THE UNIVERSITY OF HULL

REGISTRATION TECHNIQUES FOR COMPUTER ASSISTED ORTHOPAEDIC SURGERY

being a Thesis submitted for the Degree of Doctor of Philosophy in the University of Hull

By

Qingde Li, BSc

APRIL 2002

Table of Contents

List of Tables

List of Figures

Abstract

The registration of 3D preoperative medical data to patients is a key task in developing computer assisted surgery systems. In computer assisted surgery, the patient in the operation theatre must be aligned with the coordinate system in which the preoperative data has been acquired, so that the planned surgery based on the preoperative data can be carried out under the guidance of the computer assisted surgery system.

The aim of this research is to investigate registration algorithms for developing computer assisted bone surgery systems. We start with reference mark registration. New interpretations are given to the development of well known algorithms based on singular value decomposition, polar decomposition techniques and the unit quaternion representation of the rotation matrix. In addition, a new algorithm is developed based on the estimate of the rotation axis. For non-land mark registration, we first develop iterative closest line segment and iterative closest triangle patch registrations, similar to the well known iterative closest point registration, when the preoperative data are dense enough. We then move to the situation where the preoperative data are not dense enough. Implicit fitting is considered to interpolate the gaps between the data. A new ellipsoid fitting algorithm and a new constructive implicit fitting strategy are developed. Finally, a region to region matching procedure is proposed based on our novel constructive implicit fitting technique. Experiments demonstrate that the new algorithm is very stable and very efficient.

Acknowledgements

I would like to thank John G. Griffiths, my supervisor, for his great supervision and many valuable suggestions. His constant support during this research have greatly encouraged me in completing my study. I am also grateful to professor Roger Phillips for providing me with the opportunity to study in the Department, as well as for his guidance through the first year of my research in the Spinal System Project. Sincere thanks also go to Warren J. Viant for his frequent help in many respects during my research.

I also wish to thank James Ward, George Sisias, Michael Bielby, and Dr Yonggen Zhu for their help and friendship.

Finally, I would like to express my sincere thanks to my wife Jin Jin. Without her support of my research in Hull and my family in China, my research would have taken much longer to finish.

Hull, England
January 31, 2000

Qingde Li

Main Contribution

1. A constructive implicit fitting technique has been developed to fit a set of scattered points using gate functions. With this technique, the data are first partitioned with geometric primitives into small data sets such that each sub-data set can be described well by a low degree algebraic surface. These locally fitted shapes are then combined with gate functions to obtain the overall fitting.

2. The iterative closest line segment (ICL) and iterative closest triangle patch (ICT) algorithms have been developed along similar lines to the ICP algorithm. Compared with the ICP algorithm, the ICL and the ICT algorithms are much less sensitive to the initial orientation. Our main work in these areas has been published in *Computers and Mathematics with Applications*[57].

3. A technique for fitting an ellipsoid to a set of scattered points is given. It is known that an equation of the second degree in three variables represents an ellipsoid when the leading form is positive definite. To fit an ellipsoid based on this constraint leads to a nonlinear optimization procedure with multiple constraints, which in general cannot guarantee an optimal solution. The presented fitting algorithm is a very simple and very stable procedure based on solving a generalized eigen system. It is almost a closed form solution, in the sense that in most cases, it takes just one iteration. This part of the work has been submitted for publication and is currently under review.

4. A region to region matching procedure is proposed for non landmark registration using our constructive implicit fitting techniques.

5. For reference mark registration, the conventional algorithms based on singular value decomposition, the unit Quaternion and Polar decomposition have been shown in more compact and direct ways. In addition, a new algorithm based on an estimate of the rotation axis is given, which involves only the computation of eigenvectors of a real 3×3 symmetric matrix.

6. Conventional reference mark registration has been extended to general geometric primitive registration, such as line segments, plane patches, straight lines and planes in 3D. In addition, a method for fitting straight lines based on geometric distance is given by solving a simple eigen system.

Chapter 1

Introduction

Matching a preoperative medical image with intraoperative data is a fundamental task in developing a computer assisted operation system. In computer assisted surgery, images that are taken from patients preoperatively at different times and by different sensors or from different viewpoints are input into computers. With computer graphics, image processing and image registration techniques, these images are analyzed, compared and synthesized to obtain sufficient information for diagnosis, surgical planning, carrying out surgical procedures and post-operative evaluation. In order to apply the pre-designed operation plan based on preoperative images during the operation, the preoperative data need to be registered with the patient in the operation theatre. That is, the virtual coordinate system corresponding to the computer graphics model must be aligned with the physical coordinate system used in the operation theatre for obtaining intraoperative data. Once two such coordinate systems are aligned, the operation can be carried out under the guidance of the computer assisted operation system. As can be seen, such a matching must be accurate and quick enough to be carried out in real time.

1.1 Survey of Current Techniques

In the last ten years or so, various techniques have been developed to deal with medical data matching problems in general, not only in the medical field, but also in computer vision. A survey of image registration techniques in general has been presented in [14] and [41].

Medical image registration is a wide research area. It is so diversified that it can be subdivided into many sub-areas, any of which has its own specific type of medical data to be registered. First of all, the images to be registered can have different modalities. For example, in medicine a body can be imaged through computed tomography(CT), magnetic resonance imaging(MRI), positron emission tomography(PET), single photon emission computed tomography(SPECT), and ultrasound. Secondly, the image can have different dimensionalities. It can be two dimensional or three dimensional. Thirdly, the transformation that links the images to be registered can be rigid, affine, projective, or curved. Finally, during the registration procedure, external land marks might be used. All these aspects make the registration problems very much diversified. Any combination gives a different type of registration problem.

In computer assisted surgery, registration approaches can be roughly classified either as landmark registration or as non-landmark registration according to whether external artificial fiducial markers are used or not. Usually, the markers used in computer assisted surgery systems can be either frames or a set of points[69]. In reference marker registration, external marks are attached to the operation area of patients before preoperative data are taken, in order to establish precise correspondence among the measurements made in the different image modalities. As reference mark registration techniques are usually quick and efficient, they are widely implemented in computer assisted operation systems.

Finding the relationship between two coordinate systems by using pairs of measured coordinates of a number of points in both systems is a classical problem. The solution has application in computer vision, robotics, and computer assisted surgery systems. In practice, an iterative algorithm can be used to compute the unknown rigid transformation. Since a good correspondence is known to exist, a good initial guess can be obtained directly either by a geometric method or a linear algebraic method. This initial guess is then applied to an iterative optimization procedure to refine the initial guess. Since the initial guess is likely to be very close to the true value, a very accurate result can be found by this method. The problem is that, with the increase in the number of data points, the optimization procedure becomes very slow.

The first closed-form solution for the problem was given by Farrell and Stuelpnagel using polar decomposition in 1966 [29]. However, their result is not known to most of us and is seldom mentioned in literature in the area of computer vision and medical data registration. In 1987, Horn established an algorithm by representing a rotation with a unit quaternion [39]. This is a complete and also a closed-form solution for the problem. But this algorithm is not popularly used in computer assisted surgery systems. The problem may be that the author used too much space to discuss the theory of quaternions which is not closely related to the algorithm. Actually, all the algorithm needs is just the quaternion representation of a rotation (formed by Rodrigues parameters) which has been long established[92]. In Chapter 3, we will see how directly the rigid transformation can be estimated accurately by using the unit quaternion representation. In the same year, Arun, Huang and Blostein developed another algorithm based on Singular value decomposition(SVD) [4], which was immediately welcomed by researchers in the area of robotics and computer vision as well as computer assisted surgery. The reason is not that the algorithm is better than the quaternion algorithm, but because the algorithm has been stated more directly. This first version of the SVD algorithm does not work properly when the data points are almost coplannar. In this case, a reflection rather than a proper rotation may be returned with the SVD algorithm. The SVD algorithm was later improved by Uneyama in 1991 [99]. The modified SVD algorithm will always return a rotation no matter how the data points are distributed in space. In 1988, Horn et al. developed another algorithm by using polar decomposition [40]. But this algorithm is not new because the solution given by Farrell and Stuelpnagel in 1966 is more general and more complete. In fact, Horn's algorithm given in [40] has the same problem as the original version of the SVD algorithm when the data sets are almost coplanar. In 1994, Kanatani [44] revised the three algorithms in a refined form and modified the polar decomposition algorithm.

Though the reference mark registration technique is widely used in many computer assisted operation systems, there is more and more interest in non-landmark registration. This is because reference mark registration techniques are invasive. Over the past few years, great efforts have been made towards non-landmark registration, which uses information extracted purely from images to be registered [10] [31] [15] [54] [53] [52] [65] [94]. For 3D-3D data matching, the non reference-marker registration methods can basically be grouped either as surface matching or as volume matching.

As the volume matching method is not quite suitable to our problem, it is not considered in this thesis. As far as surface matching techniques are concerned, they can be further classified into two categories: the iterative closest point (ICP) methods and the 'Head-Hat' methods. The idea of the ICP algorithm is to turn non-landmark registration problems into landmark registration problems by iteratively calculating the closest points for one data set to another. During the procedure, reference mark registration techniques are repeatedly used to find new transformations. This procedure is always convergent, but the problem is that it does not always converge correctly to the optimal solution. Unlike the ICP technique, the Head-Hat methods treat one data set as 'hat' and the second data set as 'head'. The transformation is found by matching the hat to the head with the help of some numerical minimizing procedures. As can be seen, when iterative numerical optimization is used, there is always a risk that the solution falls into a local minimum, and hence the true transformation cannot always be found correctly if a good initial guess is not available. In addition, in order to speed up the registration procedure, most 'Head-Hat' algorithms involve computing distance maps, and they are inevitably computationally expensive.

1.2 The focus of this research and research methodology

This research will focus on the development of 3D medical data registration techniques for computer assisted orthopaedic surgery systems. As the surgical objects can be treated as rigid, our problem can be formally addressed as follows:

Let two sets of points D_{pre} and D_{intra} be sampled from same surface of a rigid human bone according to different coordinate systems C_{pre} and C_{intra} respectively. We wish to estimate the rigid transformation that links the two coordinate systems from the information provided by D_{pre} and D_{intra}.

In computer assisted surgery, preoperative 3D medical information is often provided by CT, MRI, X-ray or any synthesized data with different types of formats. It is usually a very large and dense data set. In most cases, the overall shape of the surgical object can be reconstructed with predictable accuracy from preoperative data. On the other hand, the intraoperative data are much smaller. Sometimes the data set just contains very few points. This is because during an operation, the exposure of patients to radiation should be kept as low as possible. Secondly, once the patient is on the operation table, some data scanners such as the CT might be too cumbersome to be used any more. However, small numbers of points from the surface of the surgical object can be obtained much more easily. For example, the 3D position can be reconstructed from two orthogonal x-ray images, or can be directly obtained using a surgical pointer. In any of these cases, it is difficult to obtain a huge quantity of data points, and so the precise shape of the surgical object cannot be reconstructed. This implies that the algorithms based on feature matching, statistical correlation analysis techniques cannot be used.

For the above shape matching problem, one solution could be the use of the ICP algorithm. However, this algorithm has the drawback that it does not always converge to the expected orientation [43]. When an initialization does not lead to an expected convergence, a new randomly specified initial guess should be applied again until a satisfactory result is obtained. Therefore the ICP algorithm must be improved to increase its stability and efficiency, as the registration problem in consideration must be sufficiently accurate and capable of finishing in a very short time. One possible extension for the ICP algorithm is to provide geometric information in the matching data such as the distance between two points. When two data sets are linked by a rigid transformation, the distance between any pair of points in one data set will be very close to the distance between their closest counterparts in the second data set. Instead of searching for the closest points in each iteration in the ICP algorithm, closest line segments or closest triangle patches can be considered. Following this idea, algorithms called iterative closest line segment and iterative closest triangle patches will be developed.

Another consideration for our registration problem follows the idea of 'Head-Hat' matching. The conventional 'Head-Hat' matching technique models the 'head' as a mesh surface. The matching procedure is basically a minimization of a sum of distances. As the computation of distance for each point to this kind of model surface is obtained by searching all the possible polygons in the mesh, the computation is very costly [69]. The way to improve the matching procedure is to compute the distance map and store it in computer memory before the optimization starts. Though this will greatly shorten the registration time, it requires the computer to have an extremely large memory, as the distance map can be as large as hundreds of megabytes [48]. Most Head-Hat matching techniques are used in the area of pattern recognition, image analysis, medical diagnosis, but not widely used in developing computer assisted surgery systems. In fact, when the intraoperative data are too sparse and the surface of the operation object is too rough and too bumpy, it is almost impossible to recover the orientation precisely. On the other hand, when the surface of the object is quite smooth, it might be able to approximate the surgical surface with some simple 'distance' functions, and thus the massive computation involved in computing the distance from a point to the surgical object can be avoided.

However, fitting a set of scattered points with a distance function is a very difficult task. In practice, we often fit the data with an implicit function, as it is comparatively easier than fitting a distance function. In mathematics, the implicit surface is represented as the roots of a function, which naturally divides the space into three parts: the points that lies on the surface and the two data sets that lie on either side of the surface, according to whether the value of the function is 0, larger than 0 or less than 0. When the function is represented by a polynomial, the corresponding shape is usually called an algebraic surface. Most fitting techniques currently available are related to algebraic surface fitting. Some good discussion and investigation on implicit polynomial fitting can be found in [94] [95] [46] [93] [47]. However, pure implicit polynomial fitting methods have drawbacks in representing general geometric shapes. The main problem is that the shapes of low degree polynomials are too simple for general shapes and the shapes of high degree polynomials are unpredictable. The shape fitted can be much more complicated than the shape actually represented by the data. One feasible strategy is to break the whole data set into small pieces, with each subset of data fitted with a low degree algebraic surface, and then combine all these individually fitted shapes together to obtain the overall fitting. In this procedure, the key technique will be how to combine different implicitly represented shapes together. In this thesis, a constructive procedure to fit surface data with implicit functions has been developed.

1.3 Thesis organization

The main part of the dissertation is organized as follows. In Chapter 2, we provide some mathematical background and formulations for the problem. The concept of matrix scalar product is introduced, and some its properties are investigated. This is followed by a survey on the parameterization of a rotation matrix. The results are not new, but the survey reflects our own understanding of the problem. As the rotation matrix will be one of the main objects discussed in later chapters, some of its properties are discussed. The most important result we obtained is that the trace of a rotation matrix cannot be larger than $n - 2 + 2r_{ii}$ for any diagonal element r_{ii} of the matrix. Chapter 3 will review the three main algorithms used in the reference mark registration problem. We will see that all these algorithms can developed more directly and simply. In addition, a new algorithm based on an estimate of the rotation axis will be presented. Chapter 4 will discuss some generalizations of the result presented in Chapter 3. We will discuss how to estimate the rigid transformation from the correspondence of geometric primitives, such as from straight lines correspondence and plane correspondence. Chapter 5 address the iterative closest line segment registration and the iterative closest triangle patch registration, which are developed similarly to the conventional iterative closest point algorithm. The remaining two chapters are about implicit fitting. Chapter 6 will mainly discuss how to fit a set of 3D data with an ellipsoid, the only central quadric which is bounded. This fitting technique is extremely useful for fitting local implicit surface in our constructive implicit surface fitting method given in Chapter 7. It is also useful for establishing an initial guess in our region-to-region matching strategy, which will be discussed in Chapter 8. In Chapter 7, a constructive implicit fitting method is developed using gate functions. With this technique, a complex shape can be fitted with an implicit function by combining locally fitted implicit shapes. The fitted implicit function can served as an approximation to the distance function from a point to the surface in the 'Head-hat' matching algorithm. In Chapter 8, we will discuss the region to region matching strategy with the help of the constructive implicit surface fitting algorithm. Finally, Chapter 9 contains a summary and Chapter 10 contains possible future work.

