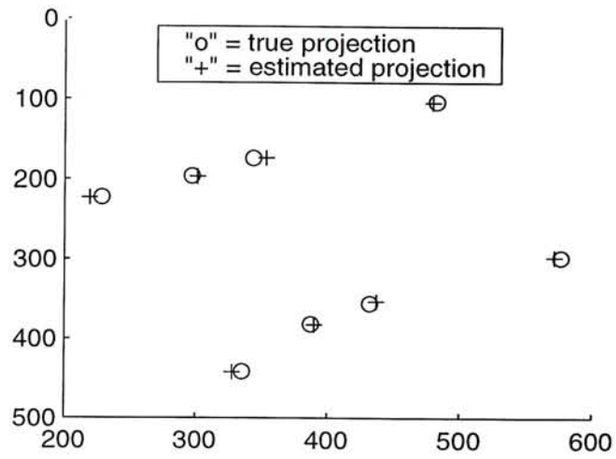


(d) the figure indicating the angle between the two planes

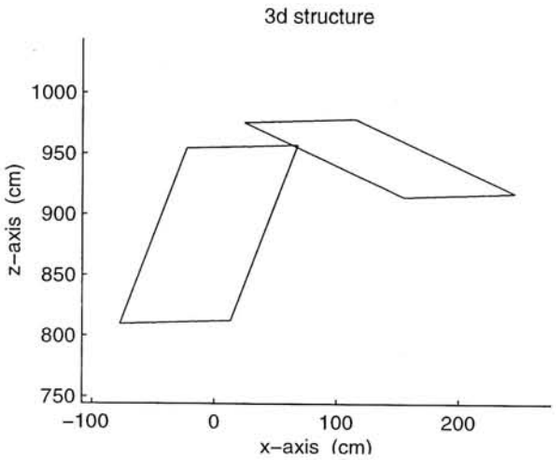
Figure 15: Result of the image in Figure (14 a)



(a) front view



(b) the 2D image



(c) top view

Figure 16: Result of the image in Figure (14 b)

3.3.4.2 With Real Data

In this experiment, the images of an object are taken at time k and time $k+1$, which are shown in Fig (14 a) and (14 b) respectively. In each figure, 8 points (each is marked with a cross) are used as the image points. These points are used to calculate the motion of this object, hence the 3D coordinates of the corresponding model points in this object can be found. After applying the *original Six-point algorithm* to the data, the result based on the image in Fig (14 a) is shown in Fig (15).

Fig (15 a) shows the front view of the reconstructed object. Fig (15 b) indicates the error between the captured image (true data) and the reconstructed data. Moreover, Fig (15 c) indicates the top view of the object. Finally, (15 d) indicates the angle between the two planes, which is 90^o and it is the same as the real angle. So, it shows that there is no deformation in the reconstructed object.

The result for Fig (14 b) is shown in Fig (16).

3.3.5 Summary of the Six-Point Algorithm

In this section, both synthetic data and real data is used to verify the two versions of Six-point algorithm. Also, a comparison between the two versions of Six-point algorithm, original Faugeras's algorithm and the modified version of Faugeras's algorithm is performed. It is observed that:

1. The two versions of the Six-point algorithm and the modified Faugeras's algorithm have comparative performances in terms of accuracy of the results. Also, the accuracy in the calculations of the translational parameters of these three algorithms are better than that of original Faugeras's algorithm.

2. The original Six-point algorithm has the fastest computation speed among the other four approaches.

3. The accuracy of these algorithms increases with the increase of number of feature points used.

For the original Faugeras's algorithm, Faugeras [12] stated that the nullspace is of dimension greater than 1 and there is an infinite number of solutions to Equation (2.22) when the rank of χ in Equation (2.22) is less than 11. Also, he showed that the rank of χ is equal to 8 when all the feature points are coplanar in 3D space [12]. Therefore, the 3D feature points should not be chosen to lie in the same plane.

The experiments using coplanar 3D feature points have also been done in order to test these four approaches. However, the error of the results of these four approaches is very bad even if there is no noise added to the systems. Also, the ranks of the matrix χ in Equation (2.22) and matrix $\mathbf{A}$ in Equation (3.51) are equal to 8 in these experiments. Actually, the ranks of the two matrices (χ in Equation (2.22) and $\mathbf{A}$ Equation (3.51)) are equal because both these two matrices are depending on the 3D and 2D coordinates of the feature points and they are only different in the orders of some columns.

As a result, *the Six-point algorithm is not suitable to the case when all 3D feature points are on the same plane as that happens in Faugeras's algorithm (both original and modified versions).* Therefore, users should pay attention in selecting the feature points.

In the following parts of this thesis, the Six-point algorithm is applied in several areas. Since the performance of the original Six-point algorithm in terms of accuracy and computation speed is better than the second version. Therefore, the original Six-point algorithm will be applied in those areas. In the remaining parts of this thesis, the term "Six-point algorithm" stands for the "original Six-point algorithm".

3.3.6 A Visual Tracking System by using Six-point Algorithm

In this subsection, a real-time visual tracking system by using the Six-point algorithm is introduced. A CCD camera is used as an input device, which will capture images of the target object and then determine the rigid motion of the target. The outline of the system is shown in Figure (17). Since the Six-point algorithm is used, the model (3D measurements of the features) of the target is required. The detailed operation of this system is described in the following:

1. Firstly, the camera captures the first image of the target object (the input device). Then a user is required to input the locations of the feature points (which are used to represent the model of the target object) in this image. Then the system will continuously capture the images of the target.

2. Secondly, once an image is captured, the system will automatically search the projection of all the feature points by using the correlation coefficient technique [73]. Then the system will calculate the new 3D position and orientation of the target object. The motion parameters of the target will also be determined. A feedback of these results will be given to the user. In this thesis, a 3D human face is rendered and displayed on the screen. The motion of this 3D human face represents the true motion of the target (input device).

In this thesis, this real-time visual tracking system is implemented in a PENTIUM II-233. All the programs are coded in Visual C++ 5.0. The 3D human head is rendered by OpenGL. The model of the video grabber card is BS600 produced by BOSER. By using a 3D accelerator card (this card includes the function of a display card), the frame rate of tracking is about 10 fps (frames per second). However, this frame rate is mainly determined by the computation time of the

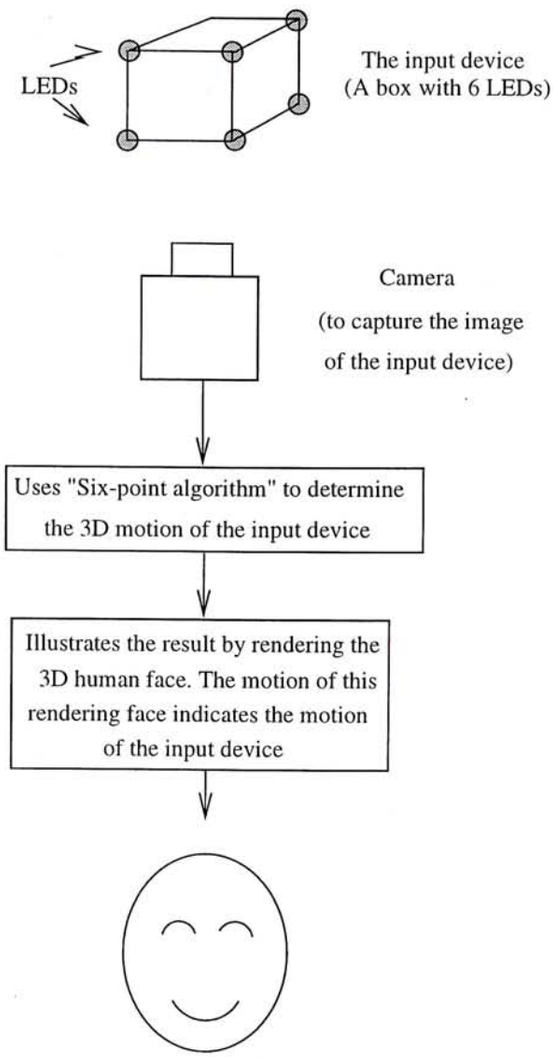


Figure 17: The outline of the visual tracking system.

correlation coefficients [74][73] and rendering of the 3D human face.

However, there is a weakness of this system. If the motion of the target is too large, the projection of the feature points may not be determined correctly because:

1. The rotation of the target object is so large that the patterns near the feature points may have a large difference; or

2. Some feature points may move out of the search windows (a search window is to limit the searching inside the area specified. The larger the area, the

	TL-algorithm	Six-point algorithm
specified for PnP problem	$n = 4$	$n \geq 6$
iterative/linear method	iterative	linear
require initial guess	Yes	No

Table 52: Properties of the two algorithms.

slower the searching).

This problem can be solved by using a faster computer, video grabber card and display card. Also, a better pattern recognition technique can be used in order to increase the accuracy of finding the point correspondences.

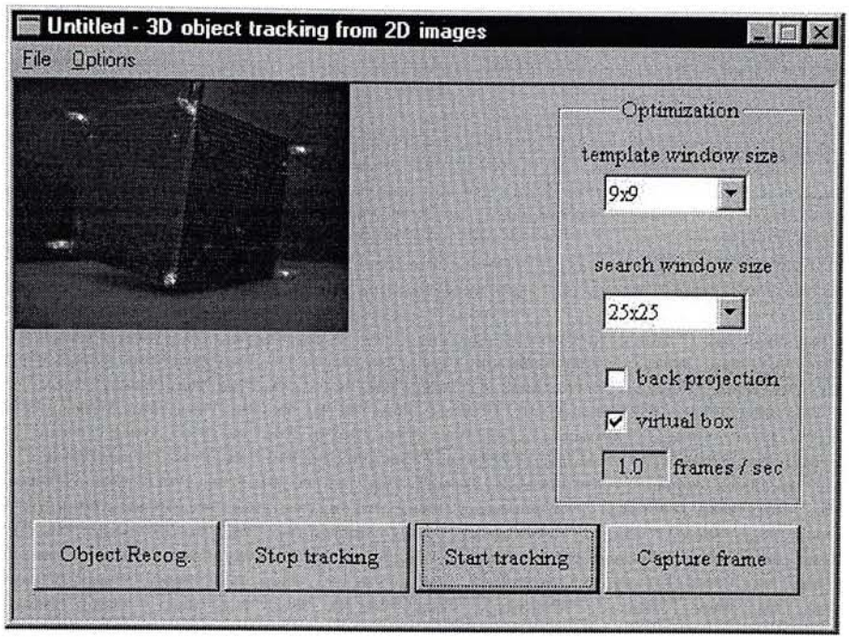
However, this system is only used to illustrate the possibility of using the Six-point algorithm to implement a visual tracking system in a PC (personal computer). Increasing the performance of this system by improving the searching of the point correspondences and updating the hardware will not be the aim of this thesis.

Figure (18) - (20) illustrate the results of the visual tracking system between time k and $k + 2$.

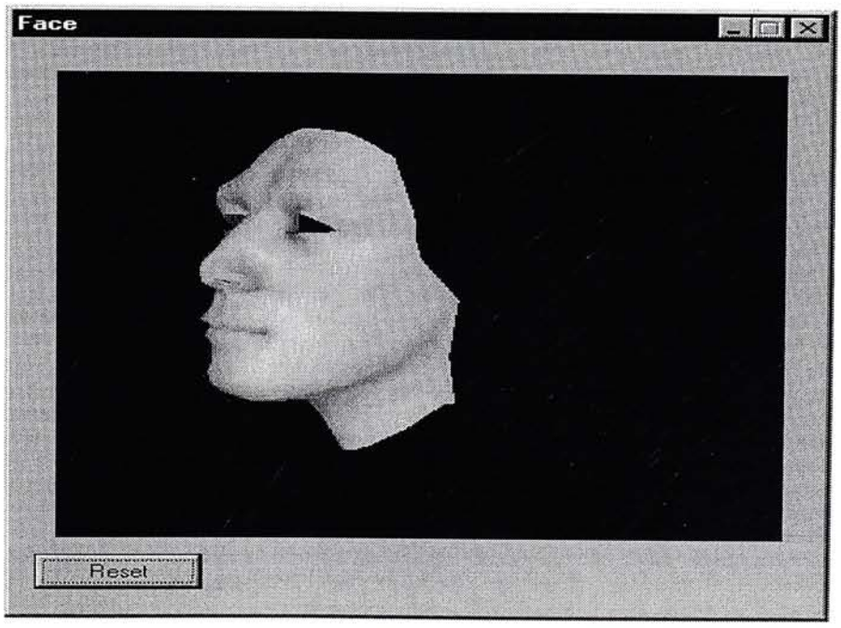
Although the input device used in this system is simple. A complex object can be used instead, for example a real human head.

3.4 Comparison between TL-algorithm and Six-point Algorithm developed

These two algorithms are proposed by us for model-based motion analysis with 2D-to-3D point correspondences. Their properties are shown in Table (52):

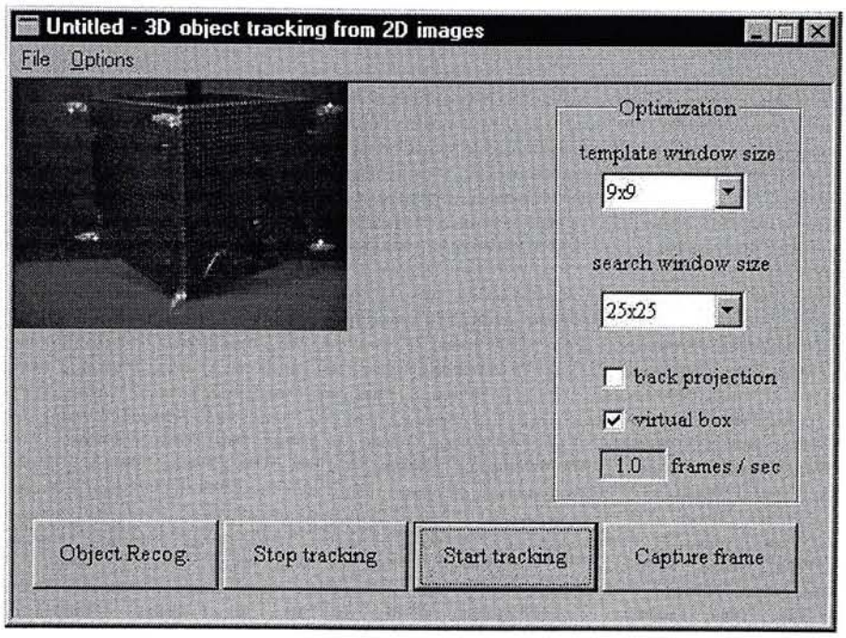


(a) This subfigure shows the control panel of the visual tracking system. The box inside the capture window is used as the input device. The six LEDs are regarded as the six feature points of the box. This image is captured at time k.

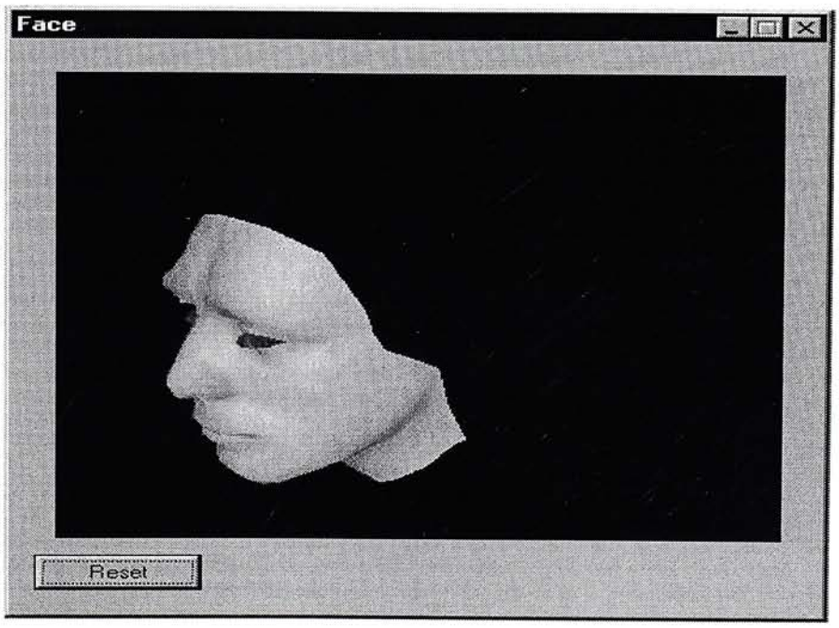


(b) The 3D human face indicates the position and the orientation of the input device at time k.