Chapter 2

Mathematical preliminaries

2.1 The matrix scalar product and the Frobenius norm

Definition 2.1.1. Let $A = (a_{ij})$ be an $n \times m$ matrix. The Frobenius norm is defined by

$$\|A\| = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}^2}. \tag{2.1.1}$$

The following properties for Frobenius norm for any $n \times m$ matrices are obvious [91]:

$$A \neq 0 \text{ if and only if } \|A\| > 0,$$

$$\|\alpha A\| = |\alpha|\|A\|,$$

$$\|A + B\| \leq \|A\| + \|B\|,$$

where α is a real number.

The matrix norm has been defined in different ways in mathematics. In this thesis, the matrix norm always means the Frobenius norm defined above.

Definition 2.1.2. Let $A = (a_{ij})$ and $B = (b_{ij})$ be two $n \times m$ matrices. The scalar product of the two matrices, denoted by $A \cdot B$, is defined as

$$A \cdot B = \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij} b_{ij}. \tag{2.1.2}$$

It is evident that this definition is a natural generalization of the scalar product of two vectors.

Let A, B be $n \times m$ real matrices. Let A_{1*}, A_{2*}, $\cdots$, A_{n*} and A_{*1}, A_{*2}, $\cdots$, A_{*m} be the row vectors and column vectors of A, and let B_{1*}, B_{2*}, $\cdots$, B_{n*} and B_{*1}, B_{*2}, $\cdots$, B_{*m} the row vectors and column vectors of B.

Proposition 2.1.1.

$$A \cdot B = A_{1*} \cdot B_{1*} + A_{2*} \cdot B_{2*} + \cdots + A_{n*} \cdot B_{n*} \tag{2.1.3}$$

$$= A_{*1} \cdot B_{*1} + A_{*2} \cdot B_{*2} + \cdots + A_{*m} \cdot B_{*m}. \tag{2.1.4}$$

Proposition 2.1.2. *(1) For the identity $n \times n$ matrix I and any $n \times n$ matrix A,*

$$I \cdot A = tr(A). \tag{2.1.5}$$

(2) For any $n \times m$ matrices A and B,

$$A \cdot B = B \cdot A. \tag{2.1.6}$$

(3) For any $n \times m$ matrices A,B and C and any real numbers α and β,

$$(\alpha A + \beta B) \cdot C = \alpha(A \cdot C) + \beta(B \cdot C). \tag{2.1.7}$$

(4) For any $n \times m$ matrices A and B,

$$A \cdot B = tr(A^T B) = tr(AB^T) = tr(B^T A) = tr(BA^T). \tag{2.1.8}$$

(5) For any matrix A,

$$A \cdot A = \|A\|^2. \tag{2.1.9}$$

Proposition 2.1.3. *Let $A = (a_{ij})$ be an $n \times m$ matrix, and let X be an m-dimensional vector, and Y an n-dimensional vector. Then*

$$Y \cdot (AX) = A \cdot (YX^T). \tag{2.1.10}$$

Proof.

$$Y \cdot (AX) = \sum_{i=1}^{n} y_i \sum_{j=1}^{m} a_{ij} x_j = \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij} y_i x_j = A \cdot (YX^T).$$

□

Proposition 2.1.4. *Let A, B, C be $n \times m$, $n \times k$ and $k \times m$ matrices respectively. Then*

$$A \cdot (BC) = (AC^T) \cdot B = (B^T A) \cdot C. \tag{2.1.11}$$

Proof. From Proposition 2.1.2 (4),

$$A \cdot (BC) = tr(A(BC)^T) = tr((AC^T)B^T) = AC^T \cdot B.$$

In a similar way, we can show that

$$A \cdot (BC) = (B^T A) \cdot C.$$

□

Proposition 2.1.5. *Let A, B be $n \times m$ matrices. Then*

$$\|A - B\|^2 = \|A\|^2 + \|B\|^2 - 2A \cdot B. \tag{2.1.12}$$

Proof. According to the definition of the Frobenius norm, we have

$$\begin{aligned} \|A-B\|^2 &= \sum_{j=1}^{m} \|A_{*j} - B_{*j}\|^2 &(2.1.13)\\ &= \sum_{j=1}^{m}(\|A_{*j}\|^2 + \|B_{*j}\|^2 - 2A_{*j}\cdot B_{*j}) &(2.1.14)\\ &= \|A\|^2 + \|B\|^2 - 2A\cdot B. &(2.1.15)\end{aligned}$$

□

Corollary 2.1.6. *If R is a real orthogonal matrix, then*

$$\|A - RB\|^2 = \|A\|^2 + \|B\|^2 - 2R\cdot(AB^T). \tag{2.1.16}$$

Proof. The proof follows directly from propositions 2.1.5 and 2.1.4. □

2.2 Representation of rotation

Let $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ be a set of mutually orthogonal unit vectors which start at the origin in three dimensional Euclidean space. If $\mathbf{e}_3 = \mathbf{e}_1 \times \mathbf{e}_2$, such a set of vectors establishes a right-handed orthonormal system. Let $\{\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3\}$ be another right-handed orthonormal system. A transformation F such that $F\mathbf{e}_i$=$\mathbf{r}_i$, $(i = 1, 2, 3)$ can be represented by an orthonormal matrix R, i.e., $RR^T = I$, $det(R) = 1$. Conversely, any orthonormal matrix R with $det(R) = 1$ maps a right-handed orthogonal system to a right-handed orthogonal system. This kind of transformation is called a rotation transformation. Let G be the collection of all possible rotation transformations, G then forms a group with respect to the matrix product. This mathematical structure is normally referred to as $SO(3)$ (Special Orthogonal 3×3 matrices) in the theory of group representation.

Rotations represented by 3×3 matrices need nine parameters with the constraint that $RR^T = I$ and $det(R) = 1$. In practice, it is possible to represent a rotation by a set of fewer than nine parameters. In this section, we present some of the parametrizations of the rotation matrix that are in common use, each of which has certain advantages and disadvantages.

2.2.1 The rotation matrix in terms of the Euler angles

There are different ways to represent a rotation matrix with Euler angles. Let $R_3(\gamma)R_2(\beta)R_1(\alpha)$ denote the result of the following three consecutive rotations: first rotate by an angle α around axis 1, then a rotate by an angle β around axis 2, and finally rotate by an angle γ around axis 3. Then any rotation can be accomplished by one of the following 12 ways [87]:

$$
\begin{array}{llll}
R_x(\gamma)R_y(\beta)R_z(\alpha), & R_x(\gamma)R_y(\beta)R_x(\alpha), & R_x(\gamma)R_z(\beta)R_y(\alpha), & R_x(\gamma)R_z(\beta)R_x(\alpha), \\
R_y(\gamma)R_x(\beta)R_z(\alpha), & R_y(\gamma)R_x(\beta)R_y(\alpha), & R_y(\gamma)R_z(\beta)R_x(\alpha), & R_y(\gamma)R_z(\beta)R_y(\alpha), \\
R_z(\gamma)R_x(\beta)R_y(\alpha), & R_z(\gamma)R_x(\beta)R_z(\alpha), & R_z(\gamma)R_y(\beta)R_x(\alpha), & R_z(\gamma)R_y(\beta)R_z(\alpha).
\end{array}
$$

where R_x, R_y and R_z represent the rotation around x-axis, y-axis and z-axis respectively. Here we chose the form $R_z(\gamma)R_x(\beta)R_z(\alpha)$, which appears now to be the most popular Euler angle representation, where angle β is between $[0, \pi]$ and other two angles are within $[0, 2\pi)$. The rotation R can be written as

$$
\begin{aligned}
R(\alpha, \beta, \gamma) &= R_z(\gamma)R_x(\beta)R_z(\alpha) \\
&= \begin{pmatrix} C_1C_3 - S_1C_2S_3 & S_1C_3 + C_1C_2S_3 & S_2S_3 \\ -C_1S_3 - S_1C_2C_3 & -S_1S_3 + C_1C_2C_3 & S_2C_3 \\ S_1S_2 & -C_1S_2 & C_2 \end{pmatrix}, \qquad (2.2.1)
\end{aligned}
$$

where

$$C_1 = \cos\alpha, \qquad C_2 = \cos\beta, \qquad C_3 = \cos\gamma,$$

$$S_1 = \sin\alpha, \qquad S_2 = \sin\beta, \qquad S_3 = \sin\gamma.$$

A major problem of representing a rotation with Euler angles is the loss of one dimension in some special cases and this means that smooth rotation does not always correspond to smooth parametrization.

2.2.2 The rotation matrix in terms of the rotation axis and the rotation angle

Let R be a rotation matrix. Then there must exist a unit vector $\mathbf{n}$ such that $R\mathbf{n} = \mathbf{n}$. In fact, let λ be an eigenvalue of R, and let $\mathbf{n}$ be the unit eigenvector associated with λ, i.e., $R\mathbf{n} = \lambda\mathbf{n}$. Since R is real, we must have $R\bar{\mathbf{n}} = \bar{\lambda}\bar{\mathbf{n}}$, where $\bar{\lambda}$ and $\bar{\mathbf{n}}$ represent the complex conjugates of number λ and vector $\mathbf{n}$. Therefore, we have

$$1 = \bar{\mathbf{n}}^T R^T R\mathbf{n} = \bar{\lambda}\lambda\mathbf{n}\bar{\mathbf{n}} = \bar{\lambda}\lambda.$$

Thus, $|\lambda| = 1$. Let the three eigenvalues of R be λ_1, λ_2 and λ_3. As complex eigenvalues will appear pairwise, from $det(R) = \lambda_1\lambda_2\lambda_3 = 1$ we know that there must exist at least one real eigenvalue, whose value is 1.

For any rotation matrix, we call the unit eigenvector associated with the eigenvalue 1 the rotation axis and is denoted by $\mathbf{n}$. This unit vector can then be expanded into a right-handed orthonormal system $\mathbf{n}, \mathbf{n}_1, \mathbf{n}_2$. Let matrix $P = (\mathbf{n}, \mathbf{n}_1, \mathbf{n}_2)$, then P is a rotation and

$$P^T R P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{pmatrix}.$$

where θ is the rotation angle of R by axis $\mathbf{n}$. From this we have

$$R = P \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{pmatrix} P^T.$$

The geometric meaning of this decomposition is evident. To rotate, we could first rotate the rotation axis $\mathbf{n}$ with P^T such that it coincides with x-axis $\mathbf{e}_1 = (1,0,0)^T$, followed by a rotation about the x-axis by an angle θ, finally rotating $\mathbf{n}$ back to its original position with the inverse rotation of P^T.

For any 3D vector $\mathbf{r}$,

$$\begin{aligned} R\mathbf{r} &= P \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} P^T\mathbf{r} & (2.2.2) \\ &= (\mathbf{n}\cdot\mathbf{r})\mathbf{n} + \cos\theta((\mathbf{n}_1\cdot\mathbf{r})\mathbf{n}_1 + (\mathbf{n}_2\cdot\mathbf{r})\mathbf{n}_2) + \sin\theta((\mathbf{n}_1\cdot\mathbf{r})\mathbf{n}_2 - (\mathbf{n}_2\cdot\mathbf{r})\mathbf{n}_1) \\ &= \cos\theta\mathbf{r} + (1-\cos\theta)(\mathbf{n}\cdot\mathbf{r})\mathbf{n} + \sin\theta(\mathbf{n}\times\mathbf{r}), & (2.2.3) \end{aligned}$$

since

$$\mathbf{r} = (\mathbf{n}\cdot\mathbf{r})\mathbf{n} + (\mathbf{n}_1\cdot\mathbf{r})\mathbf{n}_1 + (\mathbf{n}_2\cdot\mathbf{r})\mathbf{n}_2,$$

$$\mathbf{n}\times\mathbf{r} = (\mathbf{n}_1\cdot\mathbf{r})\mathbf{n}_2 - (\mathbf{n}_2\cdot\mathbf{r})\mathbf{n}_1.$$

For $\mathbf{n} = (\alpha,\beta,\gamma)^T$, if we set

$$N = \begin{pmatrix} 0 & -\gamma & \beta \\ \gamma & 0 & -\alpha \\ -\beta & \alpha & 0 \end{pmatrix}, \tag{2.2.4}$$

then equation (2.2.2) can be further written as

$$R\mathbf{r} = \left(\cos\theta\mathbf{I} + (1-\cos\theta)\mathbf{n}\mathbf{n}^T + \sin\theta N\right)\mathbf{r}. \tag{2.2.5}$$

from which the rotation matrix R can be presented in the following way

$$R = \cos\theta I + (1-\cos\theta)\mathbf{n}\mathbf{n}^T + \sin\theta N. \tag{2.2.6}$$

The above representation can also be obtained geometrically [3]. Here we work mainly from the algebraic point of view.

2.2.3 The rotation matrix in terms of a 3D Vector

A rotation matrix can also be represented by a 3D vector $\mathbf{r} = (u, v, w)^T$ with $\|\mathbf{r}\| > 0$. For $\mathbf{r}$, let $\theta = \|\mathbf{r}\|$, and let $\mathbf{n} = \mathbf{r}/\theta = (\alpha, \beta, \gamma)^T$. From $\mathbf{r}$, A skew-symmetric matrix S can be constructed as:

$$S = \begin{pmatrix} 0 & -w & v \\ w & 0 & -u \\ -v & u & 0 \end{pmatrix} = \theta N, \tag{2.2.7}$$

where

$$N = \begin{pmatrix} 0 & -\gamma & \beta \\ \gamma & 0 & -\alpha \\ -\beta & \alpha & 0 \end{pmatrix}, \tag{2.2.8}$$

Let

$$R = \exp S = I + \theta N + \frac{1}{2!}\theta^2 N^2 + \frac{1}{3!}\theta^3 N^3 + \cdots \tag{2.2.9}$$

Then R has following properties

$$RR^T = I, \tag{2.2.10}$$

$$R\mathbf{n} = \mathbf{n}. \tag{2.2.11}$$

since for N, $N^T = -N$ and $N\mathbf{n} = \mathbf{n} \times \mathbf{n} = 0$. This shows that R is orthonormal and has a rotation axis $\mathbf{n}$. Note that the characteristic function of N is $det(\lambda I - N) =$

$\lambda^3 + \lambda$, $N^3 = -N$, (2.2.9) can be further simplified as

$$R = I + \sin\theta N + (1 - \cos\theta)N^2. \tag{2.2.12}$$

To show that R is a rotation, we need only to show that R transforms a right-handed coordinate system into a right-handed coordinate system. Let $\{\mathbf{n}, \mathbf{n}_1, \mathbf{n}_2\}$ be a right-handed coordinate system expanded from $\mathbf{n}$. Then from equation (2.2.12), it can be seen that

$$R\mathbf{n}_1 \quad = \sin\theta\mathbf{n}_2 + \cos\theta\mathbf{n}_1, \tag{2.2.13}$$

$$R\mathbf{n}_2 \quad = -\sin\theta\mathbf{n}_1 + \cos\theta\mathbf{n}_2. \tag{2.2.14}$$

Hence,

$$\begin{aligned} R\mathbf{n}_1 \times R\mathbf{n}_2 &= \cos^2\theta(\mathbf{n}_1 \times \mathbf{n}_2) - \sin^2\theta(\mathbf{n}_2 \times \mathbf{n}_1) \\ &= \mathbf{n}_1 \times \mathbf{n}_2 = \mathbf{n} = R\mathbf{n}. \end{aligned} \tag{2.2.15}$$

That is, R maps a right-handed orthogonal system into a right-handed orthogonal system.

Conversely, let R be a rotation. From section 2.2.2, R has a rotation axis $\mathbf{n} = (\alpha, \beta, \gamma)^T$ and a rotation angle θ. If the value of θ is set to be 2π when $R = I$, then the vector $\mathbf{r} = \theta\mathbf{n}$ will never be zero. Let S be the skew-symmetric matrix constructed from $\mathbf{r}$ according to (2.2.7) and let the rotation R_1 be defined as (2.2.9), then

$$R_1 = I + \sin\theta N + (1 - \cos\theta)N^2. \tag{2.2.16}$$

However, $N^2 = \mathbf{n}\mathbf{n}^T - I$. Thus

$$R_1 = \cos\theta I + (1 - \cos\theta)\mathbf{n}\mathbf{n}^T + \sin\theta N, \tag{2.2.17}$$

which is exactly the same as (2.2.6). This shows that any rotation matrix R can always be expressed by a 3D vector.

2.2.4 Cayley's parameterization of the rotation matrix

Cayley's parameterization, also using a skew-symmetric matrix, is another way to parameterize a rotation matrix. If $S \neq 0$ is a skew-symmetric 3×3 matrix defined by (2.2.7), with a vector $\mathbf{r} = (u, v, w)$, and

$$R = (I - S)(I + S)^{-1}, \tag{2.2.18}$$

then,

$$\begin{aligned} R^T R &= (I - S)^{-1}(I + S)(I - S)(I + S)^{-1} \\ &= (I - S)^{-1}(I - S)(I + S)(I + S)^{-1} = I. \end{aligned}$$

This shows that R defined in (2.2.18) is orthogonal. Let vector $\mathbf{n} = \mathbf{r}/\sigma = (\alpha, \beta, \gamma)$, where $\sigma = \|\mathbf{r}\| \neq 0$. Let N be defined according to (2.2.8) from $\mathbf{n}$, then

$$\begin{aligned} (I - S)(I + S)^{-1} &= I - 2(\sigma N - \sigma^2 N^2 + \sigma^3 N^3 - \cdots) \\ &= I - \frac{2\sigma}{1 + \sigma^2} N + \frac{2\sigma^2}{1 + \sigma^2} N^2 \\ &= I - \frac{2}{1 + \sigma^2} S + \frac{2}{1 + \sigma^2} S^2, \end{aligned} \tag{2.2.19}$$

since $N^3 = -N$. To see that R is a rotation, we need only to show that R transforms a right-handed coordinate system into a right-handed coordinate system. Expand $\mathbf{n}$ into a right-handed coordinate system $\{\mathbf{n}, \mathbf{n}_1, \mathbf{n}_2\}$. Then from equation (2.2.19), it can be shown directly that $\mathbf{n} = R\mathbf{n} = R\mathbf{n}_1 \times R\mathbf{n}_2$ as $\mathbf{n} = \mathbf{n}_1 \times \mathbf{n}_2$. This shows R is indeed a rotation.

Conversely, if R is a rotation. When R does not have $\lambda = -1$ as its eigenvalue, $R + I$ will be non-singular. Let

$$S = (I - R)(I + R)^{-1}, \tag{2.2.20}$$

then S will be skew-symmetric. In fact, from equation (2.2.20),

$$S(I+R) = I - R,$$

which implies that

$$(R+I)S^T = R - I,$$

since $RR^T = I$. Thus

$$S^T = (R+I)^{-1}(R-I) = (R-I)(R+I)^{-1} = -S$$

and the fact that

$$R = (I-S)(I+S)^{-1}$$

follows directly from this reversible relation between R and S.

It can be seen from equation (2.2.18) that a rotation that can be expressed in this way cannot have eigenvalue -1.

From equation (2.2.19), any rotation matrix that does not have eigenvalue -1 can be expressed in the following way

$$R = \rho \begin{pmatrix} 1+u^2-v^2-w^2 & 2(uv-w) & 2(uw+v) \\ 2(uv+w) & 1-u^2+v^2-w^2 & 2(vw-u) \\ 2(uw-v) & 2(vw+u) & 1-u^2-v^2+w^2 \end{pmatrix}, \qquad (2.2.21)$$

where $\rho = (1+u^2+v^2+w^2)^{-1}$.

2.2.5 Rotation in terms of quaternions

The quaternion, invented by Hamilton [3], has a similar form to a complex number. Hamilton had been interested in complex numbers since the early 1830s. In 1833, he first showed that complex numbers form an algebra of couples, i.e., they can be put in the form $a+b\mathbf{i}$ with $\mathbf{i}^2=-1$. Over the next ten years, he tried to extend this concept to define a triplet, with one real and two imaginary units, $\mathbf{i}$ and $\mathbf{j}$. However, he could not do this . In 1843, when he was walking past a bridge, he suddenly realized that three, rather than two, imaginary units were needed, with the following properties:

$$\mathbf{i}^2=\mathbf{j}^2=\mathbf{k}^2=-1, \quad \mathbf{ij}=\mathbf{k}, \quad \mathbf{ji}=-\mathbf{k}.$$

A mathematical object in the form $\mathbf{q}=a+b\mathbf{i}+c\mathbf{j}+d\mathbf{k}$, with a,b,c,d real numbers, has been defined by Hamilton and is called a quaternion.

Extending the idea of a complex number, a quaternion can be thought of as a point in 4 dimensional space, or as a composition of a scalar and an ordinary vector, or as a complex number with three different imaginary parts.

The multiplication of two quaternions is also similar to the multiplication of two complex numbers. To see such a similarity, let $\mathbf{I}=(\mathbf{i},\mathbf{j},\mathbf{k})^T$ be column vector with components $\mathbf{i},\mathbf{j},\mathbf{k}$. A quaternion can then be represented as

$$\mathbf{q}=s+\mathbf{v}\cdot\mathbf{I},$$

where s is a real number and $\mathbf{v}$ is a 3×1 real vector.

Let $\mathbf{q}_1$ and $\mathbf{q}_2$ be two quaternions, then

$$\begin{aligned}\mathbf{q}_1\mathbf{q}_2 &= (s_1+\mathbf{v}_1\cdot\mathbf{I})(s_2+\mathbf{v}_2\cdot\mathbf{I}) \qquad (2.2.22)\\ &= s_1s_2+(s_1\mathbf{v}_2+s_2\mathbf{v}_1)\cdot\mathbf{I}+(\mathbf{v}_1\cdot\mathbf{I})(\mathbf{v}_2\cdot\mathbf{I}).\end{aligned}$$

Since

$$
\begin{aligned}
(\mathbf{v}_1 \cdot \mathbf{I})(\mathbf{v}_2 \cdot \mathbf{I}) &= (v_{1x}\mathbf{i} + v_{1y}\mathbf{j} + v_{1z}\mathbf{k})(v_{2x}\mathbf{i} + v_{2y}\mathbf{j} + v_{2z}\mathbf{k}) \\
&= -(v_{1x}v_{2x} + v_{1y}v_{2y} + v_{1z}v_{2z}) + (v_{1y}v_{2z} - v_{1z}v_{2y})\mathbf{i} \\
&\quad +(v_{1z}v_{2x} - v_{1x}v_{2z})\mathbf{j} + (v_{1x}v_{2y} - v_{1y}v_{2x})\mathbf{k} \\
&= -\mathbf{v}_1 \cdot \mathbf{v}_2 + (\mathbf{v}_1 \times \mathbf{v}_2) \cdot \mathbf{I},
\end{aligned}
$$

from (2.2.22), we have

$$
\mathbf{q}_1\mathbf{q}_2 = (s_1 s_2 - \mathbf{v}_1 \cdot \mathbf{v}_2) + (s_1\mathbf{v}_2 + s_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2) \cdot \mathbf{I}. \tag{2.2.23}
$$

As with complex numbers, the conjugate of a quaternion $\mathbf{q} = s + \mathbf{v} \cdot \mathbf{I}$ can be defined as $\mathbf{q}^* = s - \mathbf{v} \cdot \mathbf{I}$. From (2.2.23), we can see that

$$
\mathbf{q}\mathbf{q}^* = s^2 + \|\mathbf{v}\|^2,
$$

which is just the squared norm of vector $(s, v_x, v_y, v_z)^T$ and $\sqrt{\mathbf{q}\mathbf{q}^*}$ is called the magnitude of the quaternion and is denoted by $\|\mathbf{q}\|$.

For any quaternion $\mathbf{q}_1 = s_1 + \mathbf{v}_1 \cdot \mathbf{I}, \mathbf{q}_2 = s_2 + \mathbf{v}_2 \cdot \mathbf{I}$, we have

$$
\|\mathbf{q}_1\mathbf{q}_2\| = \|\mathbf{q}_1\|\|\mathbf{q}_2\|. \tag{2.2.24}
$$

In fact, from (2.2.23), we have

$$
\begin{aligned}
\|\mathbf{q}_1\mathbf{q}_2\|^2 &= (s_1 s_2 - \mathbf{v}_1 \cdot \mathbf{v}_2)^2 + \|s_1\mathbf{v}_2 + s_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2\|^2 \\
&= s_1^2 s_2^2 + (\mathbf{v}_1 \cdot \mathbf{v}_2)^2 + s_1^2\|\mathbf{v}_2\|^2 + s_2^2\|\mathbf{v}_1\|^2 + \|\mathbf{v}_1 \times \mathbf{v}_2\|^2 \\
&= (s_1^2 + \|\mathbf{v}_1\|^2)(s_2^2 + \|\mathbf{v}_2\|^2) = \|\mathbf{q}_1\|^2\|\mathbf{q}_2\|^2,
\end{aligned}
$$

since

$$
(\mathbf{v}_1 \cdot \mathbf{v}_2)^2 + \|\mathbf{v}_1 \times \mathbf{v}_2\|^2 = \|\mathbf{v}_1\|^2\|\mathbf{v}_2\|^2.
$$

A quaternion with a magnitude of one is called unit quaternion. It can be seen, for any non-zero quaternion $\mathbf{q}$, there always exists a quatenion $\mathbf{q}^{-1}$ such that $\mathbf{q}\mathbf{q}^{-1} = 1$ and $\mathbf{q}^{-1}$ is called the inverse of quaternion $\mathbf{q}$. The set of all non-zero quaternions plus the operation defined above form a group. Since $\mathbf{q}_1\mathbf{q}_2 \neq \mathbf{q}_2\mathbf{q}_1$ in general, the group is non-commutative. From equation (2.2.24), it can be seen that the multiplication of two unit quaternions will still be a unit quaternion, and so all unit quaternions form a subgroup, which, we will see later, is closely related to rotation.

The matrix representation of a unit quaternion

Any quaternion can always be represented by a 4×4 matrix. Let quaternion $\mathbf{q} = s + \mathbf{v} \cdot \mathbf{I}$ be a quaternion with $\mathbf{v} = (v_x, v_y, v_z)^T$, let

$$\mathbf{Q} = \begin{pmatrix} s & -v_x & -v_y & -v_z \\ v_x & s & -v_z & v_y \\ v_y & v_z & s & -v_x \\ v_z & -v_y & v_x & s \end{pmatrix} = \begin{pmatrix} s & -\mathbf{v}^T \\ \mathbf{v} & sI_{3\times 3} + C_v \end{pmatrix}, \tag{2.2.25}$$

where

$$C_v = \begin{pmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{pmatrix}.$$

The remarkable thing is that this representation establishes a one-to-one mapping between the set of quaternions and the set of matrices that have form (2.2.25) which preserves the quaternion operation. More precisely, the matrix representation for the product of two quaternions is just the matrix product of their corresponding matrix representations. Mathematically, we have

Proposition 2.2.1. *Let* $\mathbf{q}_1 = s_1 + \mathbf{v}_1 \cdot \mathbf{I}$ *and* $\mathbf{q}_2 = s_2 + \mathbf{v}_2 \cdot \mathbf{I}$ *be two quaternions, and let* $\mathbf{Q}_1$ *and* $\mathbf{Q}_2$ *be their matrix representations. If* $\mathbf{q}_1\mathbf{q}_2 = s + \mathbf{v} \cdot \mathbf{I}$, *then we have*

$$\mathbf{Q}_1\mathbf{Q}_2 = \begin{pmatrix} s & -\mathbf{v}^T \\ \mathbf{v} & sI_{3\times3} + C_v \end{pmatrix}, \tag{2.2.26}$$

which corresponds to quaternion $\mathbf{q}_1\mathbf{q}_2$.

Proof. Let $\mathbf{Q}_1, \mathbf{Q}_2$ be the matrices representations of quaternions $\mathbf{q}_1, \mathbf{q}_2$ respectively. Then

$$\mathbf{Q}_1 = s_1 I_{4\times4} + \begin{pmatrix} 0 & -\mathbf{v}_1^T \\ \mathbf{v}_1 & C_{v_1} \end{pmatrix}, \qquad \mathbf{Q}_2 = s_2 I_{4\times4} + \begin{pmatrix} 0 & -\mathbf{v}_2^T \\ \mathbf{v}_2 & C_{v_2} \end{pmatrix}, \tag{2.2.27}$$

$$\begin{aligned} \mathbf{Q}_1\mathbf{Q}_2 &= s_1 s_2 I_{4\times4} + s_1 \begin{pmatrix} 0 & -\mathbf{v}_2^T \\ \mathbf{v}_2 & C_{v_2} \end{pmatrix} + s_2 \begin{pmatrix} 0 & -\mathbf{v}_1^T \\ \mathbf{v}_1 & C_{v_1} \end{pmatrix} \\ &+ \begin{pmatrix} -\mathbf{v}_1 \cdot \mathbf{v}_2 & -(\mathbf{v}_1 \times \mathbf{v}_2)^T \\ \mathbf{v}1 \times \mathbf{v}_2 & -\mathbf{v}_1\mathbf{v}_2^T + C_{v_1}C_{v_2} \end{pmatrix}. \end{aligned} \tag{2.2.28}$$

It can be shown that

$$-\mathbf{v}_1\mathbf{v}_2^T + C_{v_1}C_{v_2} = -\mathbf{v}_1 \cdot \mathbf{v}_2 I + C_{v_1 \times v_2}.$$

Thus

$$\begin{aligned} \mathbf{Q}_1\mathbf{Q}_2 &= (s_1 s_2 - \mathbf{v}_1 \cdot \mathbf{v}_2) I_{4\times4} \\ &+ \begin{pmatrix} 0 & -(s_1\mathbf{v}_2 + s_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2)^T \\ s_1\mathbf{v}_2 + s_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2 & C_{s_1\mathbf{v}_2 + s_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2} \end{pmatrix}. \end{aligned} \tag{2.2.29}$$

□

Proposition 2.2.2. *Let* $\mathbf{Q}_1$ *be the matrix representation of a quaternion* $\mathbf{q}_1$*, then*

$$\mathbf{q}_1\mathbf{q}_2 = \mathbf{Q}_1\mathbf{q}_2, \tag{2.2.30}$$

where quaternions are treated as four dimensional vectors.

The proof is direct.

Let $\mathbf{Q}$ be the matrix representation of quaternion $\mathbf{q} = s + \mathbf{v} \cdot \mathbf{I}$. From the definition of the matrix representation, the matrix representation for $\mathbf{q}^*$ is just the transpose of matrix $\mathbf{Q}$, i.e., $\mathbf{Q}^T$. The interesting thing is that

$$\mathbf{Q}\mathbf{Q}^T = (s^2 + \|\mathbf{v}\|^2)I_{4\times4} = \|\mathbf{q}\|^2 I_{4\times4}, \tag{2.2.31}$$

which means that the matrix $\mathbf{Q}$ will always be orthogonal whenever $\mathbf{q} \neq 0$.