Figure 18: The experimental result of the visual tracking system at time k

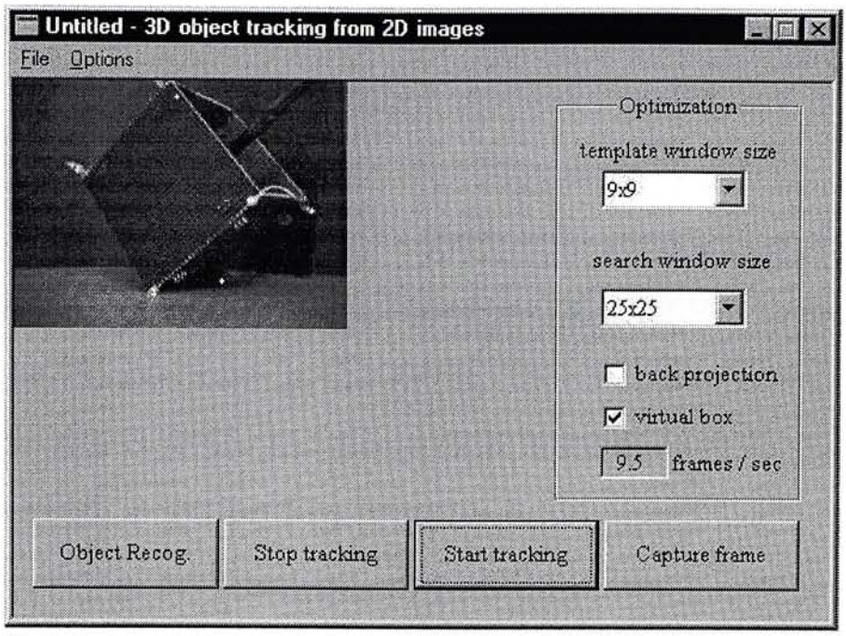


(a) This image containing the input device is captured at time $k+1$.

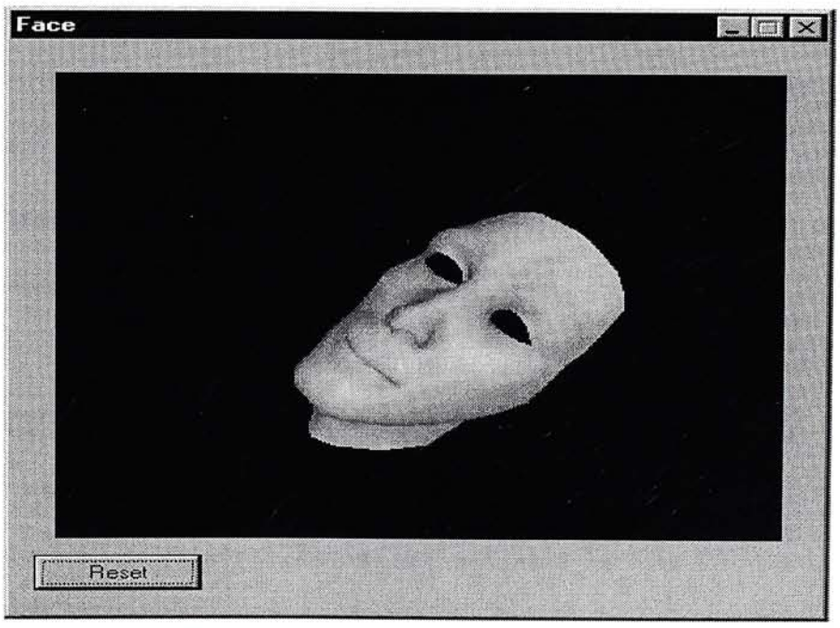


(b) The 3D human face indicates the position and the orientation of the input device at time $k+1$.

Figure 19: The experimental result of the visual tracking system at time k+1



(a) This image containing the input device is captured at time $k+2$.



(b) The 3D human face indicates the position and the orientation of the input device at time $k+2$.

Figure 20: The experimental result of the visual tracking system at time k+2

Since the TL-algorithm is an iterative method, and an initial guess is required, the computation speed is quite slow. Both speed of the computation and the accuracy heavily reliess on the value of the initial guess. If the initial guess is very close to the answer, the computation time will be short too. However, if the initial guess is very far from the answer, the computation time will be very long and there may be a chance for the system to converge to another local minimum since multiple solutions may exist in the P4P problem [25]. Therefore, the initial guess is the key of the TL-algorithm. In this thesis, the initial guess of the TL-algorithm is set to be the answer in the last frame. It is based on the assumption that the motion of the target object between two consecutive frames is not so large, so the answers between these two frames are similar.

The Six-point algorithm is a linear algorithm, so the initial guess is not required. Therefore the robustness is greatly improved. Also, the computation speed of the Six-point algorithm is quite fast and it only depends on the number of feature points used in the process. However, the Six-point algorithm can only be used in PnP problems, for $n \geq 6$. Some experiments have been done to show the performance of the Six-point algorithm in PnP problems, for $n < 6$. However, the rank of the matrix $\mathbf{A}$ in Equation (3.51) is less than 11 for $n < 6$. So, multiple solutions exist in this case [12]. Also, initial guess is not required in the Six-point algorithm, the answer cannot be guided to a correct solution, so the answer obtained may not be the correct one and it may be one of the multiple solutions. As a result, the Six-point algorithm is not suitable for the PnP problems, where $n < 6$.

Therefore, TL-algorithm and Six-point algorithm are used to solve different kinds of model-based motion estimation problem with 2D-to-3D point correspondences. No one can take place of another.

3.5 Summary

In this chapter, two approaches are proposed for model-based motion analysis with 2D-to-3D point correspondences. The first one is an iterative solution for the P4P problem. It is called TL-algorithm. The other one is a direct and linear method for the 2D-to-3D point correspondence problem with six or more feature points in general position. It is called the Six-point algorithm. Also, an enhancement to Faugeras's algorithm [12] is introduced. This modified Faugeras's algorithm improves the accuracy of the the translational components in the result of the original version of Faugeras's algorithm.

- For the TL-algorithm, experiments have been performed to verify the algorithm. Also, a comparison between Lowe's algorithm [10] and the TL-algorithm has been done in terms of their computation time. From the result, it is observed that the computation time of the TL-algorithm is faster than that of Lowe's algorithm.

- For the enhancement of the Faugeras's algorithm, a new method is proposed to estimate the translational part of the motion. Also, experiments have been done to compare the performances of original Faugeras's algorithm and this modified version in terms of the accuracy of the translational components in the results. From the result, it is observed that the accuracy of this modified version is better than the original one especially when the noise level in the system is high.

- For the Six-point algorithm, there are two versions. The Six-point algorithm is similar to Faugeras's algorithm. They are different in applying different constraints in the step of constrained optimization. Also, they use different methods to preserve the orthogonality of the rotation matrix.

On the other hand, there are two versions for the Six-point algorithm. Both versions can also be divided into two parts: 1) The linear part; 2) the constraint satisfactory part. These two versions are only different in applying different constraints in the step of constrained optimization in the linear part. In the linear part, a rough solution for the motion of the object can be obtained. In the constraint satisfactory part, the constraint of the orthogonality of the rotation matrix is applied to refine the result in order to prevent deformation of the reconstructed object. Moreover, the technique of SVD is proposed to calculate the rotation matrix in order to enhance the speed of the computation.

Also, in this chapter, experiments have been done by using both synthetic data and real data to verify our algorithm. A comparison has been done to compare the performances of the two versions of the Six-point algorithm, the original Faugeras's algorithm and the modified version of Faugeras's algorithm. From the results, we find that the accuracy of the two versions of the Six-point algorithm and the modified Faugeras's algorithm are comparative and are better than that of the original Faugeras's algorithm. Also, it is shown that the computation speed of the original Six-point algorithm using the technique of SVD in the constraint satisfactory part is faster than the other approaches.

Moreover, the original Six-point algorithm is used to implement a real-time visual tracking system. An experiment has been done to show the performance of this system.

CHAPTER 4

MOTION-BASED SEGMENTATION

In this chapter, motion-based segmentation in model-based computer vision is discussed. In Section 4.1, a new approach for motion-based segmentation with 3D-to-3D point correspondences is proposed. In Section 4.2, another approach for the problem with 2D-to-3D point correspondences is presented. Both of these algorithms can segment all the feature points into a number of clusters and the motion of each cluster is determined (In this chapter, a cluster contains the feature points having the same motion). Unlike other approaches proposed by other scholars, the number of clusters can also be automatically determined by these two algorithms. On the other hand, experiments have been performed to verify these two new approaches.

4.1 A new Approach with 3D-to-3D Point Correspondences

In this section, a novel algorithm is proposed to determine the number of motions in the system and to perform motion-based segmentation automatically. Based on the segmentation result, the motions of multiple moving rigid objects are computed. It is actually an approach for the motion-based segmentation with 3D-to-3D point correspondences. The 3D coordinates of points on the rigid objects are used as the 3D features. They can be pre-computed by using a stereo vision system or other range-finding devices. Since the aim of this chapter is to handle the problem of multiple rigid motions, the details of the way of obtaining the 3D

points is not included. Also, for simplicity, we assume the feature selection and tracking have been performed prior to the execution of our algorithm.

Moreover, for each individual group with the common-motion, the Arun method [15] is used to evaluate the motion. Unlike the algorithm suggested by Hung et al. [48], the technique of incremental clustering [75] is used instead of the RANSAC-based (RANSAC stands for Random Sample Consensus) [25] clustering technique.

4.1.1 Algorithm

Supposing that we are given two sets of 3D points, $\{\mathbf{p}_i\}$ and $\{\mathbf{p}'_i\}$ for $i = 1, \ldots, N$. $\mathbf{p}_i = [x_i\,,\, y_i\,,\, z_i]^T$ is the 3D coordinates of the $i-th$ feature point at time k while $\mathbf{p}'_i = [x'_i\,,\, y'_i\,,\, z'_i]^T$ is the 3D coordinates of the $i-th$ feature point at time $k+1$. Each feature point has its own motion. When several feature points are belonging to the same rigid body, they will have the same motion. If the $i-th$ feature point is belonging to the $j-th$ rigid body, the following equation can be obtained:

$$\mathbf{p}'_i = \mathbf{R}_j \mathbf{p}_i + \mathbf{t}_j, \tag{4.79}$$

where $\mathbf{R}_j$ and $\mathbf{t}_j$ are the rotation matrix and the translation matrix respectively, which represent the motion of the $i-th$ feature point between time k and $k+1$.

In order to determine the motion of the feature points, the feature points of the same motion should be grouped together first. So, we should group points with the same motion into a cluster. However, the number of clusters is not given. Therefore, the following problems need to be solved:

1. To find the number of clusters, each contains feature points of the same motion;

2. To segment all the feature points into a number of clusters;

3. To compute the motion of each cluster.

As mentioned above, we do not know a prior which feature points belong to which motion cluster and the number of clusters is also an unknown, so advanced clustering techniques are required. We chose the technique of incremental clustering because it can handle this situation easily. Also, the algorithm proposed by Umeyama [21], which is an enhancement of the algorithm proposed by Arun et al. [15], is used to evaluate the motion in each cluster. Our proposed algorithm is as follows:

Firstly, for the $i-th$ feature point at time $k+1$, where $i = 1, \ldots, N$, we select s ($s \geq 2$) nearest feature points at that instant. For each combination of 3 out of the $s+1$ feature points (including the $i-th$ feature point and its s nearest neighbours selected), the motion is computed by using the Umeyama [21] method. Since the motion of the $i-th$ feature point is considered, the $i-th$ feature point must be chosen in each motion calculation. Therefore, there are totally ${}_sC_2$ different motions as the candidates for the $i-th$ feature point. Since, the $i-th$ feature point can only have one motion at an instant, only one motion is selected from the ${}_sC_2$ different motions. The chosen motion should minimize the following cost function:

$$F(\mathbf{R}, \mathbf{t}) = \left\| \mathbf{p}'_i - (\mathbf{R}\mathbf{p}_i + \mathbf{t}) \right\|^2 . \tag{4.80}$$

Then the optimal rotation matrix and translation matrix of the $i-th$ feature point are respectively $\mathbf{R}^i$ and $\mathbf{t}^i$.

After all the feature points have their own optimal motions, the incremental clustering technique is used with these motion parameters to segment the feature points. However, the rotation matrices cannot be used directly in the clustering process because rotation matrices should be 3×3 orthogonal matrices (i.e.

$(\mathbf{R}^i)^T(\mathbf{R}^i) = (\mathbf{R}^i)(\mathbf{R}^i)^T = \mathbf{I}$ and $\det(\mathbf{R}^i) = 1$, for $i = 1, \ldots, N$). So the 9 elements in a rotation matrix are mutually dependent. Therefore, for each feature point, the three rotation angles about the three rotation axes (x, y and z axes) are extracted from the rotation matrix.

For the $i-th$ feature point, the three rotation angles about the x, y and z axes are respectively γ^i, β^i and α^i. They are given by the following formulae:

$$\beta^i = \sin^{-1}(r_{13}), \tag{4.81}$$

$$\gamma^i = \begin{cases} \cos^{-1}(r_{33}/\cos(\beta^i)) & \text{if } -r_{23}/\cos(\beta^i) \geq 0 \\ -\cos^{-1}(r_{33}/\cos(\beta^i)) & \text{otherwise} \end{cases}, \tag{4.82}$$

$$\alpha^i = \begin{cases} \cos^{-1}(r_{11}/\cos(\beta^i)) & \text{if } -r_{12}/\cos(\beta^i) \geq 0 \\ -\cos^{-1}(r_{11}/\cos(\beta^i)) & \text{otherwise} \end{cases}, \tag{4.83}$$

where r_{ab} is the element of $\mathbf{R}^i$ in the $a-th$ row and $b-th$ column.

However, many combinations of the three rotation angles can be formed by the same rotation matrix. In order to eliminate ambiguity, the rotation angles are restricted by the following rules:

$$\begin{aligned} &\beta^i \text{ is between } -\frac{\pi}{2} \text{ and } \frac{\pi}{2}; \\ &\alpha^i \text{ is between } -\pi \text{ and } \pi; \\ &\gamma^i \text{ is between } -\pi \text{ and } \pi. \end{aligned} \tag{4.84}$$

As a result, the clustering process can be performed for a sample in the six-dimensional motion parameter space (3 for the rotation angles and 3 for the translational parameters).

The motion of the $i-th$ feature point is described by the following vector:

$$\varpi_i = \left[\gamma^i, \ \beta^i, \ \alpha^i, \ t_x^i, \ t_y^i, \ t_z^i \right]^T, \tag{4.85}$$

where $\left[t_x^i, \ t_y^i, \ t_z^i \right]^T = \mathbf{T}^i$.

However, the three rotation angles cannot be used directly in clustering. It is because there may be a very large difference between the magnitudes of the rotation angles and that of the translational parameters, the translational parameters may be dominant and the result of the clustering may be mostly determined by the translational parameters. Therefore, the 6D motion vectors should be normalized. In this paper, we normalize the motion vectors as the following way:

$$\mathbf{v}_i = \left[w\gamma^i, \ w\beta^i, \ w\alpha^i, \ t_x^i, \ t_y^i, \ t_z^i \right]^T, \tag{4.86}$$

where

$$w = \frac{\sqrt{\frac{1}{N}\sum_{l=1}^{N}\left((t_x^l)^2 + \left(t_y^l\right)^2 + (t_z^l)^2\right)}}{\sqrt{\frac{1}{N}\sum_{l=1}^{N}\left((\alpha^l)^2 + (\beta^l)^2 + (\gamma^l)^2\right)}}. \tag{4.87}$$

From the experiment, we find that the normalization of the motion vectors gives a more reliable and accurate result. Then the normalized 6D motion vectors are used in the incremental clustering procedure as in the following:

1. Let $\mathbf{m}_1 = \mathbf{v}_1$, where $\mathbf{m}_l$ is the mean motion vector of the $l-th$ cluster.

2. For $i = 2$, number of cluster $\eta = 1$, number of members in the first cluster $\sigma_1 = 1$.

3. Calculate the distances between the $i-th$ feature point and the means of all clusters (i.e. $d_{ij} = \|\mathbf{v}_i - \mathbf{m}_j\|^2$,for $j = 1, \ldots, \eta$). And determine the smallest d_{ij} for all j.