As we know, multiplying a unit complex number by another complex number is equivalent to a rotation in the 2D plane. Similarly, multiplying a unit quaternion by another quaternion is equivalent to a rotation in 4D space. In fact, when quaternion $\mathbf{q}$ is unit, the matrix $\mathbf{Q}$ will represent a rotation in four dimensional space, as we have

Proposition 2.2.3. *For a unit quaternion, its corresponding matrix representation is a rotation. That is*

$$\mathbf{Q}\mathbf{Q}^T = I_{4\times4}, \quad det(\mathbf{Q}) = 1. \tag{2.2.32}$$

Proof. If $\mathbf{q} = s + \mathbf{v} \cdot \mathbf{I}$ is a unit quaternion, then $s^2 + \|\mathbf{v}\|^2 = 1$. According to (2.2.25),

$$\mathbf{Q} = \begin{pmatrix} s & -\mathbf{v}^T \\ \mathbf{v} & sI_{3\times3} + C_v \end{pmatrix},$$

$$\mathbf{Q}^T = \begin{pmatrix} s & \mathbf{v}^T \\ -\mathbf{v} & sI_{3\times3} - C_v \end{pmatrix}.$$

Thus,

$$\mathbf{Q}\mathbf{Q}^T = \begin{pmatrix} s^2 + \|\mathbf{v}\|^2 & 0 \\ 0 & \mathbf{v}\mathbf{v}^T + s^2 I_{3\times 3} - C_v C_v \end{pmatrix}.$$

It follows from

$$C_v C_v = \mathbf{v}\mathbf{v}^T - \|\mathbf{v}\|^2 I_{3\times 3}$$

that

$$\mathbf{Q}\mathbf{Q}^T = \begin{pmatrix} s^2 + \|\mathbf{v}\|^2 & 0 \\ 0 & (s^2 + \|\mathbf{v}\|^2) I_{3\times 3} \end{pmatrix} = (s^2 + \|\mathbf{v}\|^2) I_{4\times 4} = I_{4\times 4}.$$

In addition,

$$\begin{aligned} det(\mathbf{Q}) &= det \begin{pmatrix} s & -v_x & -v_y & -v_z \\ v_x & s & -v_z & v_y \\ v_y & v_z & s & -v_x \\ v_z & -v_y & v_x & s \end{pmatrix} \\ &= s \times det \begin{pmatrix} s & -v_z & v_y \\ v_z & s & -v_x \\ -v_y & v_x & s \end{pmatrix} + v_x \times det \begin{pmatrix} v_x & -v_z & v_y \\ v_y & s & -v_x \\ v_z & v_x & s \end{pmatrix} \\ &\quad -v_y \times det \begin{pmatrix} v_x & s & v_y \\ v_y & v_z & -v_x \\ v_z & -v_y & s \end{pmatrix} + v_z \times det \begin{pmatrix} v_x & s & -v_z \\ v_y & v_z & s \\ v_z & -v_y & v_x \end{pmatrix}. \\ &= s^4 + s^2(v_x^2 + v_y^2 + v_z^2) + v_x^4 + v_x^2(s^2 + v_y^2 + v_z^2) \\ &\quad + v_y^4 + v_y^2(s^2 + v_x^2 + v_z^2) + v_z^4 + v_z^2(s^2 + v_x^2 + v_y^2) \end{aligned}$$

$$= s^4 + s^2(1-s^2) + v_x^4 + v_x^2(1-v_x^2) + v_y^4 + v_y^2(1-v_y^2) + v_z^4 + v_z^2(1-v_z^2)$$
$$= s^2 + v_x^2 + v_y^2 + v_z^2 = 1.$$

□

Similarly, for each quaternion $\mathbf{q} = s + \mathbf{v} \cdot \mathbf{I}$, let

$$\bar{\mathbf{Q}} = \begin{pmatrix} s & \mathbf{v}^T \\ -\mathbf{v} & sI_{3\times 3} + C_v \end{pmatrix}. \tag{2.2.33}$$

It is clear that it has similar properties to $\mathbf{Q}$, defined in (2.2.25). When $\mathbf{q}$ is unit, $\bar{\mathbf{Q}}$ is also a rotation in four dimensional space, that is, we also have

$$\bar{\mathbf{Q}}\bar{\mathbf{Q}}^T = I \quad \text{and} \quad det(\bar{\mathbf{Q}}) = 1. \tag{2.2.34}$$

Quaternions and rotations in 3D space

In the previous section, we saw that any unit quaternion corresponds to two rotational matrices in four dimensional space. In this section, we discuss how a quaternion is related to an ordinary rotation matrix in three dimensional space. There are several ways to establish the link between unit quaternions and rotations. Here we provide a direct and natural approach using the matrix representation. The information about quaternions and other methods to establish the correspondence between a quaternion and a rotation can be found in [3] [104] [61] [105].

For a unit quaternion $\mathbf{q} = s + \mathbf{v} \cdot \mathbf{I}$, let $\mathbf{Q}$ and $\bar{\mathbf{Q}}$ be defined by (2.2.25) and (2.2.33). It follows that both $\mathbf{Q}$ and $\bar{\mathbf{Q}}$ are 4×4 rotation matrices and therefore that $\mathbf{Q}\bar{\mathbf{Q}}$ is a

4×4 rotation matrix. However,

$$\begin{aligned}
\mathbf{Q}\bar{\mathbf{Q}} &= \begin{pmatrix} s & -\mathbf{v}^T \\ \mathbf{v} & sI_{3\times3} + C_v \end{pmatrix} \begin{pmatrix} s & \mathbf{v}^T \\ -\mathbf{v} & sI_{3\times3} + C_v \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ 0 & \mathbf{v}\mathbf{v}^T + (sI_{3\times3} + C_v)^2 \end{pmatrix}. \qquad (2.2.35)
\end{aligned}$$

Therefore, the 3×3 matrix $\mathbf{v}\mathbf{v}^T + (sI_{3\times3} + C_v)^2$ must be a rotation matrix, of the following form:

$$\begin{aligned}
R(\mathbf{q}) &= \mathbf{v}\mathbf{v}^T + (sI_{3\times3} + C_v)^2 \\
&= (2s^2 - 1)I_{3\times3} + 2(sC_v + \mathbf{v}\mathbf{v}^T) \\
&= \begin{pmatrix} s^2 + v_x^2 - v_y^2 - v_z^2 & 2(v_xv_y - sv_z) & 2(v_xv_z + sv_y) \\ 2(v_xv_y + sv_z) & s^2 - v_x^2 + v_y^2 - v_z^2 & 2(v_yv_z - sv_x) \\ 2(v_xv_z - sv_y) & 2(v_yv_z + sv_x) & s^2 - v_x^2 - v_y^2 + v_z^2 \end{pmatrix} (2.2.36)
\end{aligned}$$

Now we see that any quaternion corresponds to a rotation that has a matrix representation given by (2.2.36).

Conversely, for any given rotation R, there always exists a quaternion $\mathbf{q}$, such that $R(\mathbf{q}) = R$. Let

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

be an arbitrary rotation matrix and let $\mathbf{q} = s + v_x\mathbf{i} + v_y\mathbf{j} + v_z\mathbf{k}$ be the unit quaternion corresponding to the rotation matrix R. Then from (2.2.36), we should have

$$4s^2 - 1 = tr(R), \qquad (2.2.37)$$

$$R(\mathbf{q}) - R(\mathbf{q})^T = R - R^T. \qquad (2.2.38)$$

From (2.2.37), the value of s can be obtained either as $\frac{\sqrt{tr(R)+1}}{2}$ or as $-\frac{\sqrt{tr(R)+1}}{2}$. When $tr(R) \neq -1$, and $s \neq 0$, the values of v_x, v_y, v_z can be obtained directly from (2.2.38) as

$$v_x = \frac{r_{32} - r_{23}}{4s}, \qquad v_y = \frac{r_{13} - r_{31}}{4s}, \qquad v_z = \frac{r_{21} - r_{12}}{4s}. \tag{2.2.39}$$

When $tr(R) = -1$, $s = 0$ and $\mathbf{q}$ will have the form $\mathbf{v} \cdot \mathbf{I}$ and its corresponding rotation matrix given in (2.2.36) can be written in the form $R(\mathbf{q}) = 2\mathbf{v}\mathbf{v}^T - I$. On the other hand, it can be shown that rotation R can also be written in form as $2\mathbf{n}\mathbf{n}^T - I$ when $tr(R) = -1$, where $\mathbf{n}$ is a unit vector corresponding to the rotation axis. In fact, if R is a rotation, it always has a unit eigenvalue and an associated unit eigenvector $\mathbf{n}$. Let the other two eigenvalues be λ_1 and λ_2, then

$$1 + \lambda_1 + \lambda_2 = tr(R) = -1 \qquad \lambda_1\lambda_2 = det(R) = 1.$$

This implies that $\lambda_1 = -1$ and $\lambda_2 = -1$. Let ξ, η be the eigenvectors corresponding to λ_1 and λ_2 respectively, and let 3×3 matrix $P = (\mathbf{n}, \xi, \eta)$ be formed with unit column vectors $\mathbf{n}, \xi, \eta$, then we have

$$R = P \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} P^T = P\,(2E - I)\,P^T = 2\mathbf{n}\mathbf{n}^T - I,$$

where

$$E = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Thus, to find $\mathbf{v}$, we need only compute the rotation axis $\mathbf{n}$ from R by solving the equation $(R - I)\mathbf{n} = \mathbf{0}$.

The unit quaternion that corresponds to the given rotation R can also be given in terms of a rotation axis $\mathbf{n}$ and a rotation angle θ, as $\mathbf{q} = (\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\mathbf{n})$. In fact, for a unit quaternion $\mathbf{q} = (s, \mathbf{v})$, we can write $s = \cos\phi$ and $\mathbf{v} = \sin\phi\ \mathbf{u}$ for a real number ϕ and a unit 3D vector $\mathbf{u} = (\alpha, \beta, \gamma)$, since $s^2 + \|\mathbf{v}\|^2 = 1$. Let U be defined from $\mathbf{u}$ similar to C_v from vector $\mathbf{v}$. From (2.2.35), The rotation from $\mathbf{q}$ is

$$\begin{aligned} \mathbf{v}\mathbf{v}^T + (sI + C_v)^2 &= \sin^2(\phi)\mathbf{u}\mathbf{u}^T + (\cos(\phi)I + \sin(\phi)U)^2 \\ &= \cos(2\phi)I + \sin(2\phi)U + (1 - \cos(2\phi))\mathbf{u}\mathbf{u}^T. \end{aligned}$$

Comparing this representation with the rotation represented with rotation axis and rotation angle given in (2.2.6), it can be seen that the rotation represented by a unit quaternion $\mathbf{q} = (\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\mathbf{n})$ is exactly the same as the rotation R which has rotation axis $\mathbf{n}$ and rotation angle θ.

2.3 Some properties of the rotation matrix

In the previous section, we discussed the definition and different ways to represent a rotation matrix. In this section, we will investigate some of the properties possessed by a rotation matrix. The rotation matrices considered in this section are completely general, so as to emphasize the complete generality of the discussion.

Definition 2.3.1. A $n \times n$ real matrix R is called a rotation, if $RR^T = I$ and $det(R) = 1$.

We have shown in section 2.2 that for a rotation in 3D Euclidean space there always exists a vector called a rotation axis. This result is no longer true for higher dimensional rotation matrices in general.

Lemma 2.3.1. *Let R be an orthogonal matrix, and $\lambda = a + b\mathrm{i}$ be an eigenvalue of R. Then $|\lambda| = 1$.*

Proof. Let

$$R\xi = \lambda\xi, \tag{2.3.1}$$

where ξ is the unit eigenvector associated with λ. If $\bar{\lambda}$ and $\bar{\xi}$ are the conjugates of λ and ξ respectively,

$$R\bar{\xi} = \bar{\lambda}\bar{\xi}. \tag{2.3.2}$$

Therefore,

$$1 = \bar{\xi}^T\xi = \bar{\xi}^T R^T R\xi = \bar{\lambda}\lambda\bar{\xi}^T\xi = |\lambda|^2. \tag{2.3.3}$$

□

Lemma 2.3.2. *Let R be an orthogonal matrix, and $\lambda = a+b\mathrm{i}$ be a complex eigenvalue of modulus 1 with $b \neq 0$. If vector $\xi = \mathbf{x} + \mathbf{y}\mathrm{i}$ is the eigenvector associated with λ, then $\|\mathbf{x}\| = \|\mathbf{y}\|$, and $\mathbf{x} \cdot \mathbf{y} = 0$. That is, vector $\mathbf{x}$ and $\mathbf{y}$ are perpendicular to each other and have same length.*

Proof. From

$$R\xi = \lambda\xi \tag{2.3.4}$$

and the orthogonality of R, we have

$$R^T\xi = \bar{\lambda}\xi, \tag{2.3.5}$$

where $\bar{\lambda}$ denotes the complex conjugate of λ. From equation (2.3.4), we have

$$\xi^T R\xi = \lambda\xi^T\xi = \lambda(A + B\mathrm{i}), \tag{2.3.6}$$

where $A = \|\mathbf{x}\|^2 - \|\mathbf{y}\|^2$ and $B = 2\mathbf{x}\cdot\mathbf{y}$. From equation (2.3.5), we have

$$\xi^T R^T \xi = \bar{\lambda}\xi^T\xi = \bar{\lambda}(A + B\mathbf{i}). \tag{2.3.7}$$

But $\xi^T R \xi = \xi^T R^T \xi$, and this implies that

$$(\lambda - \bar{\lambda})(A + B\mathbf{i}) = 2b\mathbf{i}(A + B\mathbf{i}) = 0.$$

Thus $A = 0, B = 0$ since $b \neq 0$. That is $\|\mathbf{x}\| = \|\mathbf{y}\|$, $\mathbf{x}\cdot\mathbf{y} = 0$. ☐

Lemma 2.3.3. *Let R be an orthogonal matrix. Then there exists an orthogonal matrix P such that*

$$P^T R P = \begin{pmatrix} \cos\theta_1 & \sin\theta_1 & & & & & & \\ -\sin\theta_1 & \cos\theta_1 & & & & & & \\ & & \ddots & & & & & \\ & & & \cos\theta_k & \sin\theta_k & & & \\ & & & -\sin\theta_k & \cos\theta_k & & & \\ & & & & & \pm 1 & & \\ & & & & & & \ddots & \\ & & & & & & & \pm 1 \end{pmatrix}. \tag{2.3.8}$$

Proof. Proceeding inductively, we need only show that for any $n\times n$ orthogonal matrix with $n > 1$, there exists an orthogonal matrix P, such that R can be either reduced to the form

$$P^T R P = \begin{pmatrix} \pm 1 & \\ & R_1 \end{pmatrix}, \tag{2.3.9}$$

or to the form

$$P^T R P = \begin{pmatrix} \cos\theta_1 & \sin\theta_1 & \\ -\sin\theta_1 & \cos\theta_1 & \\ & & R_1 \end{pmatrix}, \tag{2.3.10}$$

where R_1 is also an orthogonal matrix.

Let λ be a eigenvalue of R. From lemma 2.3.1, $|\lambda| = 1$.