4. If the smallest distance $d_{i\lambda} < \varepsilon$ then this $i-th$ feature point is belonging to the $\lambda - th$ cluster. The new mean motion vector of the $\lambda - th$ cluster is:

$$\mathbf{m}_{\lambda}^{(new)} = \frac{\sigma_{\lambda}^{(old)}\mathbf{m}_{\lambda}^{(old)} + \mathbf{v}_i}{\sigma_{\lambda}^{(old)} + 1}, \tag{4.88}$$

then the number of the members in this cluster is updated (i.e. $\sigma_{\lambda}^{(new)} = \sigma_{\lambda}^{(old)} + 1$).

5. However, if the smallest distance $d_{i\lambda} \geq \varepsilon$ then a new cluster is generated and its mean is assigned to be the normalized motion vector of the $i-th$ feature point (i.e. $\eta^{(new)} = \eta^{(old)} + 1$, $\mathbf{m}_{\eta} = \mathbf{v}_i$ and $\sigma_{\eta} = 1$).

6. Consider another feature point and repeat the steps 3-5 until all feature points are examined.

From the experiments, the result is good when the threshold ε is set to be:

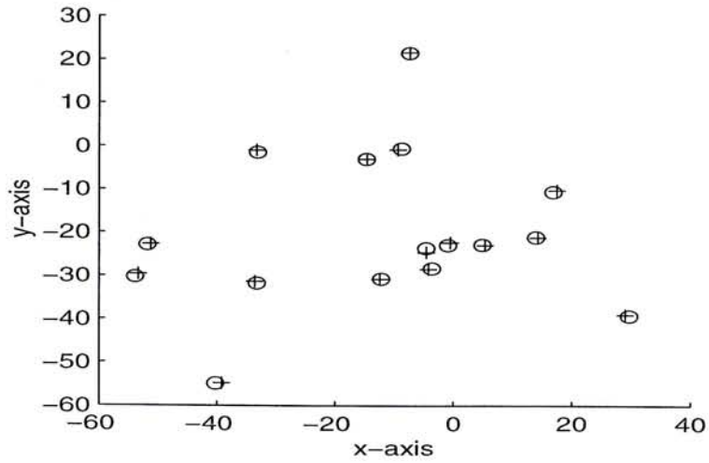
$$\varepsilon = \sqrt{\rho}, \tag{4.89}$$

where ρ is the largest eigen value of the covariance matrix of the motion vectors.

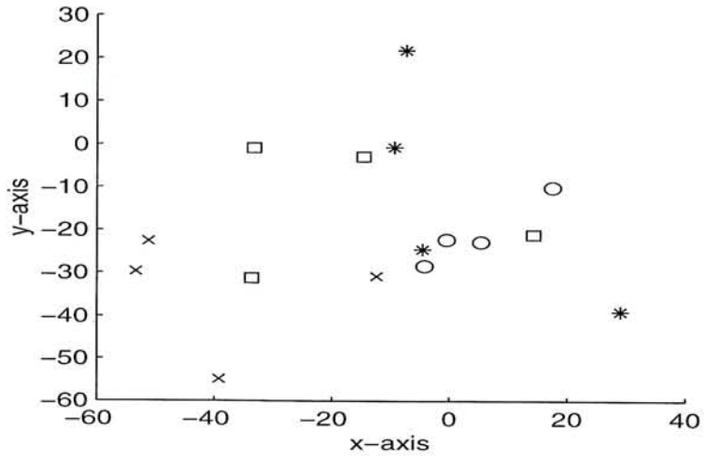
Then the motions of all the clusters can be obtained. Therefore, the number of motions is found, and the motion based segmentation is complete. Afterwards, the motion of each cluster can be calculated by using all the point correspondences in this cluster with the Umeyama's method [21]. If a cluster has less than three members, this cluster will be neglected and the members in this cluster will not be considered.

4.1.2 Experiment

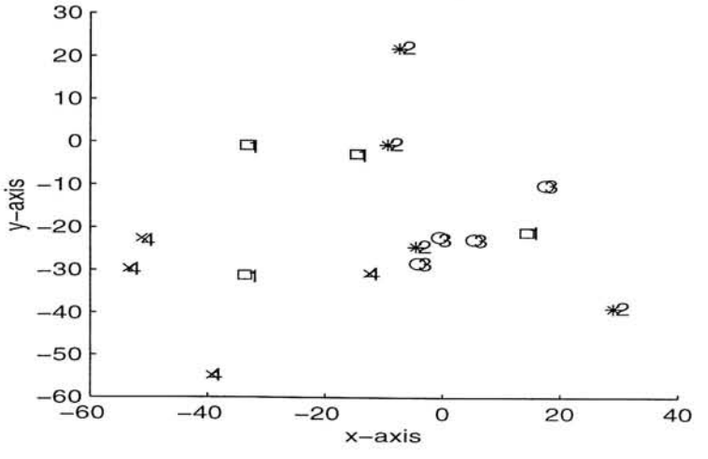
In the experiments, we generate a number of synthetic data with noise to verify our algorithm. All the experiments are conducted on a SUN ULTRA 1/170 machine and all the programs are coded in Matlab 5.0.



(a) Distributions of the transformations. "o"=free of noise; "+"=corrupted by noise.



(b) Distributions of the rigid objects. Points with same marker belong to the same rigid object.



(c) Result of segmentation. Points indicated by the same number belongs to the same cluster.

Figure 21: The experimental results of the algorithm for the motion-based segmentation with 3D-to-3D point correspondences.

% error of the noise added to the system	2.21%
% error of the rotation angles	1.17%
% error of the translational parameters	2.38%
% error of difference between the estimated 3D points and the true 3D points	1.26%

Table 53: The statistical result.

In this example, 16 feature points (in 3D space) are generated randomly from an uniform distribution within a cube of size $120 \times 120 \times 120$ with the center at (0,0,0). Then we randomly partition all the feature points into 4 segments and each segment has 4 feature points as its members. Afterwards, the motion is generated randomly for each segment. The rotation is generated randomly from an uniform distribution, such that each of the three rotation angles about the three axes (x, y and z axes) are varying between $-\pi$ and π, while the translation vector is generated randomly from an uniform distribution in the range [-50,...,+50]. Then the new coordinates of the feature points after the transformations are computed. In order to simulate a noisy environment, Gaussian distributed noise with zero mean and 0.5 standard deviation is added to corrupt the coordinates of the 3D feature points after the transformations. Also, for each feature points, 6 nearest neighbours will be used to evaluate its optimal motion (i.e. $s = 6$).

Figure (21 a) shows the distributions of the feature points after the transformations (free of noise) and that of the feature points corrupted by noise. Figure (21 b) shows the segmentation of the feature points. Figure (21 c) shows the result of the motion-based segmentation. This figure shows that four clusters are found and there are four members in each cluster. Also, it is shown that the result of the segmentation in this example is correct.

Also, Table (53) shows the statistical results in this example. It indicates

the percentage error of the noise added to the system (i.e. the noise added to corrupt the new coordinates of the feature points after the transformations), and the percentage error of the difference between the true 3D points after the transformations (free of noise) and the estimated 3D points after the whole process. The percentage error of the motion is also be shown.

4.2 A new Approach with 2D-to-3D Point Correspondences

In this section, a novel algorithm is developed to solve the multiple rigid motions from a monocular perspective image sequence. First, an image sequence is taken by using a pin-hole camera. This image sequence is containing the image of some 3D objects with known 3D structures. Then this new algorithm will estimate the number of motions in the system and perform motion-based segmentation automatically. Afterwards, the 3D motions of all the moving rigid objects are computed based on the segmentation result. In this section, points are selected to be the 3D features on the 3D objects (models), and the 2D images of these 3D objects are also used to calculate the positions and orientations. Therefore, this proposed algorithm is used to solve the motion-based segmentation problem with 2D-to-3D point correspondences. Also, for simplicity, it is assumed that the feature selection and tracking have been performed prior to the execution of our algorithm. Actually, this approach is similar to the approach presented in Section 4.1. They are just designed for two different kinds of problems. The approach described in Section 4.1 is used to handle motion-based segmentation with 3D-to-3D point correspondences. While the approach described here is used to deal with the problem with 2D-to-3D point correspondences.

4.2.1 Algorithm

Supposing that we are given a set $\mathbf{p}_i^{k-1}$ of 3D coordinates of the 3D model feature points on the objects at time $k-1$, and the set of the corresponding 2D

projection (image points) $\mathbf{q}_i^k$ of those feature points at time k is. In which $\mathbf{p}_i^k = [X_i^k\,,\, Y_i^k\,,\, Z_i^k]^T$ is the 3D coordinates of the $i-th$ feature point and $\mathbf{q}_i^k = [x_i^k\,,\, y_i^k]^T$ is the 2D image coordinates of the $i-th$ feature at time k, for $i = 1\,,\, 2, ..., N$ where N is the total number of feature points used. When several feature points are belonging to the same rigid body, they will undergo the same motion. If the $i-th$ feature point is belonging to the $j-th$ rigid body, the following equation can be obtained:

$$\mathbf{p}_i^k = \mathbf{R}_j^k \mathbf{p}_i^{k-1} + \mathbf{t}_j^k, \tag{4.90}$$

where $\mathbf{R}_j^k$ and $\mathbf{t}_j^k$ are the rotation matrix and the translation matrix respectively, which represent the motion of the $i-th$ feature point between time $k-1$ and k. By the definition of full perspective projection, the following equations can be obtained:

$$x_i^k = f\frac{X_i^k}{Z_i^k} \quad \text{and} \quad y_i^k = f\frac{Y_i^k}{Z_i^k}, \tag{4.91}$$

where f is the focal length of the camera.

In order to determine the motion of the feature points, the feature points of the same motion should be grouped together first. So, we should group points with the same motion into a cluster. However, the number of clusters is an unknown. Therefore, the following problems (The same problems mentioned in Section 4.1.1) are required to be solved:

1. Find the number of motion clusters, each cluster should contain feature points with same motion;

2. Segment all the feature points into a number of clusters;

3. Compute the motion of each cluster.

As mentioned above, the number of clusters and the prior knowledge of which feature points belong to which motion cluster are not given, so advanced clustering technique is required. As the same result, incremental clustering is also chosen in this approach.

After the process of motion segmentation, the motion of individual motion cluster will be calculated by any algorithms solving the 2D-to-3D point correspondences problem as described in[3][11][76]. In this thesis, the Six-point algorithm is used. Our proposed algorithm is as follows:

Firstly, for the $i-th$ feature point at time k, where $i = 1, ..., N$, we select s ($s \geq 5$) image points which are the nearest to the image point of the $i-th$ feature point at that instant. For each combination of 6 out of the $s+1$ feature points (including the $i-th$ feature itself and its s nearest neighbours selected), the motion is computed by using the Six-point algorithm. Since the motion of the $i-th$ feature point is considered, the $i-th$ feature point must be chosen in each motion estimation. Therefore, there are totally ${}_sC_5$ different motions as the candidates for the $i-th$ feature point. Since, the $i-th$ feature point can only have one motion at an instant, so only one motion is selected from the ${}_sC_5$ different motions. The chosen motion should minimize the following cost function:

$$F(\mathbf{R}, \mathbf{t}) = \left\| \mathbf{q}_i^k - \mathbf{h} \right\|^2, \tag{4.92}$$

where $\mathbf{R}$ and $\mathbf{t}$ represent one of the ${}_sC_5$ estimated motions for the $i-th$ feature point, and $\mathbf{h} = [h_x\,,\ h_y]^T$ in which,

$$h_x = f\frac{X_h}{Z_h} \ \text{ and } \ h_y = f\frac{Y_h}{Z_h}, \tag{4.93}$$

where

$$[X_h\,,\ Y_h\,,\ Z_h]^T = \mathbf{R}\boldsymbol{p}_i^{k-1} + \mathbf{t}, \tag{4.94}$$

Then the optimal rotation and translation of the i-th feature point are respectively $\mathbf{R}^i$ and $\mathbf{t}^i$.

After all the feature points have their own optimal motions, the incremental clustering technique is used with these motion parameters to segment the feature points. However, the rotation matrices cannot be used directly. The reason is discussed in Section 4.1.1. The 3 rotation angles about the three axes (x, y and z axes) are extracted from the rotation matrix. The rotation angles of the $i-th$ feature point about the x, y and z axes are respectively γ^i, β^i and α^i. Their calculations are given in Equations (4.81)-(4.84).

As a result, the clustering process can be performed for a sample in the six-dimensional motion parameter space (3 for the rotation angles and 3 for the translational parameters).

The motion of the $i-th$ feature point is described by the following vector:

$$\varpi_i = [\gamma^i\,,\ \beta^i\,,\ \alpha^i\,,\ t_x^i\,,\ t_y^i\,,\ t_z^i]^T, \tag{4.95}$$

where $[t_x^i\,,\ t_y^i\,,\ t_z^i]^T = \mathbf{t}^i$.

However, there may be a very large difference between the magnitudes of the rotation angles and that of translation parameters. Therefore, the three rotation angles cannot be used directly in clustering, and the 6D motion vectors should be normalized. The motion vectors are normalized in the following way:

$$\mathbf{v}_i = [\omega\gamma^i\,,\ \omega\beta^i\,,\ \omega\alpha^i\,,\ t_x^i\,,\ t_y^i\,,\ t_z^i]^T, \tag{4.96}$$

where

$$\omega = \frac{\sqrt{\frac{1}{N}\sum_{l=1}^{N}\left((t_x^l)^2 + \left(t_y^l\right)^2 + (t_z^l)^2\right)}}{\sqrt{\frac{1}{N}\sum_{l=1}^{N}\left((\alpha^l)^2 + (\beta^l)^2 + (\gamma^l)^2\right)}}. \tag{4.97}$$

From the experiment, we found that the normalization of the motion vectors

gives a more reliable and accurate result. Then the normalized 6D motion vectors are used in the incremental clustering procedure as described in Section 4.1.1.

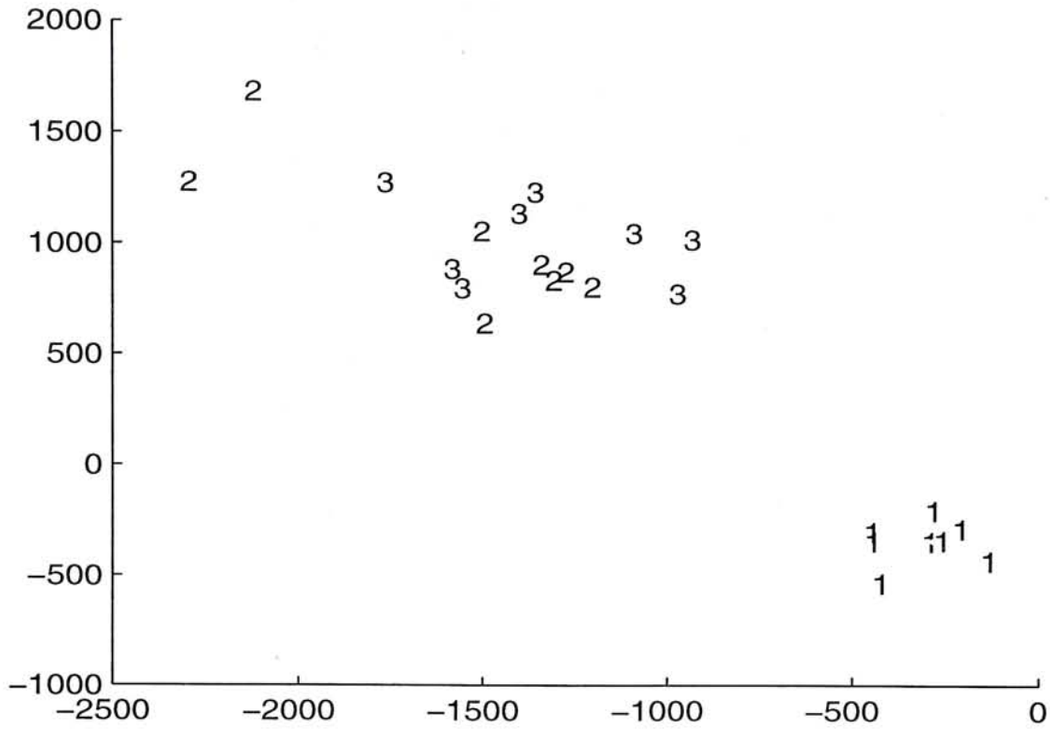
As a result, the number of motions is found, and the motion based segmentation is completed. Afterwards, the motion of each cluster can be calculated by using all the point correspondences in this cluster with the Six-point algorithm. If a cluster has less than six members, this cluster will be neglected and the members in this cluster will not be considered since the 3D rigid motion with less than 6 feature points cannot be determined uniquely [25].

4.2.2 Experiment

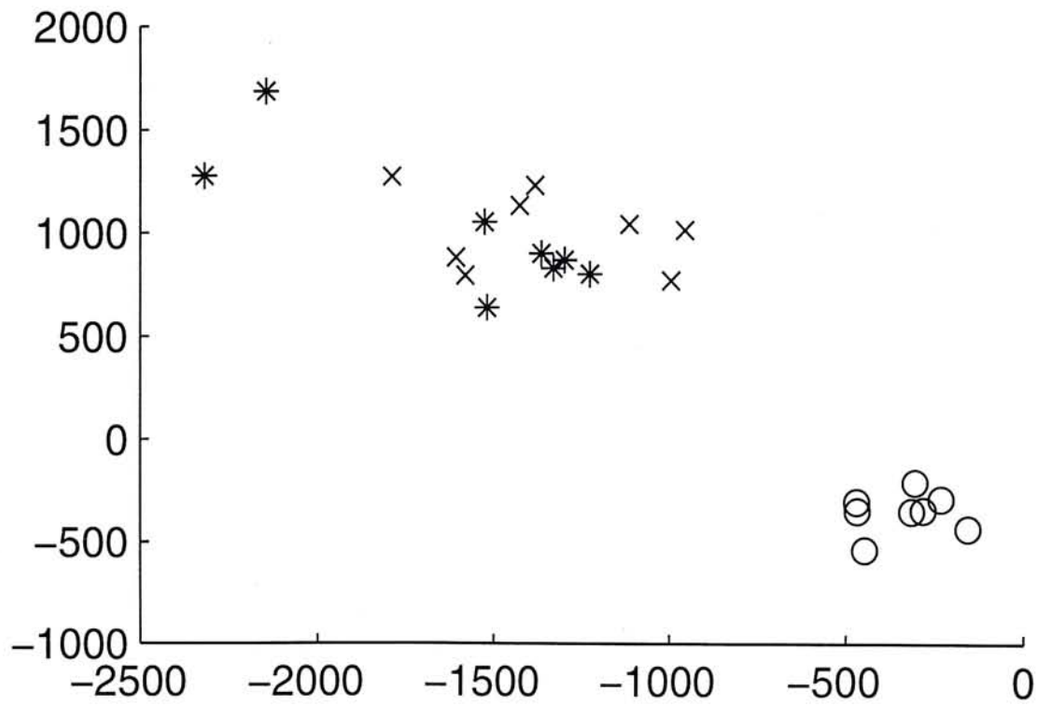
In this section, both synthetic data and real data are used to verify our algorithm. All the experiments are conducted on a SUN ULTRA 1/170 machine and all the programs are coded in Matlab 5.0.

4.2.2.1 Experiment using synthetic data

In the experiment using synthetic data, 24 feature points (in 3D space) are generated randomly. Then we randomly partition all the feature points into 3 segments and each segment has 8 feature points as its members. Afterwards, the motion is generated randomly for each segment. Then the new coordinates of the feature points after the motion are computed. An example is shown in Figure (22). In Figure (22 a), it is observed that some of the image points of the second rigid object (marked by "2") appear between some of the image points of the third rigid object (marked by "3"). It is similar to the case that some of the objects in the scene are transparent. However, Figure (22 b) shows that all the 3 sets of image points can be segmented correctly by using our algorithm.

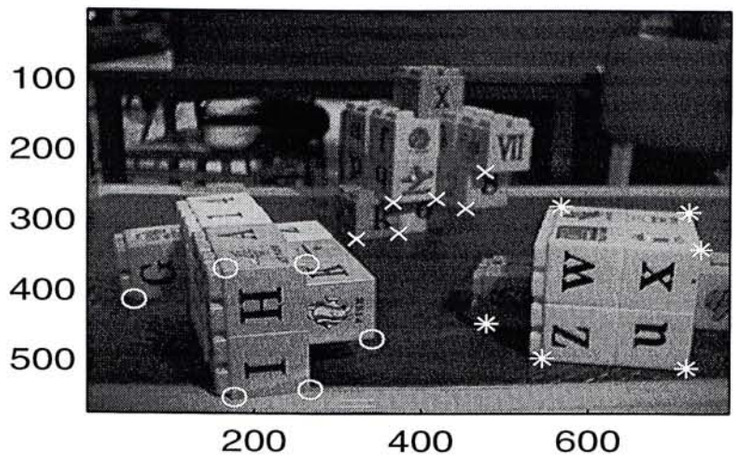


(a) The synthetic image. Image points with same number are belonging to the same rigid object.

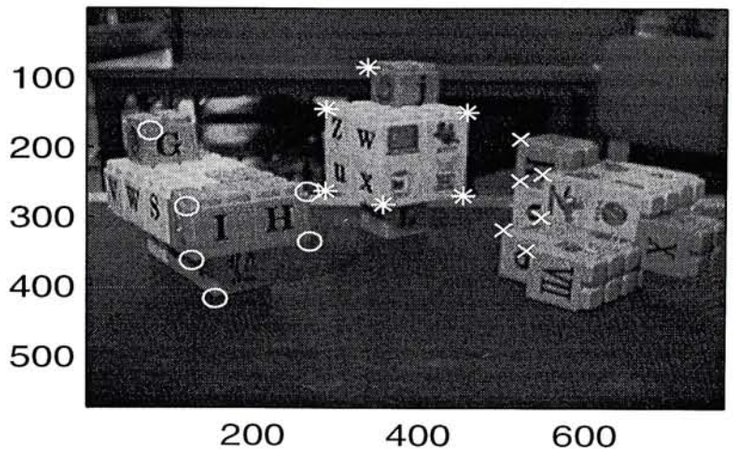


(b) The estimated result after motion segmentation and motion estimation of each individual rigid object. Image points with same symbol belong to the same rigid object.

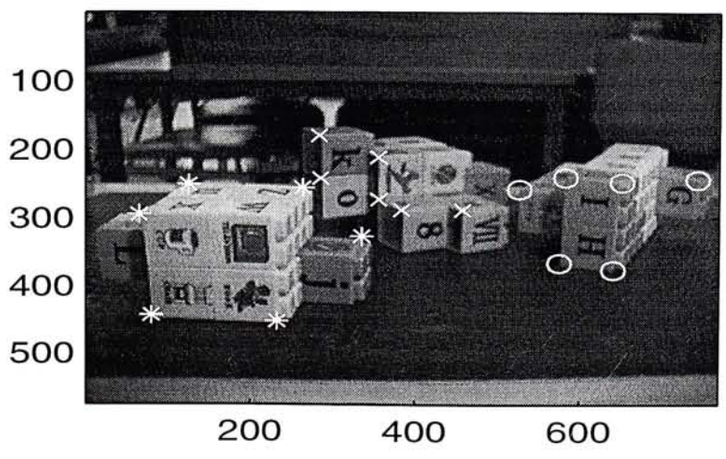
Figure 22: The experimental results using synthetic data



(a) Image at time k. Image points with same symbol belong to the same rigid object.



(b) The result of segmentation and motion estimation of image at time $k+1$. Points with same symbol belong to same object.



(c) The result of segmentation and motion estimation of image at time $k+2$. Points with same symbol belong to same object.

Figure 23: The experimental result using a real image sequence

4.2.2.2 Experiment using real image sequence

In this experiment, a real image sequence is captured. Figure (23) shows 3 images from the sequence at time k, $k+1$ and $k+2$ respectively. It is shown that there are three individual rigid objects undergo their own motions inside these 3 images. By using our proposed algorithm, the 3D coordinates of the feature points at time k are used to perform the segmentation and motion estimation at time $k+1$. Then the estimated 3D coordinates of the feature points at time $k+1$ are used in the calculation at time $k+2$. From Figure (23), it is observed that all the three individual objects can be segmented.

4.3 Summary

In this chapter, we have proposed two new approaches to solve the problem of motion-based segmentation in model-based computer vision. The first one described in Section 4.1 is for the case with 3D-to-3D point correspondences. Another one described in Section 4.2 is for the case with 2D-to-3D point correspondences. These two new approaches are similar. They both use incremental clustering technique to segment the feature points according to their motions.

The main difference between these two approaches is that two different motion estimation algorithms are used to calculate the motions of the objects. Since the first new approach is used to handle the case with 3D-to-3D point correspondences, it is based on the famous motion estimation technique proposed by Arun [15]. On the other hand, the second new approach is based on the Six-point algorithm in order to deal with the motion estimation in the case with 2D-to-3D point correspondences.

Also, experiments using synthetic data have been done to verify our algorithms. For the second approach, an experiment using a real image sequence has

also been performed. From the results, we find that the performance of this new approach in terms of accuracy is satisfactory.

CHAPTER 5

3D OBJECT RECOGNITION

In this chapter, a new model-based object recognition approach is proposed. In this approach, the geometric information of 3D objects is stored in a database. This kind of geometric information represents the shapes (relative 3D positions of the selected feature points) of the objects in the database. Also, all the objects must be rigid bodies. By using this approach, an object in a 2D intensity image is recognized from these 3D objects in the database directly. Therefore, the process of 3D reconstruction is not required. Besides, the position and the orientation of that recognized object can also be obtained. The most similar approach is proposed by Lowe [10][11]. In his approach, the 3D objects are also recognized from a single 2D intensity image, and the pose is also calculated. The main difference between his approach and ours is that we are using a direct method as the core of the motion estimation instead of iterative method in that of Lowe. In our approach, the linear algorithm, the Six-point algorithm, is applied.

The details of our proposed approach will be described in Section 5.1. This new algorithm is used to implement two recognition systems. The description of these two recognition systems is presented in Section 5.2. Also, experiments have been performed to verify our algorithm and it will be presented in Section 5.3. Finally, a summary is given in Section 5.4.

5.1 Proposed Algorithm for the 3D Object Recognition

A new algorithm is proposed to recognize an object from a database containing a number of 3D objects using only one single intensity image. In this section, only rigid objects are allowed. Therefore, the shape of the object must be kept unchanged (i.e. no deformation is allowed). The model database only contains the relative positions of the vertices of the 3D objects. Then the vertices of each model can represent the shape of that object.

Our proposed algorithm can be divided into two steps: 1) Assume the object in the intensity image is identical to one of the models in the database. With this assumption, the 3D position and orientation of the object in the image can be obtained. Then the estimated projection of that object in the estimated position and orientation is computed. The estimated projection of the feature points is compared to the real projection in the intensity image. 2) Verify this hypothesis. After applying the hypothesis step to all models in the database, the closest model to that in the image can be recognized by comparing the differences of the projection of each model in the database.

5.1.1 Hypothesis step

Firstly, an intensity image containing an object to be recognized is captured from an unknown viewpoint. Then the vertices of the projection of this object are extracted.

Secondly, we select a model from the database, and it is assumed to be identical to the object in the intensity image (Actually, the object is selected sequentially from the database). Then we can calculate the 3D position and orientation of the object in the image.

Assuming that the $m-th$ model is selected from the database. There are k

vertices in the projection of the object in the image are extracted while the selected model from the database has N vertices. Also, $\mathbf{p}_{mi} = [X_i, Y_i, Z_i]^T$ represents the 3D position of the $i-th$ vertex of the selected model from the database, for $i = 1, ..., N$. $\mathbf{q}_j = [x_j, y_j]^T$ represents the 2D image point (projection) of the $j-th$ vertex of the object in the intensity image, for $j = 1, ..., k$.

Then, the new 3D pose of the object in the image is calculated by using the vertices of the object from the image and that of the selected model from the database. However, we have no prior knowledge about the correspondences between the vertices in the selected model from the database and those in the object from the image. So, all the combinations (i.e. the total no. of combination is $(\frac{N!}{(N-k)!})$) should be tested. Without loss of generality, it can be assumed that the following combination is being considered:

$\mathbf{p}_{mi}$ and $\mathbf{q}_i$ is assumed to be a pair of point correspondence, for $i = 1, ..., k$.

By assuming that $\mathbf{p}_i^n = [X_i^n, Y_i^n, Z_i^n]^T$ be the estimated 3D position of the $i-th$ vertex of the object in the image. Then the following relation can be obtained:

$$\mathbf{p}_i^n = \mathbf{R}_m \mathbf{p}_{mi} + \mathbf{t}_m, \tag{5.98}$$

where $\mathbf{R}_m$ is the rotation matrix, which is a 3×3 orthogonal matrix. And $\mathbf{t}_m$ is the translation matrix, which is a 3×1 matrix.

If the hypothesis that the $m-th$ model is the identical object as that in the image, and the assumption that the correspondences are correct, there must be a unique rigid motion that can describe the motions of vertices of the selected model to the 3D positions of the corresponding vertices. Therefore, the Equation (5.98) is held.

In order to find out the motion parameters of the object, the Six-point algorithm is used. This algorithm is used because it can provide the answer of the

motion parameters in a very *short execution time*. It benefits the system containing a huge number of models in the database. Another advantage is that *no initial guesses are required*, so the system will be much more robust.

After calculating the motion between the selected model and the object in the image, the estimated projection of all the vertices can also be obtained. It is given in the following:

$$x_i^n = X_i^n \frac{f}{Z_i^n} \quad \text{and} \quad y_i^n = Y_i^n \frac{f}{Z_i^n}, \tag{5.99}$$

where $[x_i^n, y_i^n]^T$ is the estimated projection of the $i-th$ vertex of the selected model after the motion, and f is the focal length of the camera.

Then the error between the real projection of the vertices of the object in the image and the estimated projection from Equation (5.99) is calculated as follows:

$$E_m = \frac{1}{k} \sum_{i=1}^{k} \sqrt{(x_i - x_i^n)^2 + (y_i - y_i^n)^2} \tag{5.100}$$

After calculating all the errors of all the combinations for the selected $m-th$ model, the combination with the least error is chosen. The corresponding error is chosen to represent the error between the object in the image and the $m-th$ model, and it is assigned to be E_m^o. And the corresponding motion parameters ($\mathbf{R}_m$ and $\mathbf{t}_m$) are chosen to represent the motion between the object in the image and the $m-th$ model, and they are assigned to be $\mathbf{R}_m^o$ and $\mathbf{t}_m^o$.

5.1.2 Verification step

The hypothesis step in subsection 5.1.1 is repeated for all the models in the database. We can obtain the errors between all the models in the database and the object in the image.

Afterwards, the hypotheses made in the hypothesis step will be verified. Actually, all the errors will be compared and the model with the least error will

be chosen to be the correct model. Then this model will be recognized as identical as the object in the image.

Also, the motion parameters of the model with the smallest error are chosen to be the result of the motion between the object in the image and the correct model in the database.

5.2 3D Object Recognition System

In this thesis, the proposed approach for 3D object recognition is implemented into two systems. The first one has been coded in Matlab 5.0 on an ULTRA-SPARC machine. This system is used as the prototype. Another is coded in Visual C++ 5.0 on a PENTIUM II 233 machine, which is implemented with the help of two final year students who have been mentioned in the Acknowledgment. This two systems are specialised to deal with the recognition of boxes (with 6 faces). The use of boxes is to simplify both the implementation of the program and the database. However, the proposed recognition algorithm can be used to handle other complex objects.

There is difference between these two systems in terms of their functionalities. The system in Matlab can automatically remove the background and retain the projection of the box in the image. Then this system can automatically detect the outer-boundary and the vertices of the box. Finally, this box can be recognised according to its dimension.

The system in Visual C++ cannot automatically remove the background and detect the vertices of the box. The vertices of the box should be entered to the system manually. However, this system can recognise the box not only by using the information of the dimension, but also by using the texture of the box. It is because the database stores not only the dimensions of different boxes, but also

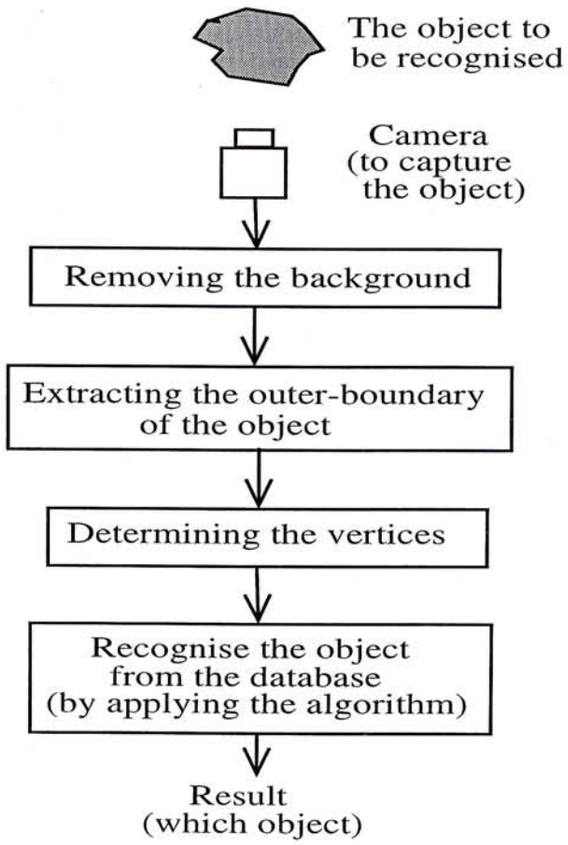


Figure 24: The outline of the 3D object recognition system in Matlab.

their 2D intensity images.