(1) If λ is real, $\lambda = \pm 1$. Let ξ be the unit eigenvector associated with λ. Then ξ can be expanded into an orthogonal base $\{\xi, \xi_1, \cdots, \xi_{n-1}\}$. Let $P = (\xi, \xi_1, \cdots, \xi_{n-1})$ be the $n \times n$ matrix, then P is orthonormal when $\xi_1, \cdots, \xi_{n-1}$ are all unit vectors. For P, we have

$$P^T R P = \begin{pmatrix} \pm 1 & \\ & R_1 \end{pmatrix}. \tag{2.3.11}$$

Since R and P are both orthogonal, R_1 must also orthogonal.

(2) If λ is complex, $\lambda = \cos\theta + \sin\theta \mathbf{i}$ with $\sin\theta \neq 0$. Let $\xi + \eta\mathbf{i}$ be the eigenvector associated with λ, then ξ, η are orthogonal. With lemma 2.3.2, we can assume that both ξ and η are unit vectors.

$$\begin{aligned} R(\xi + \eta\mathbf{i}) &= (\cos\theta + \sin\theta\mathbf{i})(\xi + \eta\mathbf{i}) \\ &= (\cos\theta\xi - \sin\theta\eta) + (\sin\theta\xi + \cos\theta\eta)\mathbf{i}. \end{aligned}$$

Hence,

$$\begin{aligned} R\xi &= \cos\theta\xi - \sin\theta\eta, \\ R\eta &= \cos\theta\xi + \sin\theta\eta, \end{aligned}$$

and

$$R(\xi, \eta) = (\xi, \eta)\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}. \tag{2.3.12}$$

Let $\{\xi, \eta, \xi_1 \cdots, \xi_{n-2}\}$ be the orthonormal system expanded from ξ, η and let the

$n \times n$ matrix $P = (\xi, \eta, \xi_1 \cdots, \xi_{n-2})$, then from equation(2.3.12),

$$P^T R P = \begin{pmatrix} \cos\theta_1 & \sin\theta_1 & \\ -\sin\theta_1 & \cos\theta_1 & \\ & & R_1 \end{pmatrix}. \tag{2.3.13}$$

□

Proposition 2.3.4. *For any rotation R, suppose that $P^T R P$ has the form (2.3.8) for some orthogonal matrix P. Let c_0 be the smallest diagonal element of $P^T R P$, then $r_{ii} \geq c_0$ for any diagonal element r_{ii} of R.*

Proof. Let D denote the matrix give in (2.3.8), then $R = PDP^T$. The diagonal element r_{ii} of R can thus be written as $r_{ii} = \mathbf{v}D\mathbf{v}^T$, where $\mathbf{v} = (x_1, x_2, \cdots, x_n)$ is the i^{th} row of P. As D is a rotation, the number of diagonal elements equal to -1 in D must be even. Thus we can use $\cos\pi$ to replace -1. Without loss of generality, we assume that the smallest element of D is $\cos\theta_1$, therefore

$$\begin{aligned} \mathbf{v}D\mathbf{v}^T &= (x_1^2 + x_2^2)\cos\theta_1 + \cdots + (x_{2k-1}^2 + x_{2k}^2)\cos\theta_k + x_{2k+1}^2 + \cdots + x_n^2. \\ &\geq (x_1^2 + x_2^2 + \cdots + x_{2k-1}^2 + x_{2k}^2)\cos\theta_1 + x_{2k+1}^2 + \cdots + x_n^2. \end{aligned}$$

Setting

$$\delta^2 = x_{2k+1}^2 + \cdots + x_n^2,$$

$$x_1^2 + x_2^2 + \cdots + x_{2k-1}^2 + x_{2k}^2 = 1 - \delta^2,$$

and hence

$$r_{ii} = \mathbf{v}D\mathbf{v}^T \geq (1-\delta^2)\cos\theta_1 + \delta^2 = \cos\theta_1 + (1-\cos\theta_1)\delta^2 \geq \cos\theta_1. \tag{2.3.14}$$

□

Theorem 2.3.5. *For any $n \times n$ rotation matrix R and any its diagonal element r_{ii}, we have the inequality*

$$tr(R) - 2r_{ii} \leq n - 2. \tag{2.3.15}$$

Proof. When R is a rotation, from lemma 2.3.3, we know that -1 must occur in pairs in the diagonal of matrix in equation (2.3.8). Assuming that the number of -1s in (2.3.8) is $2m$, then

$$tr(R) = tr(P^T RP) = 2(\cos\theta_1 + \cdots + \cos\theta_k) + n - 2(m+k) - 2m. \tag{2.3.16}$$

(1) If $m \geq 1$,

$$tr(R) \leq n - 4m \leq n - 4 \leq n - 2 + 2r_{ii}$$

for any r_{ii}.

(2) If $m = 0$, let $c_0 = min\{\cos\theta_i : 1 \leq i \leq k\}$ and $c_0 = 1$ when $k = 0$. It follows from equation 2.3.16 that

$$\begin{aligned} tr(R) &= 2(\cos\theta_1 + \cdots + \cos\theta_k) + n - 2k \\ &\leq 2c_0 + n - 2 \leq n - 2 + 2r_{ii}. \end{aligned}$$

□

Remark 1. *For a 3D rotation, the above property is almost obvious. In fact, for any rotation matrix R, $tr(R) = 1 + 2\cos\theta$ for some θ and it is always less than or equal to $1 + 2r_{ii}$, where r_{ii} is any diagonal element of R.*

Remark 2. *In [63], a similar result is given by Mirsky for the absolute values of the diagonal elements of a proper rotation matrix. Our findings given in theorem 2.3.5 seem more definite. In fact, Mirsky's result can be directly inferred from the inequality*

2.3.15. For any rotation matrix $R = (r_{ij})_{n\times n}$, *let diagonal matrix* U *be defined as* $U = diag(s(r_{11}), s(r_{22}), \cdots, s(r_{nn}))$, *where* $s(r) = 1$ *if* $r \geq 0$, *else* $s(r) = -1$. *When the number of negative diagonal elements of* R *is even,* U *will be a rotation and*

$$tr(UR) = \sum_{i=1}^{n} |r_{ii}|.$$

From inequality 2.3.15, for any diagonal element $|r_{kk}|$ *of* UR, *we have*

$$\sum_{i=1}^{n} |r_{ii}| \leq n - 2 + 2|r_{kk}|.$$

When the number of negative diagonal elements of R *is odd, let* $n \times n$ *matrix* $J = diag(1, \cdots, 1, -1)$. *Then* JU *will be a rotation and*

$$tr(JUR) = \sum_{i=1}^{n-1} |r_{ii}| - |r_{nn}|.$$

From inequality 2.3.15, for diagonal element $-|r_{nn}|$ *of* JUR, *we have*

$$tr(JUR) = \sum_{i=1}^{n-1} |r_{ii}| - |r_{nn}| \leq n - 2 - 2|r_{nn}|,$$

or equivalently,

$$\sum_{i=1}^{n} |r_{ii}| \leq n - 2.$$

This is the result given by Mirsky in [63].

Now we discuss how to obtain the rotation matrix which best approximates an arbitrary square matrix.

Proposition 2.3.6. *For any real* $n \times n$ *matrix* A, *let the singular value decomposition of* A *be*

$$A = UWV^T, \tag{2.3.17}$$

where

$$UU^T = VV^T = I, \quad W = diag(\sigma_i),$$

and $\sigma_1 \geq \sigma_2 \geq \cdots \sigma_n \geq 0$. *Let* Ω *denote the set of all* $n \times n$ *rotation matrices. Then*

(1) When $det(UV^T) = 1$, *the best rotation approximation of* A *in the sense of the Frobenius norm is attained by* UV^T, *that is:*

$$\min_{R \in \Omega} \|A - R\|^2 = \|A - UV^T\|^2 = \|W - I\|^2. \tag{2.3.18}$$

(2) When $det(UV^T) = -1$, *the best rotation approximation of* A *in the sense of the Frobenius norm is attained by* UJV^T, *that is:*

$$\min_{R \in \Omega} \|A - R\|^2 = \|A - UJV^T\|^2 = \|W - J\|^2, \tag{2.3.19}$$

where

$$J = \begin{pmatrix} I_{n-1} & 0 \\ 0 & -1 \end{pmatrix} \tag{2.3.20}$$

and I_{n-1} *is the* $(n-1) \times (n-1)$ *identity matrix.*

Proof. Let W be the diagonal matrix

$$\begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{pmatrix} \tag{2.3.21}$$

with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$, $i = 1, 2, \cdots, n$.

(1)When $det(UV^T) = 1$, URV^T is a rotation for any rotation matrix R. In this case, we have

$$\min_{R \in \Omega} \|A - R\|^2 = \min_{R \in \Omega} \|W - R\|^2.$$

For any rotation matrix R, it is immediately evident that

$$\|W-R\|^2 = n + \sum_{i=1}^{n} \sigma_i^2 - 2\sum_{i=1}^{n} \sigma_i r_{ii}. \tag{2.3.22}$$

Since

$$RR^T = I,$$

we have $r_{ii} \le 1$, $i = 1, 2, \cdots, n$. Therefore

$$\sum \sigma_i r_{ii} \le \sum \sigma_i.$$

With this inequality,

$$\begin{aligned} \|W-R\|^2 &= n + \sum_{i=1}^{n} \sigma_i^2 - 2\sum_{i=1}^{n} \sigma_i r_{ii} \\ &\ge n + \sum_{i=1}^{n} \sigma_i^2 - 2\sum_{i=1}^{n} \sigma_i = \|W-I\|^2, \end{aligned} \tag{2.3.23}$$

and it is obvious that $\|A - UV^T\|^2 = \|W-I\|^2$ since U and V are both orthogonal.

(2)When $det(UV^T) = -1$, $URJV^T$ is a rotation for any rotation matrix R. In this case, we have

$$\min_{R\in\Omega} \|A-R\|^2 = \min_{R\in\Omega} \|WJ-R\|^2.$$

For any rotation matrix R,

$$\begin{aligned} \|WJ-R\|^2 &= n + \sum_{i=1}^{n} \sigma_i^2 - 2(\sum_{i=1}^{n-1} \sigma_i r_{ii} - \sigma_n r_{nn}) \\ &= n + \sum_{i=1}^{n} \sigma_i^2 - 2\left(\sum_{i=1}^{n-1}(\sigma_i - \sigma_n) r_{ii} + \sigma_n(\sum_{i=1}^{n-1} r_{ii} - r_{nn})\right). \end{aligned}$$

Since $\sigma_i - \sigma_n \ge 0, \sigma_i \ge 0$, $r_{ii} \le 1$, $i = 1, 2, \cdots, n$, it follows from theorem 2.3.5 that

$$\begin{aligned} \sum_{i=1}^{n-1}(\sigma_i - \sigma_n) r_{ii} + \sigma_n(\sum_{i=1}^{n-1} r_{ii} - r_{nn}) &\le \sum_{i=1}^{n-1}(\sigma_i - \sigma_n) + (n-2)\sigma_n \\ &= \sum_{i=1}^{n-1} \sigma_i - \sigma_n. \end{aligned}$$

It follows from (2.3.24) that

$$\|WJ - R\|^2 \geq n + \sum_{i=1}^{n} \sigma_i^2 - 2\sum_{i=1}^{n-1} \sigma_i + 2\sigma_n = \|W - J\|^2. \qquad (2.3.24)$$

It is obvious that $\|A - UJV^T\|^2 = \|W - J\|^2$ since U and V are both orthogonal. $\square$

As a corollary, we show that the condition that $rank(AB^T) \geq m-1$ in the Lemma given by Uneyama in [99] can be removed.

Proposition 2.3.7. *Let A and B be $m \times n$ matrices, and R an $m \times m$ rotation matrix, and UWV^T a singular value decomposition of AB^T where $UU^T = VV^T = I$, and $W = diag(\sigma_i)$ with $\sigma_1 \geq \sigma_2 \geq \cdots \sigma_n \geq 0$. Then the minimum value of $\|A - RB\|^2$ with respect R is*

$$\min_R \|A - RB\|^2 = \begin{cases} \|A\|^2 + \|B\|^2 - 2\sum_{i=1}^{n} \sigma_i, & det(UV^T) = 1 \\ \\ \|A\|^2 + \|B\|^2 - 2(\sum_{i=1}^{n-1} \sigma_i - \sigma_n), & det(UV^T) = -1. \end{cases} \qquad (2.3.25)$$

Proof. From proposition 2.1.4,

$$\begin{aligned} \|A - RB\|^2 &= \|A\|^2 + \|B\|^2 - 2A \cdot (RB) \qquad (2.3.26) \\ &= \|A\|^2 + \|B\|^2 - 2(AB^T) \cdot R \end{aligned}$$

and

$$2(AB^T) \cdot R = \|AB^T\|^2 + n - \|AB^T - R\|^2. \qquad (2.3.27)$$

Thus, minimizing (2.3.26) is equivalent to minimizing $\|AB^T - R\|^2$. From proposition 2.3.6,

$$\min_R \|AB^T - R\|^2 = \begin{cases} \|W - I\|^2, & det(UV^T) = 1 \\ \\ \|W - J\|^2, & det(UV^T) = -1. \end{cases} \qquad (2.3.28)$$

Therefore,

$$\max_R (AB^T) \cdot R = \begin{cases} \sum_{i=1}^{n} \sigma_i, & det(UV^T) = 1 \\ \\ \sum_{i=1}^{n-1} \sigma_i - \sigma_n, & det(UV^T) = -1, \end{cases} \tag{2.3.29}$$

since $\|W\|^2 = \|AB^T\|^2$. Thus

$$\min_R \|A - RB\|^2 = \begin{cases} \|A\|^2 + \|B\|^2 - 2\sum_{i=1}^{n} \sigma_i, & det(UV^T) = 1 \\ \\ \|A\|^2 + \|B\|^2 - 2(\sum_{i=1}^{n-1} \sigma_i - \sigma_n), & det(UV^T) = -1. \end{cases} \tag{2.3.30}$$

□

Remark 3. *Note that when $detA > 0$, $det(UV^T) = 1$, when $detA < 0$, $det(UV^T) = -1$, and when $detA = 0$, $\|W - I\| = \|W - J\|$. Hence, this corollary is more general than the Lemma given in [99] since the condition that the rank$(AB^T) \geq m - 1$ has been removed.*

Proposition 2.3.8. *For two sets of $m \times 1$ vectors $A_1, A_2, \cdots, A_m$ and $B_1, B_2, \cdots, B_m$,*

$$\min_U \sum_{i=1}^{m} \|A_i - UB_i\|^2 \qquad \textit{subject to} \qquad UU^T = I$$

is equivalent to maximizing

$$\max_U U \cdot \sum_{i=1}^{n} A_i B_i^T \qquad \textit{subject to} \qquad UU^T = I. \tag{2.3.31}$$

Proof. According to the definition of the Frobenius norm of a matrix, and from proposition 2.1.3, we have

$$\begin{aligned}\sum_{i=1}^{m} \|A_i - UB_i\|^2 &= \sum_{i=1}^{n} (\|A_i\|^2 + \|B_i\|^2 - 2A_i \cdot UB_i) \\ &= \sum_{i=1}^{n} (\|A_i\|^2 + \|B_i\|^2) - 2U \cdot \sum_{i=1}^{n} A_i B_i^T) \\ &= const - 2U \cdot (\sum_{i=1}^{n} A_i B_i^T).\end{aligned}$$

□

To summarize, in this chapter a concept of matrix scalar product is introduced. This is followed by a survey on the parameterization of a rotation matrix, especially for the quaternion representation. We showed in a novel way that how a quaternion is linked with a 3D rotation and obtained a more compact form of representation of a rotation matrix with respect to a unit quaternion. For the properties of the rotation matrix, we showed that the trace of a rotation matrix can not be larger than $n - 2 + 2r_{ii}$ for any diagonal element r_{ii} of a rotation matrix. With this property, the condition that the $rank(AB^T) \geq m - 1$ in the Lemma given in [99] has been removed.