The detailed implementation of these two systems will be described in the following:

5.2.1 System in Matlab:

In this system, a number of boxes are stored in the database. Each box has its own dimension and no two boxes with the same dimension. The outline of this system is shown in Figure (24).

Firstly, the image of the object to be recognized is captured by a pin-hole camera. Then the background in that image is removed (It can be done by subtracting the image in the same viewpoint without the object). In Figure (25), an object is captured. After removing the background, the image is shown in Figure

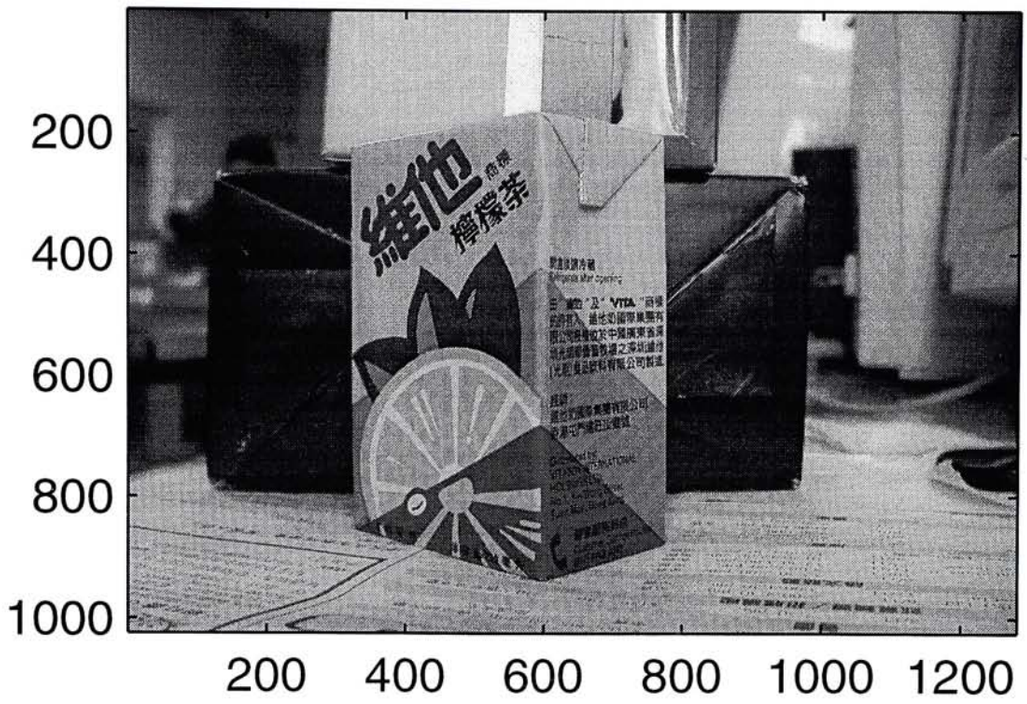


Figure 25: The captured image with the object (with background).

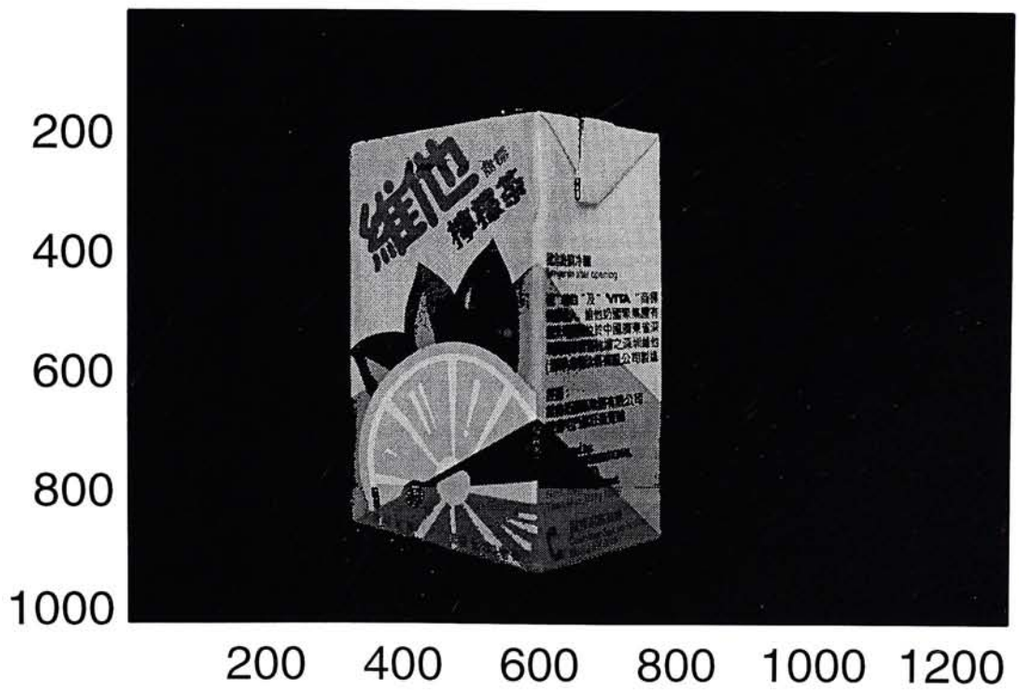


Figure 26: The image after removing background from Figure (25).

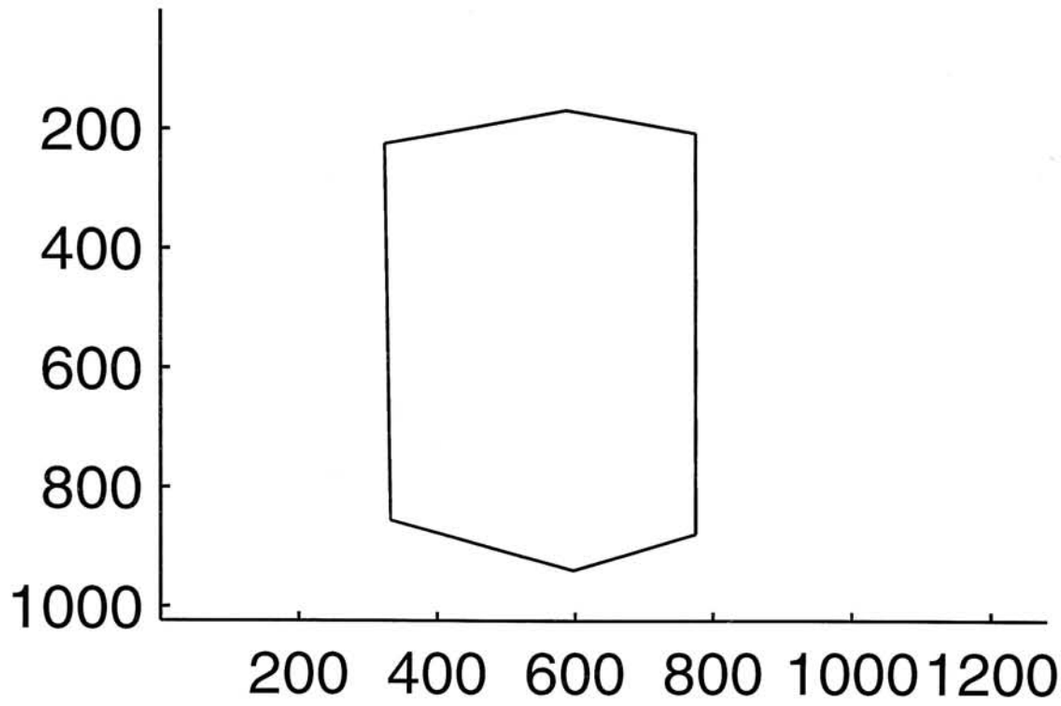


Figure 27: Outer-boundary of the projection of the object shown in Figure (26).

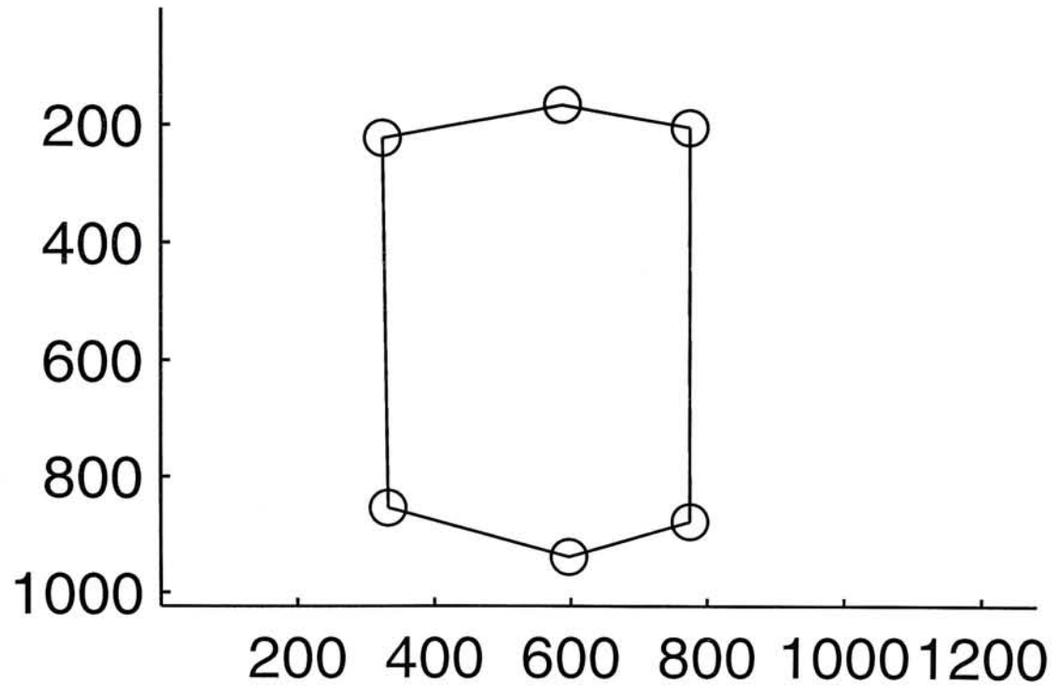


Figure 28: The vertices (indicated by the circles) of the objects in Figure (26).

(26). Then the outer-boundary of the projection of the object is extracted. Afterwards, six equations of straight lines are obtained to represent the outer-boundary of this box by using the line-case k-mean algorithm [75]. The result is shown in Figure (27). The vertices of the projection in the image are extracted by finding the intersecting points between each two adjacent lines. The result is shown in Figure (28). Finally, the recognition algorithm in Section 2 is applied to recognize the object from the database.

5.2.2 System in Visual C++

In this system, a number of boxes are stored in the database. In this case, several boxes may have the same dimension. In order to eliminate ambiguity, the intensity images of the boxes are also stored in the database. Therefore, the texture of the boxes will also be considered in the Verification Step. As a result, the database will contain both the geometric and texture information of the boxes. The outline of this system is shown in Figure (29).

Firstly, the image of the box to be recognised is captured by a pin-hole camera. Then the user should find out the projection of the vertices of this box and enter to the system. Afterwards, the system will start the process of recognition.

Figure (30) illustrates the process of the recognition in this system. It is unlike the process in the previous system. It contains two levels. The system in Level 1 will find out the candidates with the correct dimension. It is just the same as the system in Matlab. Moreover, the system in Level 2 will compare the texture of the box in the capture image and the selected candidates if there are more than one box with correct dimension. As a result, the ambiguity due to more than one box with the same dimension can be eliminated.

In Level 1, the system recognises the box according to its dimension. Also the positions (in 3D space) of the vertices of the box can be determined as well.

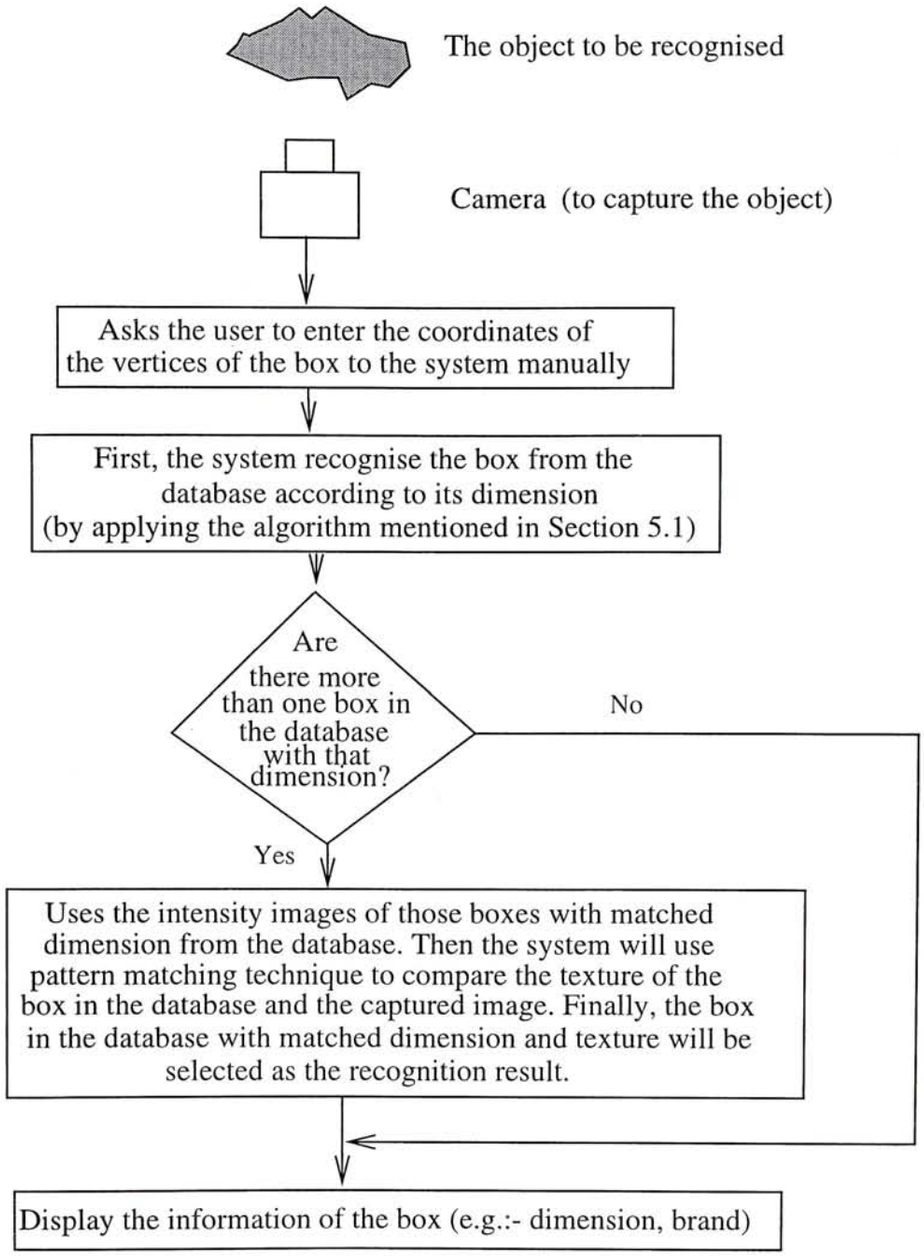


Figure 29: The outline of the recognition system in Visual C++.

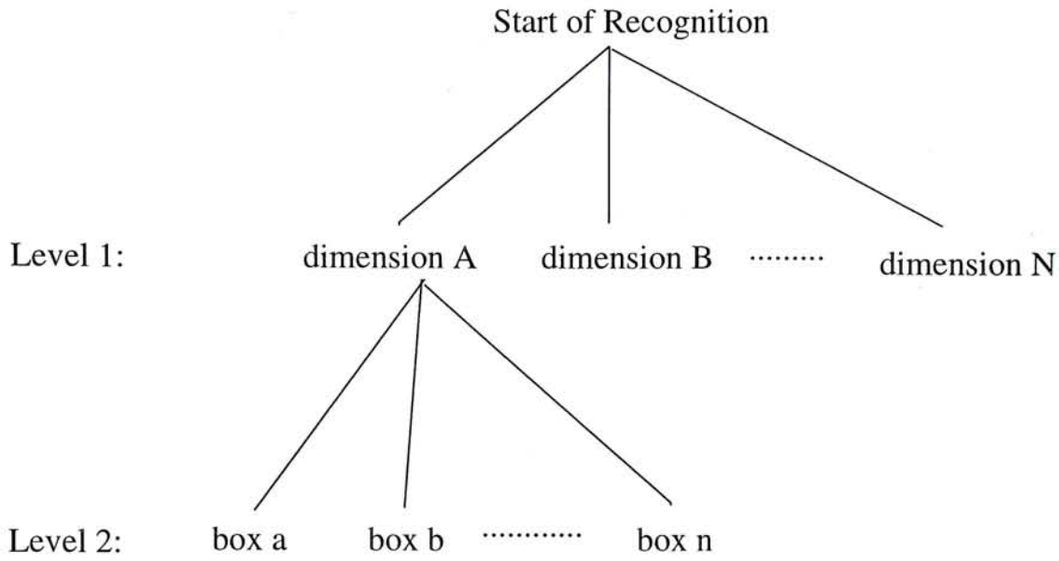


Figure 30: The process of the recognition with both information about the 3D structure and the texture of the objects. The system in Level 1 will recognise the box according to its 3D structure (dimension) and a number of candidates will be selected. Then the system in Level 2 will find out the best object from the selected candidates in Level 1 to be the answer by considering the texture information.

Then the system select one of the candidates to performs texture mapping [77]. Afterwards, the projection of this synthetic "3D box" is calculated. The correlation coefficient between this projection of the synthetic "3D box" and the true projection of the box to be recognized is calculated. This process is repeated for all the candidates to see which box in the database is best matched to the box captured by the camera.

There is a little difference between the Verification Step in this system and the original Verification Step described in Section 5.1.2. The information of textures of the objects should also be used in the Verification Step in this system.

5.3 Experiment

In this section, experiments were performed to verify our theory. Also, experiments were done by using the two recognition systems.

Boxes	Height/cm	Width/cm	Length/cm
250ml (in Figure (26))	10.5	6.5	4.0
375ml (in Figure (32))	13.5	6.5	4.5

Table 54: The table containing the dimensions of some of the boxes in the experiments.

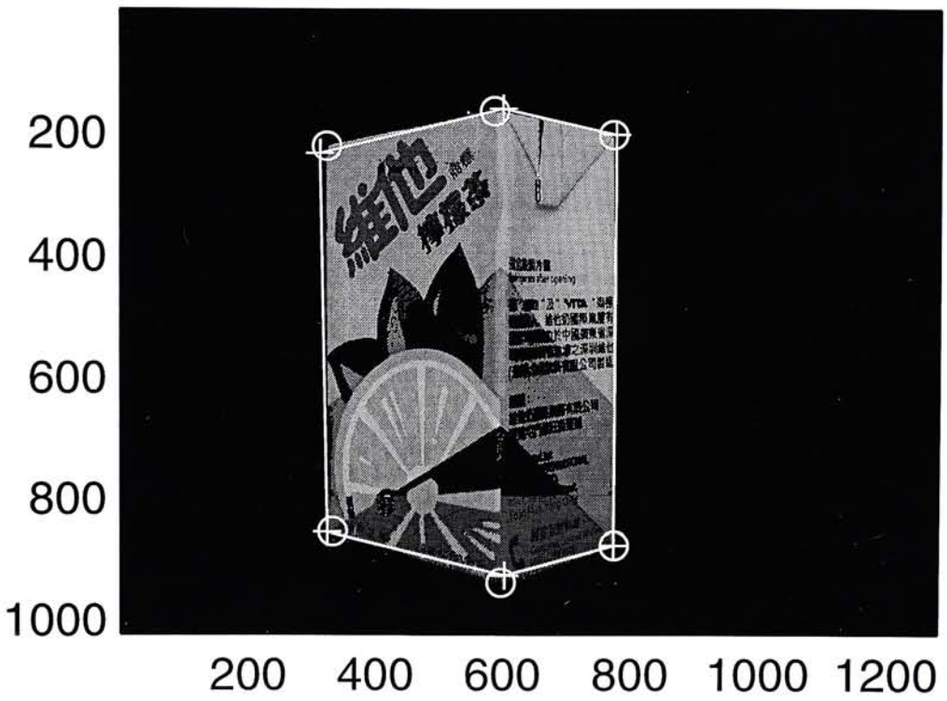


Figure 31: The recognition result using the image in Figure (25). The 'o' marks indicate the vertices of the true projection (captured) of the object while the '+' marks indicate the estimated projection of the recognised object after applying the estimated motion (estimated result). The white lines are the estimated outer-boundary of the object.

5.3.1 System in Matlab

In this experiment, 100 objects (boxes with different sizes) are in the model database.

The box shown in Figure (25) is a box of a common fruit-juice in Hong Kong. The volume of this box is 250ml and the dimension of this box is shown in Table

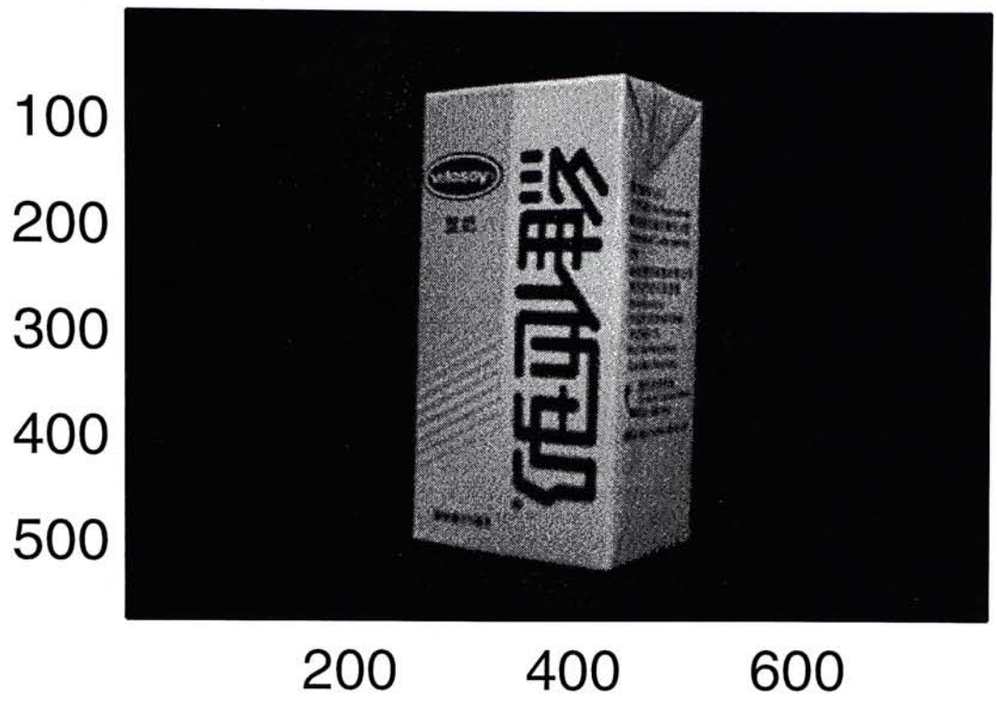


Figure 32: The image (after removing the background) of another box with volume 375ml.

(54). The recognition result of the object in Figure (25) is show in Figure (31). The object in the image is recognised to be the box with volume 250ml (i.e. the first box in Table (54)). From this example, it is not only shown that the result of the recognition of the this system is satisfactory, but it is also shown the result of the motion estimation of the object is satisfactory.

Also, in order to demonstrate the recognition system, experiment for recognising one more object is done in this paper. Figure (32) shows another box used in the experiment. The volume of this box is 375ml and the dimensions of this box is shown in Table (54) (the second box in the table). The outer-boundary of this box is shown in Figure (33) while Figure (34) shows the vertices of the outer-boundary of the object in Figure (32). Finally, the result of the recognition is shown in Figure (35). The object captured in Figure (32) was recognised to be the box with volume 375ml, so the recognition result was correct.

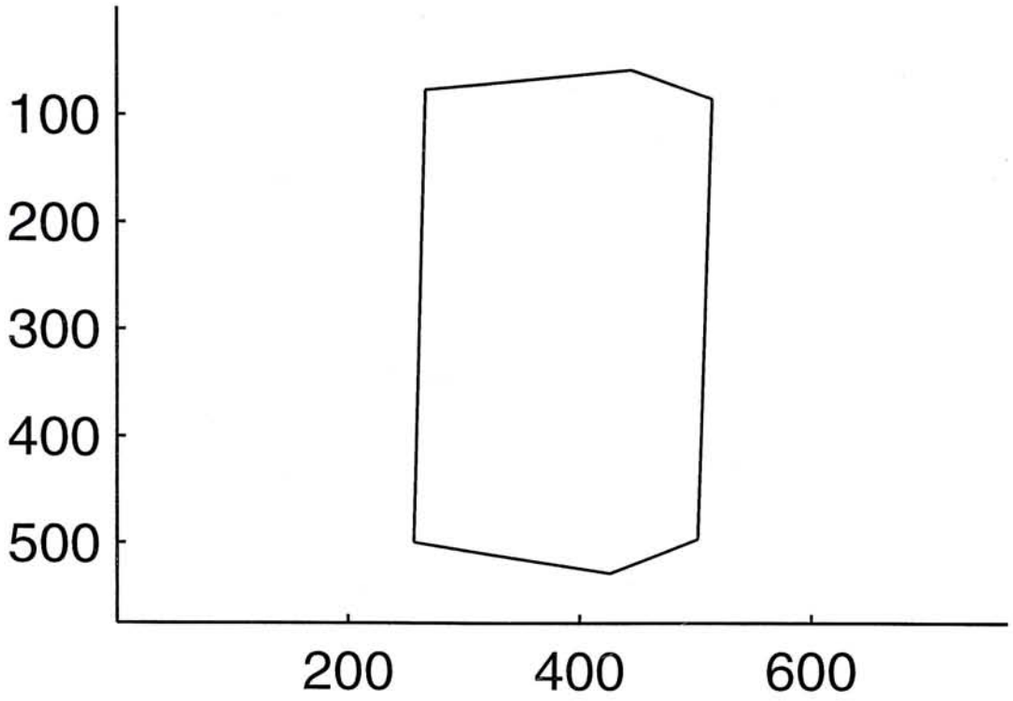


Figure 33: Outer-boundary of the projection of the object shown in Figure (32).

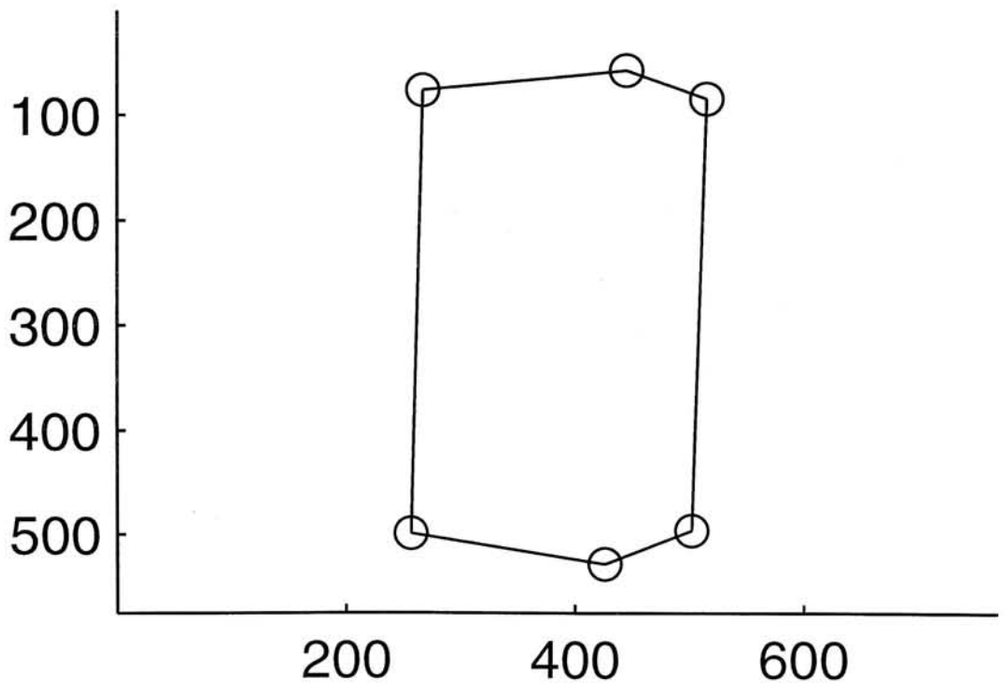


Figure 34: The vertices (indicated by the circles) of the objects in Figure (32).

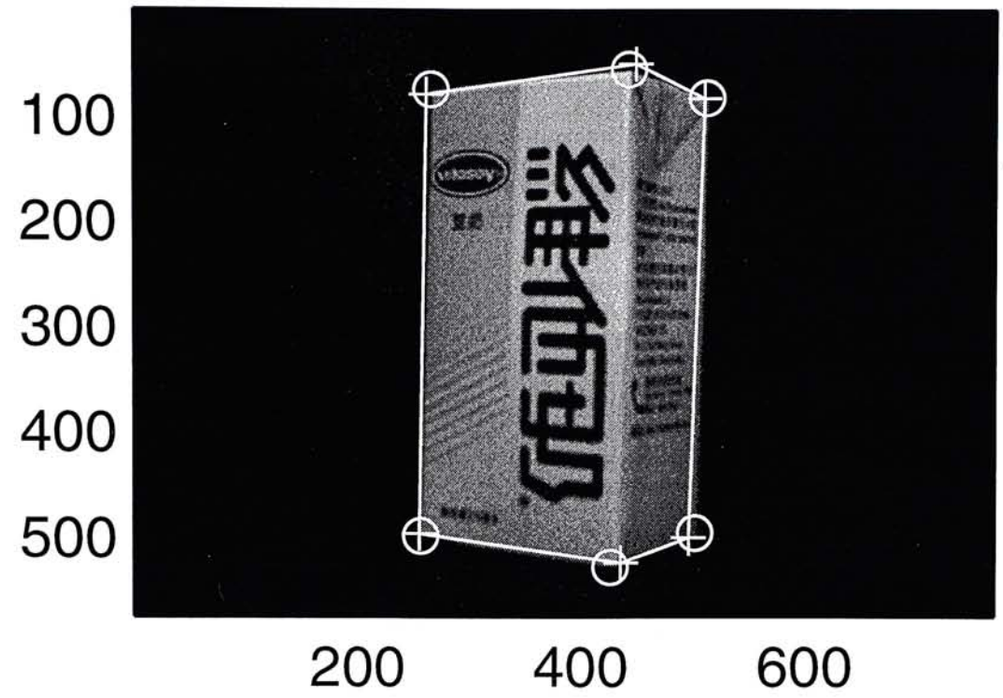


Figure 35: The recognition result using the image in Figure (32).