Chapter 3

Coordinate system alignment using reference points

3.1 Introduction

In a computer assisted surgery system, aligning a virtual computer image with a patient in the operating theatre is one of the key problems which needs to be solved. One common technique in establishing such an alignment is the use of fiducial markers. A fiducial marker is a kind of tiny object inserted onto the surgical surface in advance of operation, so that its position is available to both preoperative data and intraoperative data. Therefore, a precise correspondence between preoperative data and intraoperative data can be established by using fiducial markers. However, due to measurement errors, such a correspondence is not exact, and a mathematical algorithm is then needed to estimate the unknown rigid transformation that links the two data sets. More precisely, this problem can be stated as follows:

Let $\{P_n\}_{n=1}^N$ be a set of 3D points presented in both coordinate systems C_{pre} and C_{intra} respectively. Let $\{X_n\}_{n=1}^N$, $\{Y_n\}_{n=1}^N$ denote the corresponding measured positions of these points for the two coordinate systems. Let F be the unknown rigid transformation that links the two coordinate systems. We wish to estimate F by minimizing the sum

$$\Delta = \sum_{n=1}^{N} \rho(Y_n, FX_n), \tag{3.1.1}$$

where ρ denotes some measure of closeness between the two sets of coordinates. $\Delta = 0$ if there are no experiment errors.

To estimate the unknown transformation F, it is first necessary to define ρ to establish a criterion to measure the closeness between the two data sets. It is obvious that different criteria will lead to different algorithms. The commonly used criterion is the LSE(Least Square Estimate), by which F is estimated by minimizing the sum:

$$\sum_{n=1}^{N} \|FX_n - Y_n\|^2. \tag{3.1.2}$$

where $\|\cdot\|$ denotes the Euclidean distance between two points. Algorithms like the one developed with quaternion theory [39](called the Quaternion algorithm) and the one realized with the singular value decomposition technique [4](called the SVD algorithm) are based on this idea of optimization.

As a rigid transformation, F is a combination of a translation T and a rotation R, i.e., $FX = RX + T$. With least squares estimation, let

$$\mathcal{F}(R,T) = \sum_{n=1}^{N} \|RX_n + T - Y_n\|^2. \tag{3.1.3}$$

By setting

$$\frac{\partial \mathcal{F}(R,T)}{\partial T} = 0, \tag{3.1.4}$$

the optimal translation T can be estimated from

$$T = \bar{Y} - R\bar{X}, \tag{3.1.5}$$

where $\bar{X} = \frac{1}{n}\sum_{n=1}^{N} X_n$, $\bar{Y} = \frac{1}{n}\sum_{n=1}^{N} Y_n$ are the centroids of data set $\{X_n\}_{n=1}^{N}$ and data $\{Y_n\}_{n=1}^{N}$ respectively. This means that once the rotation R is estimated, the translation can be obtained by equation (3.1.5) immediately. Now let

$$\bar{X}_n = X_n - \bar{X}, \tag{3.1.6}$$

$$\bar{Y}_n = Y_n - \bar{Y}. \tag{3.1.7}$$

The sum in (3.1.3) can now be written as

$$\sum_{n=1}^{N} \|R\bar{X}_n - \bar{Y}_n\|^2. \tag{3.1.8}$$

From proposition 2.3.8, minimizing (3.1.8) is equivalent to maximizing

$$R \cdot \sum_{n=1}^{N} \bar{Y}_n \bar{X}_n^T. \tag{3.1.9}$$

Another way of estimating the rotation matrix can be considered by observing that the effect of translation can be removed by viewing each set of coordinates as a polyhedron. For each pair of points P_1, P_2 in space, a directed line segment, from P_1 to P_2, can be defined. Let E and E' be a pair of corresponding line segments in the two coordinate systems respectively, then the rotation that links the orientation of the two coordinate systems should match orientation of line segment E in one coordinate system with that of line segment E' in another as closely as possible. With this idea, a strategy based on edge-matching can be developed. As we are only interested in how two directed line segments differ in orientation, the position information for the two line segments will not be considered. One obvious choice is to use the scalar product of the two directed line segments to measure the closeness in orientation of the two line segmentation, that is, $E \cdot E'$. The closer the two edges in orientation, the larger the scalar product. Therefore the transformation can be estimated by maximizing the sum

$$\sum_{n=1}^{M} RE_m \cdot E'_m = R \cdot \sum_{n=1}^{M} E'_m E_m^T. \tag{3.1.10}$$

where $\{E_m\}_{m=1}^{M}$, $\{E'_m\}_{m=1}^{M}$ denote all possible edges of the object, and M is the number of edges in the two systems.

The closeness in orientation of two edges can also be measured by the angle between the edges. It can be seen that the closer the two edges in orientation, the smaller the angle between them is, and thus the bigger the cosine value will be. With this in mind, the rotation transformation can be estimated by maximizing the sum of cosine values of angles between each pair of edges. That is, rotation matrix R can be estimated by maximizing the sum:

$$\sum_{m=1}^{M} \cos\theta_m, \tag{3.1.11}$$

where θ_m represents the angle between m-th pair of edges RE_m and E'_m. As

$$\cos\theta_m = R\bar{E}_m \cdot \bar{E}'_m,$$

where $\bar{E}_m$ and $\bar{E}'_m$ represent the normalized edges of E_m and E'_m, the sum (3.1.11) can be written as

$$\sum_{n=1}^{M} R\bar{E}_m \cdot \bar{E}'_m = R \cdot \sum_{n=1}^{M} \bar{E}'_m \bar{E}_m^T. \tag{3.1.12}$$

In the final part of this section, we would like to point out that sum (3.1.9) can also be obtained from the idea of edge matching. In fact, (3.1.9) can be put in the form

$$\sum_{n=1}^{N} \bar{Y}_n \cdot (R\bar{X}_n). \tag{3.1.13}$$

If we regard each data set $\{X_n\}_{n=1}^N$ and $\{Y_n\}_{n=1}^N$ as a set of directed line segments starting from their centroids, then estimating R by maximizing (3.1.9) is just to find the rotation R such that all corresponding line segments $\bar{Y}_n$ and $R\bar{X}_n$ are as close as possible.

3.2 Least squares estimate of the rotation

In this section, we discuss how to compute the optimal rigid transformation based on the criterion of LSE. In the previous section, we saw that the optimal solution for translation can be obtained directly from the centroids of the two corresponding sets of coordinates when the rotation has been estimated. Thus the main problem now remaining is how to estimate rotation. As has been shown in (3.1.9), (3.1.10) and (3.1.12), the least squares estimate of rotation R can be obtained by maximizing a matrix scalar product

$$R \cdot A \tag{3.2.1}$$

with the rotation matrix R for a known matrix A. In practice, the matrix A can be obtained from (3.1.9) as $\sum_{n=1}^N \bar{Y}_n \cdot \bar{X}_n^T$, or from (3.1.10) as $\sum_{n=1}^M E'_m E_m^T$, or from (3.1.12) as $\sum_{n=1}^M \bar{E}'_m \bar{E}_m^T$. In the following discussion, we will focus on how to obtain the maximum solution for rotation matrix R from $R \cdot A$, assuming that the matrix A is known. As can be seen from the method of reasoning, the SVD algorithm given by

Arun, et al. in 1987 [4] and later improved by Uneyama in 1991 [99], the quaternion algorithm given by Horn in 1987 [39], and the polar decomposition algorithm given by Horn et al. in 1988 [40] can all be developed in a much simpler way with the exception of the polar decomposition algorithm given by Farrell and Stuelpnagel in 1966 [29]. Although a similar discussion has been given in the work of Kanatani[44], he used the concept of an infinitesimally small rotation, which makes his method of inference complicated.

3.2.1 The SVD algorithm

Let A_i and R_i denote the i^{th} column of A and R respectively, then (3.2.1) can be written in the form:

$$\begin{aligned} \Delta &= A_1 \cdot R_1 + A_2 \cdot R_2 + A_3 \cdot R_3 \\ &= \|A_1\| \cos\theta_1 + \|A_2\| \cos\theta_2 + \|A_3\| \cos\theta_3, \end{aligned} \tag{3.2.2}$$

where $\theta_1, \theta_2, \theta_3$ have their obvious meanings. This way of representing equation (3.2.1) suggests that three orthogonal vectors R_1, R_2 and R_3 of rotation R should be chosen to be as close as possible to the vectors A_1, A_2 and A_3 respectively. This geometrical intuition immediately leads to the SVD algorithm. Now let us see how the technique of singular value decomposition is used to locate the rotation matrix based on such geometric intuition.

Let the singular value decomposition of matrix A be:

$$A = UWV^T, \tag{3.2.3}$$

where U and V are orthogonal matrices and W is a diagonal matrix with non-negative elements. From corollary 2.1.6, maximizing (3.2.2) subject to $RR^T = I$ is equivalent to minimizing $\|A-R\|^2$ subject to $RR^T = I$. From proposition 2.3.6, the best rotation approximation to matrix A can be attained from UV^T when $det(UV^T) = 1$, and from UJV^T when $det(UV^T) = -1$. Thus, to find the rotation matrix that maximizes (3.2.2), we can first carry out singular decomposition on A to find U and V in (3.2.3) after A has been calculated. Then the rotation matrix estimated will be UV^T or UJV^T depending on whether $det(UV^T) = 1$ or -1.

From our method of reasoning, it can be seen that our development is more natural and more intuitive compared with that has been given in [99]. Some discussions on this algorithm can be found in [4] and [99]. Note that the matrix

$$A = \sum_{i=1}^{N} \bar{Y}_n \bar{X}_n^T$$

in (3.1.9) is the transpose of the corresponding matrix in [4], if we ignore whether matrix A is calculated from $\{X_n - \overline{X}\}$ and $\{Y_n - \overline{Y}\}$ or from $\{E_m\}$ and $\{E'_m\}$. Therefore, the orthogonal matrix estimated in [4] is represented as VU^T rather than UV^T.

3.2.2 The polar decomposition algorithm

A square matrix A can also be decomposed into an orthogonal matrix P with a symmetric semi-positive definite matrix S such that $A = PS$ or $A = SP$. This kind of matrix decomposition is called polar decomposition [88]. The algorithm for estimating a rotation matrix by minimizing the sum of squares (3.1.8) using the matrix polar decomposition technique has long been established by Farrell and Stuelpnagel in 1966 [29], where the algorithm has been used to estimate satellite attitude. This is an elegant closed-form solution. In [40], Horn et al. re-developed the algorithm specifically for 3D rotation. However, Horn's polar-decomposition algorithm can only guarantee an orthogonal matrix and does not always return a proper rotation matrix. This algorithm is improved by Kanatani in [44] by using the concept of infinitesimally small rotations. In fact, this algorithm can be obtained very simply as can be seen from [29] or from our following inference.

Let

$$A = PS \tag{3.2.4}$$

be a polar decomposition of matrix A, where P is orthogonal and S is symmetric and semi-positive definite. For matrix S, let $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ be its eigenvectors corresponding to eigenvalues $\sigma_1, \sigma_2, \sigma_3$ respectively. Since S is symmetric and semi-positive definite, all its eigenvalues are real and non-negative. We assume that $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$. Now let $V = (\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ and let W be the diagonal matrix with diagonal elements $\sigma_1, \sigma_2, \sigma_3$, then

$$S = VWV^T. \tag{3.2.5}$$

Combining equation (3.2.4) and equation (3.2.5), we have

$$A = PVWV^T. \tag{3.2.6}$$

Let $U = PV$, then U is orthogonal and thus (3.2.6) is a singular value decomposition of matrix A. With the result we obtained from the SVD algorithm in section 3.2.1, we know that the rotation maximizing $R \cdot A$ will be

$$R = PVV^T = P \tag{3.2.7}$$

when $det(P) = 1$, and

$$R = PVJV^T \tag{3.2.8}$$

when $det(P) = -1$.

Since

$$VJV^T = I - 2\mathbf{v}_3\mathbf{v}_3^T, \tag{3.2.9}$$

(3.2.8) can be further written as

$$R = P(I - 2\mathbf{v}_3\mathbf{v}_3^T). \tag{3.2.10}$$

Solution (3.2.8) and (3.2.10) obtained in different cases can be put together as given in [44]:

$$R = P(I + (det(P) - 1)\mathbf{v}_3\mathbf{v}_3^T). \tag{3.2.11}$$

If the polar decomposition of A is given in the form $A = SP$, as with the above discussion, the corresponding solution will be in the form

$$R = (I + (det(P) - 1)\mathbf{v}_3\mathbf{v}_3^T)P. \tag{3.2.12}$$

This gives the polar decomposition algorithm. To estimate the rotation R that maximizes $R \cdot A$ under polar decomposition, we need first to compute the polar decomposition of matrix $A = PS$(or $A = SP$). If P is already a rotation, P will be the solution; if P is not a proper rotation, we need further to modify the orthonormal matrix P with $R = P(I - 2\mathbf{v}\mathbf{v}^T)$ (or $R = (I - 2\mathbf{v}\mathbf{v}^T)P$) where $\mathbf{v}$ is a unit vector of S associated with the smallest eigenvalue.

The methods of computing the polar decomposition of a matrix can be found in [40] and [88].

Remark 4. *As have been seen, the polar decomposition algorithm has been given as a corollary of the SVD algorithm. Conversely, the SVD algorithm can also be regarded as a corollary of the polar decomposition algorithm. In fact, let*

$$A = UWV^T$$

be the singular value decomposition of matrix A, where U, V are orthogonal and $W = diag(\sigma_1, \sigma_2, \sigma_3)$ with $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$. Let

$$P = UV^T, \qquad S = VWV^T.$$

Then S is symmetric and semi-positive definite and P is orthogonal. Therefore

$$A = PS$$

is a polar decomposition of matrix A. According to the polar decomposition algorithm, the rotation matrix R maximizing $R \cdot A$ is $P = UV^T$ when P is a proper rotation, and is $P(I - 2\mathbf{v}_3\mathbf{v}_3) = UV^T(VJV^T) = UJV^T$ when P is not a proper rotation. This gives the SVD algorithm.

3.2.3 The Quaternion algorithm

From section 2.2.5, the rotation matrix R in (3.2.1) can be further represented by a unit quaternion(the components of a unit quaternion are also called Euler-Rodrigues parameters). Since the elements in the rotation matrix represented by a quaternion are all quadratic forms, it follows that the matrix scalar product $R \cdot A$ in (3.2.1) is actually a quadratic form and the maximization problem can be solved using an eigen-technique. More specifically, let $\mathbf{q} = (q_0, q_1, q_2, q_3)^T$ be the unit quaternion corresponding rotation matrix R, then from (2.2.36), rotation R can be given as:

$$R = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix} \qquad (3.2.13)$$

with $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$.

Set

$$\mathbf{S} = \begin{pmatrix} a_{11}+a_{22}+a_{33} & a_{32}-a_{23} & a_{13}-a_{31} & a_{21}-a_{12} \\ a_{32}-a_{23} & a_{11}-a_{22}-a_{33} & a_{12}+a_{21} & a_{13}+a_{31} \\ a_{13}-a_{31} & a_{12}+a_{21} & -a_{11}+a_{22}-a_{33} & a_{23}+a_{32} \\ a_{21}-a_{12} & a_{13}+a_{31} & a_{23}+a_{32} & -a_{11}-a_{22}+a_{33} \end{pmatrix}, \tag{3.2.14}$$

where $\{a_{ij}\}$ are the elements of matrix A obtained in (3.2.1).