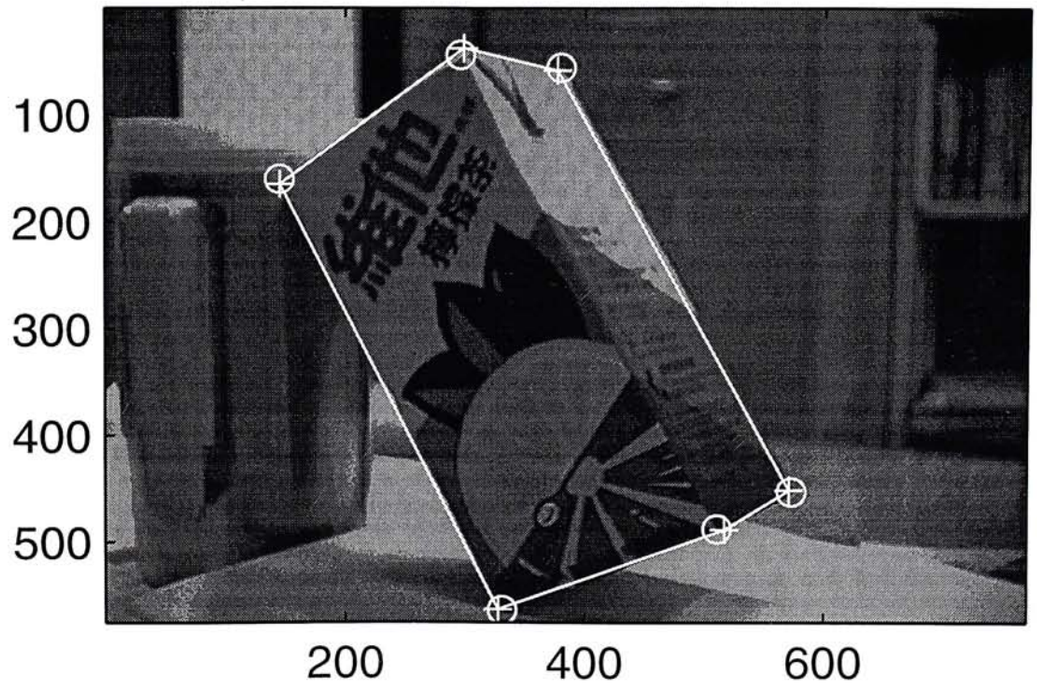


Figure 36: The recognition result of another box with volume 375ml with inclined orientation.

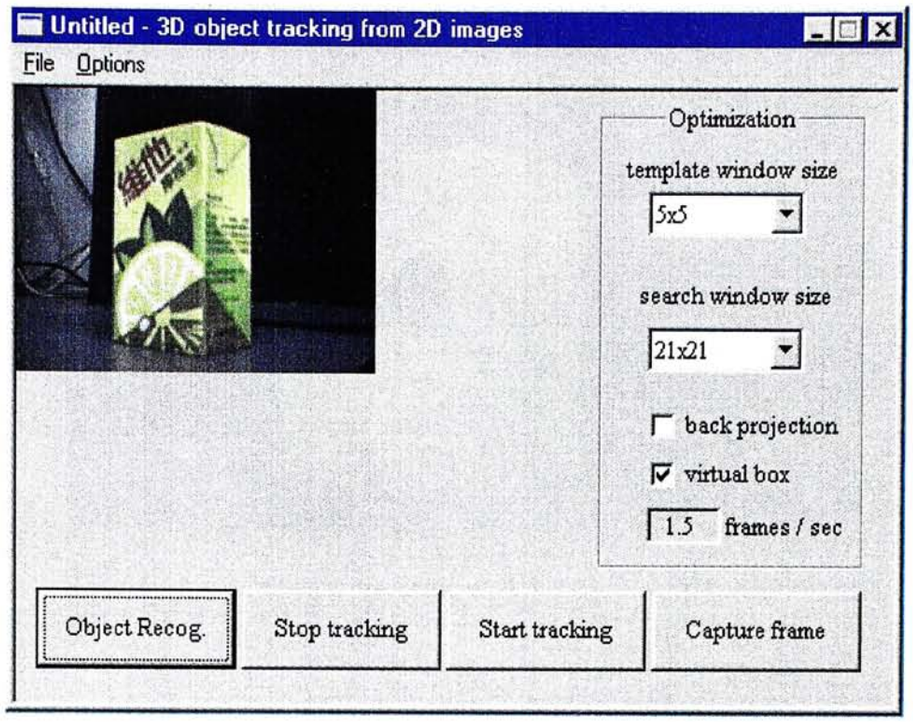


Figure 37: This figure illustrates the control panel of the 3D object recognition system in Visual C++. The box in the capture-window is a container of a common fruit-juice in Hong Kong called, "lemon-tea".

One more experiment using an inclined box (with volume 375ml) is shown in Figure (36). In this test, the inclined box was recognised to be the box with volume 375ml. Therefore, the recognition system and the proposed algorithm is verified to be workable to any orientation of the object.

The computation time for the part of recognition from the database using our proposed algorithm (i.e. the last step in the data-flow diagram in Figure (24)) was about 4 seconds, for the case that 100 objects are in the database and this system is only implemented in Matlab 5.0. Therefore, the execution speed of this algorithm by using the Six-Point Algorithm is satisfactory.

5.3.2 System in Visual C++

In this experiment, 6 boxes are used. Their geometric information and the 2D intensity images containing their textures are stored in the model database.

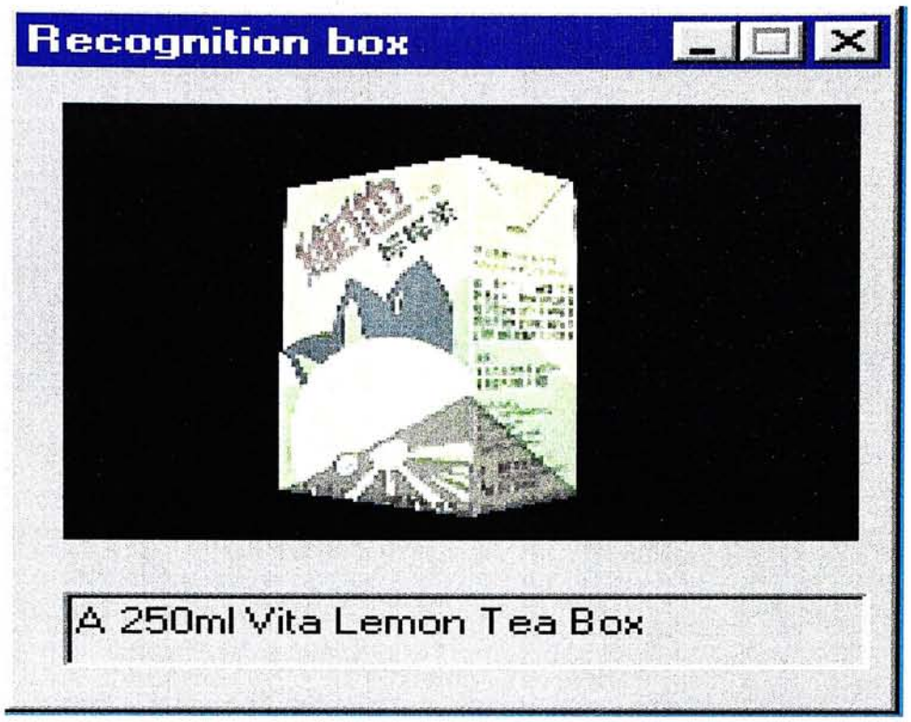


Figure 38: The recognition result of the box shown in Figure (37).

The box shown in Figure (37) is a box of a common fruit-juice in Hong Kong, which is called "lemon-tea". The volume of this box is 250ml and the dimensions of this box is shown in Table (54). The recognition result is shown in Figure (38). In this figure, the recognition system displayed the volume, the brand ("vita") and the type of product ("lemon-tea") on the window. Also, the system used the 2D intensity image of the box in the database to render the "3D box" displayed in this figure (Figure (38)).

Figure (39) shows there is another box captured. Its volume is 375ml. It is used to contain the "malted milk" produced by "Vitasoy". However, there is a hand in front of the box this time, so some area of the box is canceled by the hand. The recognition result is shown in Figure (40). From that figure, it is observed that the box can still be recognised as a container of "malted-milk" produced by "Vitasoy" and its volume is found to be 375ml. From this test, we find that the recognition result in this system can still be correct although some

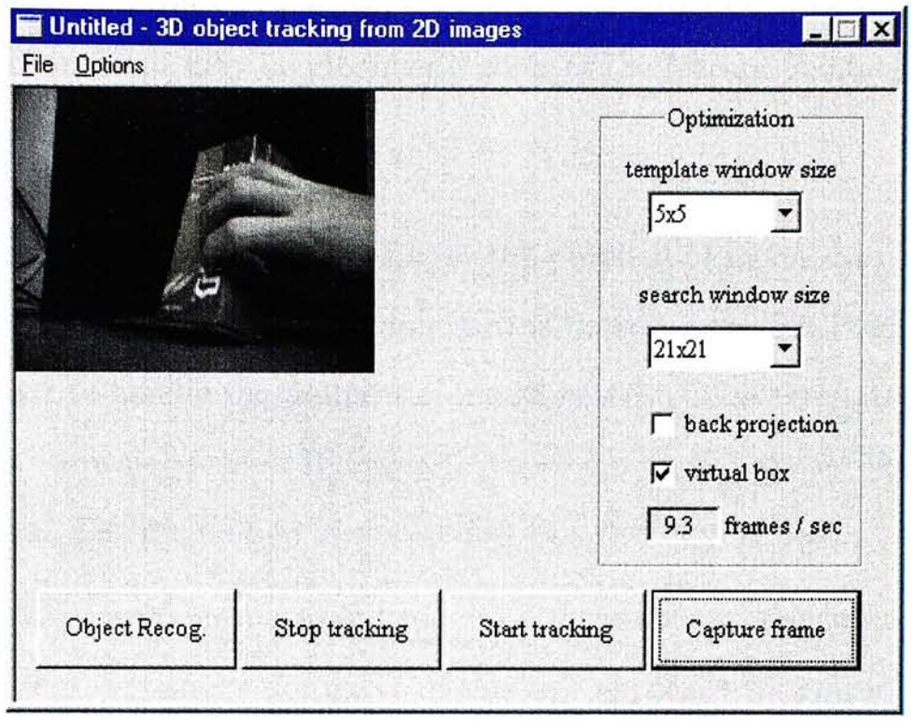


Figure 39: This figure shows another box (container of another fruit-juice). Its volume is 375ml and the name of the fruit-juice is "malted milk" which is produced by "Vitasoy" also. Moreover, a hand appeared in front of some parts of the box, so some area of the box is canceled.

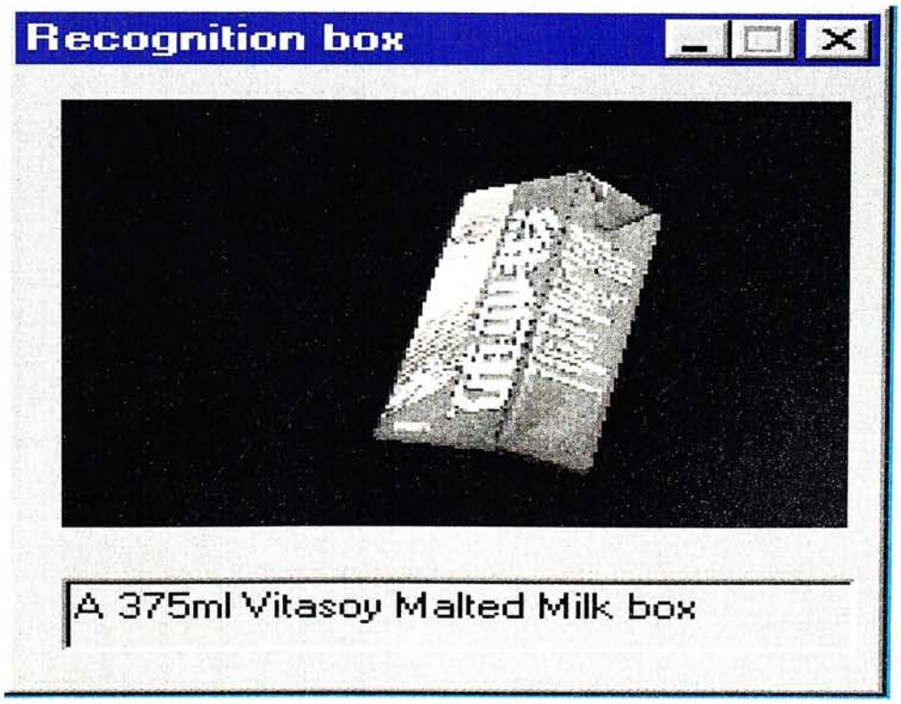


Figure 40: The recognition result of the object shown in Figure (39).

region (not including the feature points) of the target object is hidden (Of course, the recognition result may be incorrect if some of the feature points are hidden).

5.4 Summary

In this chapter, a new approach to the problem of model-based 3D object recognition is proposed. This new approach is based on the Six-Point algorithm, which is used to handle the problem of model-based motion estimation with 2D-to-3D point correspondences. By using this new approach, not only the object can be recognised, but the motion of that object can also be revealed.

Moreover, experiments have been done to verify our algorithm. From the results, we find that the performance of this new approach in term of accuracy of the recognition and motion estimation is satisfactory. On the other hand, two 3D object recognition systems are implemented by using the proposed algorithm.

CHAPTER 6

CONCLUSIONS

In this thesis, three main areas in model-based computer vision are discussed. They are:

1. Model-based motion analysis;
2. Motion-based segmentation;
3. 3D object recognition.

In this thesis, all these problems are relating to point-based correspondences, hence, other types of features (e.g. straight-lines, surface) have not been discussed.

Two approaches have been proposed for model-based motion analysis with 2D-to-3D point correspondences. The first one is an iterative algorithm for the P4P problem which is called the TL-algorithm. Also, a comparison between the TL-algorithm and Lowe's algorithm [10] (another famous iterative method) has been performed. From the result, it is observed that the TL-algorithm is better than Lowe's algorithm in terms of speed. The advantage of the TL-algorithm is more obvious when the motion of the object is pure rotation.

Another approach is a two-step linear algorithm called the Six-point algorithm. It can be applicable to any PnP problems for $n \geq 6$ and for the case that not all the feature points are on the same plane. There are two versions for the

Six-point algorithm. Both versions can be divided into two parts: 1.) The linear part; 2.) The constraint satisfactory part. The two versions of the Six-point algorithm are only different in applying different constraints in the step of constrained optimization in the linear part. In the linear part, a rough solution for the motion of the object can be obtained. In the constraint satisfactory part, the constraint of the orthogonality of the rotation matrix is applied to refine the result of the motion parameters. Also, the technique of SVD is proposed to enhance the speed of the computation. Besides, this Six-point algorithm is applied to implement a real-time visual tracking system.

Also, an enhancement to Faugeras's algorithm [12] is introduced. This modified Faugeras's algorithm improves the accuracy of the translational components as compared to the original version of Faugeras's algorithm. In this modified version a new method is proposed to estimate the translational part of the motion. Also, experiments have been done to verify the new approach and compare the performances of original Faugeras's algorithm and this modified version in terms of accuracy of the translational components. From the result, it is observed that the accuracy of this modified version is better than the original one especially when the noise level in the system is high.

Moreover, two new approaches have been proposed for the motion-based segmentation. The first approach is for the case with 3D-to-3D point correspondences. Another is for the case with 2D-to-3D point correspondences. These two approaches are similar. The incremental clustering technique is applied in both approaches to segment the feature points according to their motions. The major difference between them is that two different motion estimation algorithms are applied to compute the motions of the objects.

Finally, a new approach for the model-based 3D object recognition is presented. In this new algorithm, our Six-point algorithm is used as the core to compute the motions of the object to be recognised. Also, two 3D object recognition systems are implemented using this proposed algorithm. The first system is coded in Matlab and which can recognize the object automatically. On the other hand, another system is coded in Visual C++. In this system, the texture of the object is also used to enhance the accuracy of the recognition.

REFERENCES

[1] D. Vernon. *Machine Vision:Automated Visual Inspection and Robot Vision.* Prentice Hall, 1991.

[2] R.A. Brooks. *Mobel-based Computer Vision.* UMI Research Press, 1990.

[3] T. S. Huang and A. N. Netravali. Motion and structure from feature correspondences:- a review. *Proceeding of The IEEE*, 82(2):252–268, Feb 1994.

[4] S. Ullman. *The Interpretation of Visual Motion.* The MIT Press, 1979.

[5] G. Xu and Z. Zhang. *Epipolar Geometry in Stereo, Motion and Object Recognition: a unified approach.* Kluwer Academic Publishers, 1996.

[6] A. M. Tekalp. *Digital Video Processing.* Prentice Hall PTR, 1995.

[7] J. Weng, T. S. Huang, and N. Ahuja. *Motion and Structure from Image Sequences.* Springer-Verlag, 1993.

[8] R. Wilson and M. Spann. *Image Segmentation and Uncertainty.* Research Studies Press Ltd., 1988.

[9] D. Lowe. Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision*, 8(2):113–122, 1992.

[10] D.G. Lowe. Three-dimensional object recognition form single two-dimensional images. *Artificial Intellengence*, 31:355–395, 1987.

[11] D. G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Trans. on Pattern Analysis and Machine Intell.*, 13(5):441–450, May 1991.

[12] O. Faugeras. *Three-Dimensional Computer Vision.* MIT Press, 1993.

[13] H. Goldstein. *Classical Mechanics.* Reading, MA: Addison Wesley, 1981.

[14] T.S. Huang, S.D. Blostein, and E.A. Margerum. Least-squares estimatiopn of motion parameters from 3-d point correspondences. *Proc. IEEE Conf. Computer Vision and Pattern Recognition, Miami Beach, FL*, pages 24–26, June 1986.

[15] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intell.*, PAMI-9(5):698–700, Sep 1987.

[16] G. H. Golub and C. F. V. Loan. *Matrix Computations.* Baltimore, Md.: Johns Hopkins Univ. Press, 1989.

[17] B.K.P. Horn, H.M. Hilden, and S. Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *J. Opt. Soc. Am Ser A*, 5:1127–1135, 1988.

[18] O.D. Faugeras and M. Hebert. A 3-d recognition and positioning algorithm using geometrical matching between primitive surfaces. *Proc. Int. Joint Conf. Artificial Intelligence, Karlshrue, West Germany*, pages 996–1002, Aug. 1983.

[19] M.W. Walker, L. Shao, and R.A. Volz. Estimating 3-d location paramters using dual number quaternions. *CVGIP: Image Understanding*, 54:358–367,

1991.

[20] D. W. Eggert, A. Lorusso, and R. B. Fisher. Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Machine Vision and Applications*, 9(5-6):272–290, 1997.

[21] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. on Pattern Analysis and Machine Intell.*, 13(4):376–380, April 1991.

[22] D. P. Huttenlocker and S. Ullman. Recognizing solid objects by alignment. *presented at the DARPA Image Understanding Workshop, Cambridge MA*, pages 1114–1122, April 1988.

[23] T.D. Alter. 3d pose from 3 points using weak-perspective. *IEEE Trans. on Pattern Analysis and Machine Intell.*, 16(8):802–808, Aug. 1994.

[24] A. Gee and R. Cipolla. Fast visual tracking by temporal consensus. *Image and Vision Computing*, 14:105–114, 1996.

[25] M. A. Fischler and R.C. Bolles. Random sample consensus:- a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[26] W. Wolfe, D. Mathis, C. Sklair, and M. Magee. The perspective view of three points. *IEEE Transactions on Pattern Analysis and Machine Intell.*, 13(1):66–73, Jan. 1991.

[27] R. Horaud, O. Leboulleux B. Conio, and B. Lacolle. An analytic solution for the perspective 4-point problem. *Comput. Vis. Graphics, Image Process.*, 47(1):33–44, July 1989.

[28] M. A. Abidi and T. Chandra. A new efficient and direct solution for pose estimation using quadrangular targets: Algorithm and evaluation. *IEEE Transactions on Pattern Analysis and Machine Intell.*, 17(5):534–538, May 1995.

[29] Y. Hung, P. Yeh, and D. Harwood. Passive ranging to known planar point sets. *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1:80–85, March 1985.

[30] S. Ganapathy. Decomposition of transformation matrices for robot vision. *Pattern Recognition Letters*, 2:401–402, Dec. 1984.

[31] B.K.P. Horn and E.J. Weldon JR. Direct methods for recovering motion. *International Jpurnal of Computer Vision*, 2:51–76, 1988.

[32] L. Quan and Z. Lan. Linear n$\geq$4 -point pose determination. *ICCV'98*, Jan. 1998.

[33] W. Hoff M. Magee, L. Gatrell, M. Marietta, and W. Wolfe. Adaptive camera calibration in an industrial robotic environment. *IEA/AIE 90*, vol. I:242–251, 1990.

[34] T. Boult. Local image reconstruction and subpixel restoration algorithms. *CVGIP*, 55:63–77, Jan. 1993.

[35] D.P. McReynolds and D.G. Lowe. Rigidity checking of 3d point correspondences under perspective projection. *IEEE Trans. on Pattern Analysis and Machine Intell.*, 18(12):1174–1185, Dec. 1996.

[36] K.E. Athinson. *An Introduction to Numerical Analysis.* John Wiley and Sons, 2nd. edition, 1989.

[37] T.S. Huang and C.H. Lee. Motion and structure from orthographic projections. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11:536–540, May 1989.

[38] K. Aizawa, H. Harashima, and T. Saito. Model-based analysis-synthesis image coding (mbasic) system for a person's face. *Signal Processing: Image Communication*, 1:139–152, Oct 1989.

[39] G. Bozdagi, A.M. Tekalp, and L. Onural. An improvement to mbasic algorithm for 3-d motion and depth estimation. *IEEE Trans. on Image Processing (special issue)*, 3:711–716, June 1994.

[40] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, Sept. 1981.

[41] R. I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intell.*, 17(12):580–593, June 1997.

[42] G. Toscani and O. D. Faugeras. Structure and motion from two noisy perspective views. *Proceedings IEEE Int'l Conf. on Robotics and Automation, Raleigh, N. Carolina, USA*, pages 221–227, 1987.

[43] W. R. Hamilton. *Elements of Quaternions.* Chelsea Publishing Company, 3rd edition, 1969.

[44] J. Philip. Estimation of three-dimensional motion of rigid objects from noisy observations. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(1):61–66, Jan. 1991.

[45] S. Maybank. *Theory of Reconstruction from Image Motion.* Springer-Verlag, 1993.

[46] Z. Zhang. Motion and structure of four points from one motion of a stereo rig with unknown extrinsic parameters. *IEEE Transactions on Pattern Analysis and Machine Intell.*, 17(12):1222–1227, Dec 1995.

[47] Z. Zhang and O. D. Faugeras. Three-dimensional motion computation and object segmentation in a long sequence of stereo frames. *Int'l Journal of Comp. Vision*, 7(3):211–241, 1992.

[48] Y. P. Hung, C. Y. Tang, S. W. Shih, Z. Chen, and W. S. Lin. A 3d predictive visual tracker for tracking multiple moving objects with a stereo vision system. *Image Analysis Applications and Computer Graphics. Third Int'l Computer Science Conf., ICSC'95*, pages 25–32, 1996.

[49] W. Wang and James H. Duncan. Recovering the three-dimensional motion and structure of multiple moving objects from binocular image flows. *Computer Vision and Image Understanding*, 63(3):430–446, May 1996.

[50] W. Tao, Z. Peng, Z. Xinhua, and X. Xiaoliang. A robust algorithm for estimating multiple 3-d rigid motions. *Chinese Journal of Computers*, 17(4):290–297, April 1994.

[51] D. W. Jocobs and C. Chennubhotla. Segmenting independently moving, noisy points. *Peoceeding of 1994 IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, pages 96–103, Nov. 1994.

[52] G. Xu and S. Tsuji. Correspondence and segmentation of multiple rigid motions via epipolar geometry. *Proceedings of the 13th International Conference on Pattern Recognition*, 1:213–217, Aug. 1996.

[53] J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. *IEEE Trans. on Image Processing*, 3(5):625–638, Sept. 1994.

[54] M. J. Black and A. Jepson. Estimating multiple independent motions in segmented images using parametric models with local deformations. *Peoceeding of 1994 IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, pages 220–227, 1994.

[55] C. Dorai and A.K. Jain. Recognition of 3d free-form objects. *Proceedings of the 13th International Conference on Pattern Recognition, Vienna*, 1:697–701, Aug 1996.

[56] C. Schutz and H. Hugli. Free-form 3d object recognition. *ISPRS Third Conference on Optical 3-D Measurement Techniques, Vienna*, 1995.

[57] P. Gingins and H. Hugli. Model-based 3d object recognition by a hybrid hypothesis generation and verification approach. *Intelligent Robots and Computer Vision XIII: Algorithms and Computer Vision., SPIE 2353, Boston*, pages 67–77, Oct. 1994.

[58] T.M. Silberberg, D.A. Harwood, and L.S. Davis. Object recognition using oriented model points. *Computer Vision, Graphics and Image Processing*, 35:47–71, 1986.

[59] T. Poggio and S. Edelman. A network that learns to recognize three-dimensional objects. *Nature*, 343:263–266, Jan. 1990.

[60] Y. Kuno, Y. Okamoto, and S.Okada. Robot vision using a feature search strategy generated from a 3-d onject model. *IEEE Trans. on Pattern Analysis and Machine Intell.*, 13(10):1085–1097, Oct. 1991.

[61] D. Forsyth, J.L. Mundy, A. Zisserman, C. Coelho, A. Heller, and C. Rothwell. Invariant descriptors for 3-d object recognition and pose. *IEEE Trans. on Pattern Analysis and Machine Intell.*, 13(10):971–991, Oct. 1991.

[62] F.S. Cohen and J.Y. Wang. Part i: Modeling image curves using invariant 3-d object curve models-a path to 3-d recognition and shape estimation from image contours. *IEEE Trans. on Pattern Analysis and Machine Intell.*, 16(1):1–12, Jan. 1994.

[63] M.L. LIU and H.K. WONG. A real-time visual tracking system using a new linear algorithm for model-based pose estimation. *prepared for a journal publication*, 1998.

[64] M.L. Liu and K.H. Wong. Computer vision based real-time pose estimation using four corresponding points. *Proceedings of the Workshop on 3D Computer Vision 97*, pages 44–48, May 1997.

[65] M.L. LIU and H.K. WONG. Pose estimation using four corresponding points. *revised by Pattern Recognition Letters*, 1998.

[66] M.L. LIU and K.H. WONG. A novel algorithm for recovering the 3d motions of multiple moving rigid objects. *to appear in ICPR'98*, August 1998.

[67] M.L. LIU, H.K. WONG, W.K. LAM, and K.F. CHEUNG. Recovering the 3d motions of multiple moving rigid objects in an image sequence with clustering techniques. *to appear in NOLTA'98*, September 1998.

[68] M.L. LIU and H.K. WONG. Recovering the 3d motions of multiple moving rigid objects in an image sequence. *to appear in ICIPS'98*, August 1998.

[69] M.L. LIU and H.K. WONG. Model-based 3d objects recognition from an intensity image. *to appear 98 Workshop on Computer Vision*, 1998.

[70] S. Chen and J. Weng. Calibration for peripheral attenuation in intensity images. *Proc. 1st IEEE International Conference on Image Processing*, pages

992–996, Nov. 1994.

[71] J. Weng, P. Cohen, and M. Herniou. Camera calibration with distortion models and accuracy evaluation. *IEEE Trans. on Pattern Analysis and Machine Intell.*, 14(10):965–980, Oct. 1992.

[72] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, 4(4):629–642, Apirl 1987.

[73] R.C. Gonzalez and R. E. Woods. *Digital Image Processing.* Addison Wesley, 2nd. edition, 1993.

[74] D. Vernon. *Machine Vision: Automated Visual Inspection and Robot Vision.* Prentice Hall, 1991.

[75] W.K. Lam, H.Y. Leung, and L. Xu. Comparison of several clustering algorithms on real data. *1996 China Congress of Neural Networks (CCNN'96)*, 1996.

[76] S.H.SHOR and K.H.WONG. An efficient iterative pose estimation algorithm. *Computer Vision - ACCV'98, Proceedings*, II:559–566, Jan. 1998.

[77] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes. *Computer Graphics: Principles and Practice.* Addison-Wesley, 2nd. edition, 1990.

APPENDIX A

REPRESENTATION OF ROTATIONS BY QUATERNION

Supposing that we are going to obtain a 3×3 orthogonal rotation matrix $\mathbf{R}$ from a minimization problem with the following form:

$$\underset{\mathbf{R}}{\text{Min}} \|\mathbf{RC} - \mathbf{D}\| \quad \text{subject to } \mathbf{R} \text{ is a } 3 \times 3 \text{ orthogonal matrix,} \tag{1.101}$$

where $\mathbf{C} = \begin{bmatrix} \mathbf{C}_1 & \mathbf{C}_2 & \mathbf{C}_3 \end{bmatrix}$ and $\mathbf{D} = \begin{bmatrix} \mathbf{D}_1 & \mathbf{D}_2 & \mathbf{D}_3 \end{bmatrix}$ are some 3×3 matrices. The solution of Equation (1.101) is as follows:

Defining a 4×4 matrix $\mathbf{B}$ by

$$\mathbf{B} = \sum_{i=1}^{N} \mathbf{B}_i^T \mathbf{B}_i, \tag{1.102}$$

where

$$\mathbf{B}_i = \begin{bmatrix} 0 & (\mathbf{C}_i - \mathbf{D}_i)^T \\ \mathbf{D}_i - \mathbf{C}_i & [\mathbf{D}_i + \mathbf{C}_i]_\times \end{bmatrix}, \tag{1.103}$$

in which $[\circ]_\times$ is an operator. For $\mathbf{u} = \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix}^T$,

$$[\mathbf{u}]_\times = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix}. \tag{1.104}$$

Then the unit eigenvector of $\mathbf{B}$ corresponding to the smallest eigenvalue is assigned to be $\mathbf{q} = [q_0\,,\, q_1\,,\, q_2\,,\, q_3]^T$. The unit vector $\mathbf{q}$ is actually an unit quaternion.

Therefore the rotation matrix $\mathbf{R}$ can be computed as follows:

$$\mathbf{R} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_2 q_1 + q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_3 q_1 - q_0 q_2) & 2(q_3 q_2 + q_0 q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}. \quad (1.105)$$

APPENDIX B

CONSTRAINED OPTIMIZATION

Supposing we are going to minimize a linear system $\|\mathbf{Ax}\|$ subject to the constraint that $\|\mathbf{Bx}\|^2 = 1$, where $\mathbf{A}$ is a $N \times M$ matrix, $\mathbf{x}$ is a $N \times 1$ vector and $\mathbf{B}$ is a $P \times M$ matrix with $p < M$. The function of the matrix $\mathbf{B}$ is to select some coordinates of $\mathbf{x}$. Denoting by $\mathbf{z}$ the vector $\mathbf{Bx}$ and by $\mathbf{y}$ the $(M - P) \times 1$ vector made up of the other coordinates of vector $\mathbf{x}$, we have:

$$\mathbf{Ax} = \mathbf{Cy} + \mathbf{Dz}, \tag{2.106}$$

where $\mathbf{C}$ and $\mathbf{D}$ are respectively $N \times (M - P)$ and $N \times P$ matrices. Then the original optimization problem is now becoming:

$$\underset{\mathbf{y},\mathbf{z}}{\text{Min}} \|\mathbf{Cy} + \mathbf{Dz}\|^2 \quad \text{subject to } \|\mathbf{z}\|^2 = 1. \tag{2.107}$$

Applying the technique of Lagrange multipliers, this problem is equivalent to the following:

$$\underset{\mathbf{y}\,\mathbf{z}}{\text{Min}} F = \|\mathbf{Cy} + \mathbf{Dz}\|^2 + \lambda(1 - \|\mathbf{z}\|^2). \tag{2.108}$$

Computing the partial derivatives of criterion F with respect to $\mathbf{y}$ and $\mathbf{z}$, we have

$$\frac{\partial F}{\partial \mathbf{y}} = 2(\mathbf{C}^T\mathbf{Cy} + \mathbf{C}^T\mathbf{Dz}),$$

and

$$\frac{\partial F}{\partial \mathbf{z}} = 2(\mathbf{D}^T\mathbf{Dz} + \mathbf{D}^T\mathbf{Cy} - \lambda\mathbf{z}).$$

Then setting those partial derivatives to zero, we obtain:

$$\mathbf{y} = -(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{D}\mathbf{z}, \tag{2.109}$$

$$(\mathbf{D}^T\mathbf{D} - \mathbf{D}^T\mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{D})\mathbf{z} = \lambda\mathbf{z}. \tag{2.110}$$

According to Equation (2.110), $\mathbf{z}$ is an eigenvector of the matrix $\mathbf{E}$= $(\mathbf{D}^T\mathbf{D} - \mathbf{D}^T\mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{D})$. Also, by substituting $\mathbf{y}$ in Equation (2.107) by Equation (2.109), we have:

$$\begin{aligned} \left\|(\mathbf{I} - \mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T)\mathbf{D}\mathbf{z}\right\|^2 &= \mathbf{z}^T\mathbf{D}^T(\mathbf{I} - \mathbf{C}(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T)\mathbf{D}\mathbf{z} \\ &= \lambda\left\|\mathbf{z}\right\|^2 \\ &= \lambda. \end{aligned}$$

In order to minimize the cost function in Equation (2.107), so $\mathbf{z}$ is actually the eigenvector of the matrix $\mathbf{E}$ corresponding to the smallest eigenvalue.

Afterwards, $\mathbf{y}$ can be obtained by substituting $\mathbf{z}$ into Equation (2.109). Then this constrained optimization problem can be solved.