From (3.2.1) and (3.2.13),

$$\begin{aligned} R \cdot A &= \textstyle\sum_{i,j=1}^{3} a_{ij} r_{ij} \\ &= a_{11}(q_0^2+q_1^2-q_2^2-q_3^2) + 2a_{12}(q_1q_2-q_0q_3) + 2a_{13}(q_1q_3+q_0q_2) \\ &+2a_{21}(q_1q_2+q_0q_3) + a_{22}(q_0^2-q_1^2+q_2^2-q_3^2) + 2a_{23}(q_2q_3-q_0q_1) \\ &+2a_{31}(q_1q_3-q_0q_2) + 2a_{32}(q_2q_3+q_0q_1) + a_{33}(q_0^2-q_1^2-q_2^2+q_3^2) \\ &= (a_{11}+a_{22}+a_{33})q_0^2 + (a_{11}-a_{22}-a_{33})q_1^2 \qquad (3.2.15) \\ &+(-a_{11}+a_{22}-a_{33})q_2^2 + (-a_{11}-a_{22}+a_{33})q_3^2 \\ &+2(a_{32}-a_{23})q_0q_1 + 2(a_{13}-a_{31})q_0q_2 + 2(a_{21}-a_{12})q_0q_3 \\ &+2(a_{12}+a_{21})q_1q_2 + 2(a_{13}+a_{31})q_1q_3 + 2(a_{23}+a_{32})q_2q_3 \\ &= \mathbf{q}^T\mathbf{S}\mathbf{q}. \end{aligned}$$

where $\mathbf{q} = (q_0, q_1, q_2, q_3)^T$ is the unit quaternion corresponding to the unknown rotation R.

It is known that the unit eigenvector of $\mathbf{S}$ corresponding to the largest eigenvalue maximizes $R \cdot A$ [91]. This result leads to the quaternion algorithm. With this algorithm, to compute the rotation matrix that maximizes $R \cdot A$, we need only construct a matrix $\mathbf{S}$ using the elements of matrix A in (3.2.1), and then compute the unit eigenvector $\mathbf{q} = (q_0, q_1, q_2, q_3)^T$ associated with the largest eigenvalue of $\mathbf{S}$. The rotation estimated will be the one corresponding to $\mathbf{q}$.

The quaternion algorithm has been discussed by Horn [39] in great detail. As can be seen from the above discussion, the algorithm is simple and direct and only involves the computation of the eigenvector associated with the largest eigenvalue of a 4×4 matrix.

3.2.4 The algorithm based on estimation of the rotation axis

The quaternion algorithm is developed by representing a rotation by a unit quaternion. In this section, we will develop another algorithm by representing the rotation matrix in terms of rotation axis and a rotation angle. From section 2.2.2 or section 2.2.3, rotation can be represented by a rotation axis $\mathbf{n}$ and a rotation angle θ as

$$R = \cos\theta I + (1-\cos\theta)\mathbf{n}\mathbf{n}^T + \sin\theta N. \tag{3.2.16}$$

Thus

$$\begin{aligned} R \cdot A &\doteq \cos\theta I \cdot A + (1-\cos\theta)(\mathbf{n}\mathbf{n}^T) \cdot A + \sin\theta N \cdot A \qquad (3.2.17) \\ &= (\mathbf{n}\mathbf{n}^T) \cdot A + \sin\theta N \cdot A + \cos\theta \left(tr(A) - (\mathbf{n}\mathbf{n}^T) \cdot A \right). \end{aligned}$$

Let

$$\delta_1 = N \cdot A,$$

$$\delta_2 = tr(A) - (\mathbf{n}\mathbf{n}^T) \cdot A.$$

Once the rotation axis is known, it follows from equation (3.2.17) that $R \cdot A$ can be maximized by choosing the rotation angle θ satisfying

$$\sin\theta = \frac{\delta_1}{\sqrt{\delta_1^2 + \delta_2^2}} \qquad \cos\theta = \frac{\delta_2}{\sqrt{\delta_1^2 + \delta_2^2}} \tag{3.2.18}$$

In this case, $R \cdot A$ can reach its maximum value

$$\delta(\mathbf{n}) = (\mathbf{n}\mathbf{n}^T) \cdot A + \sqrt{\delta_1^2 + \delta_2^2}.$$

However, it is not that direct to compute the rotation axis $\mathbf{n}$ that maximizes $\delta(\mathbf{n})$. A numerical method could be used to find the optimal solution, but at the expense of computational speed and accuracy. Here we provide a technique for estimating the rotation axis from matrix A directly by solving a simple eigen system. Let

$$A = SP$$

be a polar decomposition of A, where S is symmetric and semi-positive definite and P is orthogonal. When A is nonsingular, P is uniquely defined. When A is singular, P is not unique, but it can be taken to be a proper rotation. From section 3.2.2, the rotation that maximizes $R \cdot A$ is P when P is a rotation, and is $(I - 2\mathbf{v}\mathbf{v}^T)P$ when P is not a proper rotation, where $\mathbf{v}$ is a unit eigenvector of S associated with the smallest eigenvalue. Thus matrix A can be decomposed in the following way:

$$A = S_1 R,$$

where R is a rotation and S_1 is either S or $S(I - 2\mathbf{v}\mathbf{v}^T)$. Suppose $\mathbf{n}$ is the rotation axis of R, then, we must have

$$A\mathbf{n} = S_1\mathbf{n},$$

or

$$(A - S_1)\mathbf{n} = 0,$$

which can be easily solved. Since $det(A - S_1) = det(S_1)det(R - I) = 0$, $\mathbf{n}$ is actually the eigenvector of $A - S_1$ corresponding to eigenvalue 0.

Based on this idea, we obtain the following algorithm (called 'the rotation axis based algorithm').

Algorithm 3.2.1. (1) Compute the eigenvalues and eigenvectors of matrix AA^T such that $AA^T = P\Lambda P^T$, where P is the matrix consisting of eigenvectors and Λ the diagonal matrix with elements equal to the eigenvalues. Find the eigenvector $\mathbf{v}$ of AA^T associated with the smallest eigenvalue.

(2) Let $\Lambda^{1/2}$ be the matrix whose elements are square roots of matrix Λ, and let $S = P\Lambda^{1/2}P^T$.

- if $det(A) \geq 0$, set $S_1 = S$, and compute unit vector $\mathbf{n}$ from $(A - S_1)\mathbf{n} = 0$;
- else if $det(A) < 0$, set $S_1 = S(I - 2\mathbf{v}\mathbf{v}^T)$, and compute unit vector $\mathbf{n}$ from $(A - S_1)\mathbf{n} = 0$;

(3) Compute the $\cos\theta, \sin\theta$ from equation (3.2.18) and construct the rotation matrix with equation (3.2.16).

As can be seen from the experimental results shown in the next section, the above algorithm is equivalent to the SVD algorithm and the quaternion algorithm. However, this algorithm only involves computing the eigenvalues and vectors of a 3×3 symmetric matrix, which is more stable than computing the singular decomposition.

Remark 5. *The rotation axis can also be estimated by observing that the difference vector between each corresponding pair of coordinates is perpendicular to the rotation axis once the effect of translation has been removed from the data sets. Thus, rotation axis can be estimated by minimizing the sum*

$$\Upsilon = \sum_{n=1}^{N}((Y_n - X_n)\cdot \mathbf{n})^2 = \mathbf{n}\left(\sum_{n=1}^{N}(Y_n - X_n)(Y_n - X_n)^T\right)\mathbf{n}, \tag{3.2.19}$$

assuming that the data have been moved to their centroids. Experiments have shown that the rotation axis estimated with this method is much less accurate than that computed by the algorithm presented earlier.

3.2.5 The Affine approximation and the iterative algorithms

The rotation matrix R can also be estimated by first finding an affine approximation and then using the singular value decomposition technique to find the closest rotation matrix to the affine approximation.

In addition, iterative optimization procedure can be considered. As correspondences between data sets are known, a good initial estimate can be obtained easily either from geometric intuition or from algebraic equations. Therefore, the rotation matrix R can be estimated accurately with conventional minimization procedures.

Compared with the four closed form solutions described above, these two methods are less computationally efficient in general and are no longer much used in application.

3.3 Experimental Results

All the algorithms given in this chapter have been tested both with randomly generated data and real data measured with the OPMS (Optical Position Measurement System) and CMM (Coordinate Measuring Machine). For randomly generated data sets, a set of random points is first generated uniformly within a cuboid with size of 100 by 100 by 100. To obtain the corresponding data set, it is then transformed by a rigid transformation. Both data sets are further modified by adding a normally distributed random error $N(0, \sigma^2)$ with standard deviation σ in the range $[0, 1]$. The algorithm developed with SVD, polar decomposition, unit quaternion and the algorithm based on rotation axis estimation are then applied to the data sets in turn. The estimated rotation is then compared with the true rotation by calculating the Frobenius norm of the difference matrix between the two matrices in each case. The results of the experiments are summarized in Tables 3.1 to 3.6. It can be seen from these tables that the performances of the Quaternion algorithm, the SVD algorithm and the algorithm based on the estimate of the rotation axis given in section 3.2.4 work almost equally well. As the equivalence between the polar decomposition algorithm and the SVD algorithm is guaranteed in theory, relevant testing results are not listed. The SVD algorithm and the Quaternion algorithm have also been compared with the algorithm given in section 3.2.4 by testing their performances with data sets measured with the OPMS, where system measurement errors are around 1mm, and the CMM, where the system error is less then 0.05mm. The test result shows that the matching accuracy is quite satisfactory for all these methods. Real data sets are obtained by measuring a demonstration block in different positions and orientations. For each position and orientation, 12 points are sampled, among these, up to 8 points are used as reference marks and other four are used as verification points. It can be seen from the statistics listed in the following tables that the performances of all the methods are quite similar.

As far as speed is concerned, all the algorithms based on closed form solutions are quite similar. Some experiments have also been given on the algorithm based on an iterative optimization procedure. It is shown that this way of estimating the rotation matrix also performs very well. The only disadvantage of this method is its computing speed, which is much slower compared with any previous algorithm when the number of reference points becomes large.

It is also important to know how many reference points are needed to guarantee an accurate estimate. From Figure 3.1, it can be seen that with the increase in the number of reference points, the errors between the estimated rotation matrices and the true rotation matrices tend to become smaller and smaller, and the number of reference points needed really depends on the accuracy of measurements. Figure 3.2 shows that maximum errors do not change significantly when the number of reference points is larger than 10.

Method	quaternion	svd	rot-axis-Est
Max	2.44273	2.44273	2.44037
Min	0.79894	0.79895	0.79493
Mean	1.54632	1.54632	1.54630
Trmean	1.54632	1.54632	1.54630
Median	1.46744	1.46746	1.46784
Stdev	0.42589	0.42589	0.42589

Table 3.1: The summary of maximum error for OPMS data(mm)

Method	quaternion	svb	rot-axis-Est
Max	1.73574	1.73574	1.72179
Min	0.63870	0.63870	0.63177
Mean	1.08744	1.08744	1.08744
Trmean	1.08744	1.08744	1.08744
Median	1.01565	1.01565	1.00956
Stdev	0.27315	0.27315	0.27311

Table 3.2: The summary of average maximum error for OPMS data(mm)

Method	quaternion	svd	rot-axis-Est
Max	0.90948	0.90948	0.90948
Min	0.49513	0.49513	0.49477
Mean	0.68134	0.68134	0.68134
TrMean	0.68134	0.68134	0.68134
Median	0.68156	0.68156	0.68154
Stdev	0.11087	0.11087	0.11086

Table 3.3: The summary of average error for OPMS data(mm)

Method	quaternion	svd	rot-axis-Est
Max	0.18721	0.18722	0.18722
Min	0.01238	0.01238	0.01238
Mean	0.06827	0.06849	0.06849
Trmean	0.06451	0.06536	0.06536
Median	0.06047	0.06316	0.06316
Stdev	0.03639	0.03609	0.03609

Table 3.4: The summary of maximum error for CMM data(mm)

Method	quaternion	svb	rot-axis-Est
Max	0.09319	0.09319	0.09139
Min	0.01226	0.01238	0.01215
Mean	0.04311	0.04222	0.04222
Trmean	0.04192	0.04140	0.04140
Median	0.04076	0.04076	0.04076
Stdev	0.01639	0.01757	0.01757

Table 3.5: The summary of average maximum error for CMM data(mm)

Method	quaternion	svd	rot-axis-Est
Max	0.05844	0.05844	0.05844
Min	0.01016	0.00859	0.00859
Mean	0.02717	0.02688	0.0.02725
TrMean	0.02639	0.02625	0.02616
Median	0.02593	0.02593	0.02593
Stdev	0.01103	0.01142	0.01113

Table 3.6: The summary of average error for CMM data(mm)

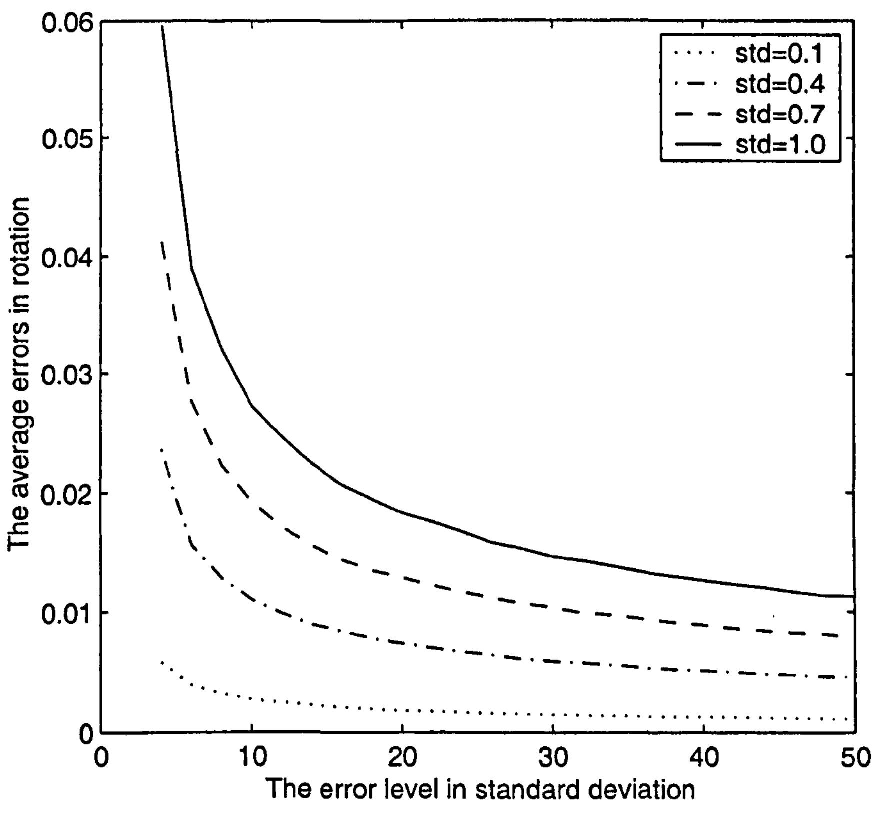

Figure 3.1: Average errors between estimated rotation and true rotation with the increase of noise level for the rotation axis based algorithm

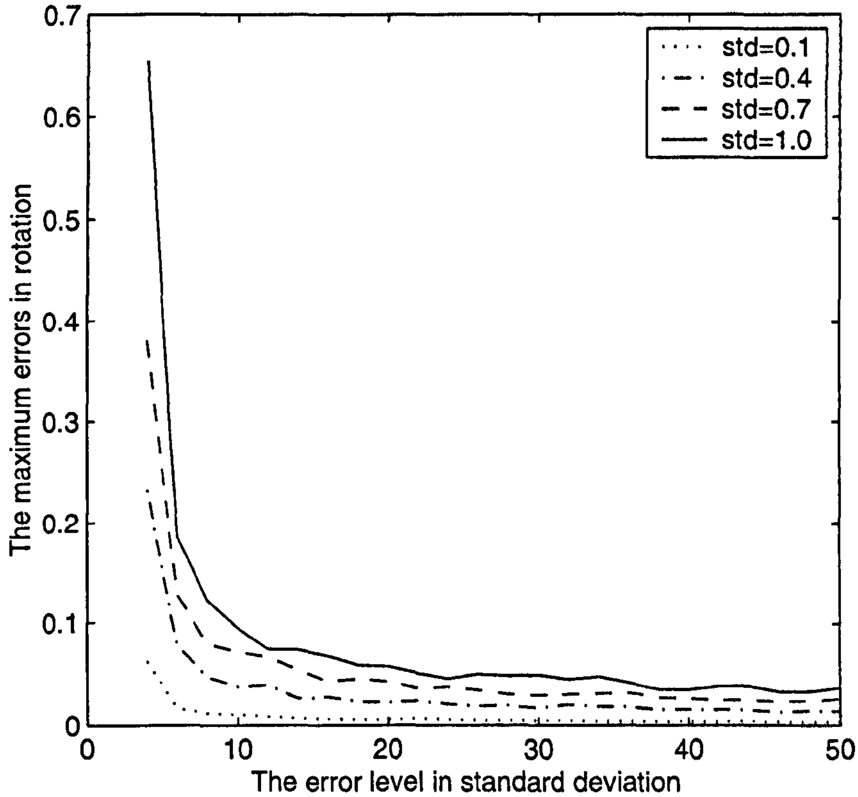

Figure 3.2: Maximum errors between estimated rotation and true rotation with the increase of noise level for the rotation axis based algorithm

Chapter 4

Coordinate system alignment using reference geometric primitives

4.1 Introduction

In order to obtain the expected registration accuracy and realize real time matching between preoperative images and patients, most computer assisted surgical systems at the movement use artificial fiducials to establish the precise matching between preoperative image and intraoperative data. Reference markers are one of the common fiducials used but other kind of fiducials are also used such as fiducial frames. In this case, the reference markers are attached on a frame rather then directly implanted onto the surface of the patients' bone. For some reference frames, the frame itself can be detected in both preoperative data and intraoperative data. For instance, the neuro-surgical system developed by Rwoh et al. [49], uses a modified stereotactic head-frame fitted with N-shaped fiducials. Cross-sections of the fiducials appear in each CT image. During operation, the head-frame is mounted at a known location with respect to the robot, providing a fixed relationship between the patient and robot. Among all the fiducial frames, a polyhedron frame like a rectangular box-shaped frame, is undoubtedly the simplest. For this kind of frame, we usually know exact correspondences at the level of the reference frame rather than at the level of reference points. That is, we know exactly on which edge a reference marker is located for each image. As the reference frame can be detected in both preoperative data and intraoperative data, it is possible to use the features of the reference frame such as the vertices or edges of the frame to align the preoperative image coordinate with intraoperative patient coordinate. The registration based on such types of fiducial can be carried out as follows. First, from the preoperative data and intraoperative data, some features of the fiducial frame such as vertices and edges can be estimated. Then the registration can be established by the technique of feature matching. In this chapter, we will focus on the simplest frames that are composed of points, straight lines and planes. Since the directions of lines and the normal directions to planes can be detected by a specially designed reference frame, the straight lines and planes dealt with in this chapter are assumed to be directed.

Suppose for a given polyhedral object, two sets of points are sampled with the object in different positions and different orientations. These two data sets can then be further divided into subsets according to the geometric features (vertices, edges and faces) of the frame. To align the two data sets, each subset is first fitted with a straight line or a plane according to whether the corresponding subset is from an edge or from a face of the object. From this procedure, two groups of geometric primitives containing points (representing frame vertices), straight lines (representing frame edges) and planes (representing frame faces) are obtained for estimating the unknown rigid transformation that links the preoperative data and the intraoperative data. This work is also the first attempt for our region to region registration problem where the region is known to consist of directed straight lines or directed planes. The matching problem for more general geometric features will be dealt with in Chapter 8. This chapter has been organized as follows. In sections 2 and 3, algorithms for matching two sets of directed lines and two sets of directed planes are discussed respectively. In section 4, we discuss how to match two geometric primitives mixed up with points, straight lines and planes. Some mathematical results used in this chapter, together with the techniques for fitting a straight lines and a plane, are put into appendices at the end of this chapter.

4.2 Matching two sets of directed lines

For a line in 3D space, the rotation of the line by an angle π will be the line itself when the rotation axis is perpendicular to the line and intersects it. Thus when no direction is specified, the rotation transformation cannot be determined properly in some cases. In this chapter, a line is defined as a pair of vectors $\mathcal{L}(P, \mathbf{v})$ in space, where vector P is a point on the line and vector $\mathbf{v}$ is unit, representing the direction of the line. Though the discussion is mainly on 3D lines, the algorithm obtained in this section can also be used in the 2D case.

4.2.1 The distance from a point to a line

Let Q be a point and $\mathcal{L}(P, \mathbf{v})$ be a line in space. The distance between the point and the line can be represented in the following way:

$$\begin{aligned} D(Q, \mathcal{L}) &= \|(Q-P) - ((Q-P)\cdot \mathbf{v})\mathbf{v}\| \\ &= \|(Q-P) - \mathbf{v}\mathbf{v}^T(Q-P)\| \\ &= \|(I - \mathbf{v}\mathbf{v}^T)(Q-P)\|, \end{aligned} \tag{4.2.1}$$

where I represents the 3×3 identity matrix.

For a unit vector $\mathbf{v} = (\alpha, \beta, \gamma)^T$, let C_v be the skew-symmetric matrix constructed from $\mathbf{v}$ in the following way,

$$C_v = \begin{pmatrix} 0 & -\gamma & \beta \\ \gamma & 0 & -\alpha \\ -\beta & \alpha & 0 \end{pmatrix}. \tag{4.2.2}$$

Then

$$C_v^2 = \mathbf{v}\mathbf{v}^T - I. \tag{4.2.3}$$

Hence, the distance between a point Q to a straight line $\mathcal{L}(P, \mathbf{v})$ can also be put into the form

$$D(Q, \mathcal{L}) = \|C_v^2(Q-P)\|. \tag{4.2.4}$$

4.2.2 The centre of a set of lines

For a set of directed straight lines $\{\mathcal{L}_i(P_i, \mathbf{v}_i)\}_{i=1}^n$, it is possible to determine the centre of these lines. The centre of the line set is defined as a point C such that the sum of squared distances from C to each line is smallest among all the points in the space, i.e.,

$$\sum_{i=1}^{n} D^2(C, \mathcal{L}_i) = \min_{P} \sum_{i=1}^{n} D^2(P, \mathcal{L}_i). \tag{4.2.5}$$

Now we investigate, for a given set of lines, how to compute their centre. Let

$$\begin{aligned} S(C) &= \sum_{i=1}^{n} D^2(C, \mathcal{L}_i) \\ &= \sum_{i=1}^{n} \|(I - \mathbf{v}_i\mathbf{v}_i^T)(C - P_i)\|^2 \\ &= \sum_{i=1}^{n} (C - P_i)^T (I - \mathbf{v}_i\mathbf{v}_i^T)^T (I - \mathbf{v}_i\mathbf{v}_i^T)(C - P_i). \end{aligned} \tag{4.2.6}$$

Since $\mathbf{v}_i$ is a unit vector, $I - \mathbf{v}_i\mathbf{v}_i^T$ is idempotent and symmetric. Thus

$$S(C) = \sum_{i=1}^{n} (C - P_i)^T (I - \mathbf{v}_i\mathbf{v}_i^T)(C - P_i). \tag{4.2.7}$$

Set

$$\frac{dS}{dC} = 2\sum_{i=1}^{n} (I - \mathbf{v}_i\mathbf{v}_i^T)(C - P_i) = 0.$$

We have

$$\left(nI - \sum_{i=1}^{n} \mathbf{v}_i\mathbf{v}_i^T\right) C = \sum_{i=1}^{n} (I - \mathbf{v}_i\mathbf{v}_i^T) P_i. \tag{4.2.8}$$

Now, if matrix

$$\Gamma = nI - \sum_{i=1}^{n} \mathbf{v}_i\mathbf{v}_i^T \tag{4.2.9}$$

is nonsingular, then the centre of the set of lines can be uniquely determined as

$$C = \left(nI - \sum_{i=1}^{n} \mathbf{v}_i\mathbf{v}_i^T\right)^{-1} \sum_{i=1}^{n} (I - \mathbf{v}_i\mathbf{v}_i^T) P_i. \tag{4.2.10}$$

It is worth knowing when a set of lines has a unique centre for our registration problem. Obviously, when all lines are parallel to each other, the centre of the set of lines cannot be uniquely determined. To know when a set of lines has a unique centre, we need only to know when the matrix Γ in (4.2.9) is nonsingular.

Proposition 4.2.1.

$$\Gamma = nI - \sum_{i=1}^{n} \mathbf{v}_i \mathbf{v}_i^T$$

will be nonsingular when there exist at least two non-parallel lines in the line set, where $\mathbf{v}_i$ *is the direction of line* $\mathcal{L}_i (i = 1, 2, \cdots, n)$.

Proof. Let $\{\mathcal{L}_i(P_i, \mathbf{v}_i)\}_{i=1}^n$ be a set of lines. Assume that there are at least two non-parallel lines in the line set $\{\mathcal{L}_i\}_{i=1}^n$. Let $\mathcal{V} = (\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n)$ be the matrix with $\mathbf{v}_i$ its i^{th} column. Then $rank(\mathcal{V}) \geq 2$. According to Proposition 4.6.2 given in Appendix C, $rank(\mathcal{V}\mathcal{V}^T) \geq 2$. Note that the matrix $\mathcal{V}\mathcal{V}^T$ is symmetric and semi-positive definite, its eigen-value must be non-negative. Suppose that the eigenvalues of $\mathcal{V}\mathcal{V}^T$ are $\lambda_1, \lambda_2, \lambda_3$. Then

$$\lambda_1 + \lambda_2 + \lambda_3 = tr(\mathcal{V}\mathcal{V}^T) = n,$$

since $\mathbf{v}_i$ is unit, $i = 1, 2, \cdots, n$.

Due to the fact that $rank(\mathcal{V}\mathcal{V}^T) \geq 2$, there exist at least two non-zero eigenvalues. Thus for all λ_i, we have $\lambda_i < n, (i = 1, 2, 3)$. This means that n cannot be the eigenvalue of matrix $\mathcal{V}\mathcal{V}^T$ and thus $nI - \sum_{i=1}^n \mathbf{v}_i \mathbf{v}_i^T$ is non-singular. In fact, let U be the orthogonal matrix such that

$$U^T(\mathcal{V}\mathcal{V}^T)U = \begin{pmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \lambda_3 \end{pmatrix},$$

then

$$\begin{aligned} det(nI - \sum_{i=1}^{n} \mathbf{v}_i \mathbf{v}_i^T) &= det(nI - \mathcal{V}\mathcal{V}^T) \\ &= det(nI - U(\mathcal{V}\mathcal{V}^T)U^T) \\ &= (n - \lambda_1)(n - \lambda_2)(n - \lambda_3) > 0. \end{aligned} \tag{4.2.11}$$

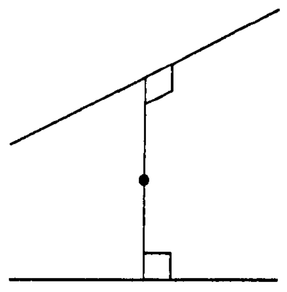

Figure 4.1: The centre of two straight lines

□

This result shows that the centre of a set of lines can be uniquely determined whenever non-parallel lines are present.

Remark 6. *When all lines $\mathcal{L}_i(P_i, \mathbf{v}_i)(i = 1, \cdots, n)$ are all mutually parallel, then $\mathbf{v}_i = \mathbf{v}_1$ for $i = 2, \cdots, n$ and equation (4.2.8) can be written as*

$$n(I - \mathbf{v}_1\mathbf{v}_1^T)C = (I - \mathbf{v}_1\mathbf{v}_1^T)\sum_{i=1}^{n} P_i.$$

The centre of this set of lines can be any point on the line $\mathcal{L}(C_0, \mathbf{v}_1)$, where C_0 is the centre of gravity of all points $\{P_i\}_{i=1}^{n}$.

Proposition 4.2.2. *The centre of a set of lines does not depend on the representation of the lines.*

Proof. It is evident that the centre is independent of the signs of line directions. We show further that it does not depend on which point the line passes through either. Suppose $(P_i, \mathbf{v}_i)$ and $(P_i', \mathbf{v}_i)$ represent the same line, $i = 1, 2, \cdots, n$. Then there exist real numbers $\alpha_i (i = 1, 2, \cdots, n)$ such that $P_i' = P_i + \alpha_i \mathbf{v}_i$. Thus

$$\begin{aligned}
\sum_{i=1}^{n}(I - \mathbf{v}_i\mathbf{v}_i^T)P_i &= \sum_{i=1}^{n}(I - \mathbf{v}_i\mathbf{v}_i^T)P_i' + \sum_{i=1}^{n}(I - \mathbf{v}_i\mathbf{v}_i^T)(P_i - P_i') \\
&= \sum_{i=1}^{n}(I - \mathbf{v}_i\mathbf{v}_i^T)P_i' + \sum_{i=1}^{n}(I - \mathbf{v}_i\mathbf{v}_i^T)(\alpha_i\mathbf{v}_i) \\
&= \sum_{i=1}^{n}(I - \mathbf{v}_i\mathbf{v}_i^T)P_i',
\end{aligned}$$

since $(I - \mathbf{v}_i\mathbf{v}_i^T)\mathbf{v}_i = 0 (i = 1, 2, \cdots, n)$. □

Proposition 4.2.3. *Let R and T be the rotation matrix and the translation vector. If C is the centre of line set $\mathcal{L}_i(P_i, \mathbf{v}_i)(i = 1, 2, \cdots, n)$, then the centre of line set $\mathcal{L}_i'(RP_i + T, R\mathbf{v}_i)$ will be $RC + T$.*

Proof. It follows from (4.2.10) that the centre of the line set $\mathcal{L}'(RP_i + T, R\mathbf{v}_i), i = 1, 2, \cdots, n$ is

$$\begin{aligned}
&\left(nI - \sum_{i=1}^{n} R\mathbf{v}_i\mathbf{v}_i^T R^T\right)^{-1} \sum_{i=1}^{n}(I - R\mathbf{v}_i\mathbf{v}_i^T R^T)(RP_i + T) \\
= \ & R\left(nI - \sum_{i=1}^{n} \mathbf{v}_i\mathbf{v}_i^T\right)^{-1} R^T \sum_{i=1}^{n} R(I - \mathbf{v}_i\mathbf{v}_i^T)R^T(RP_i + T) \\
= \ & R\left(nI - \sum_{i=1}^{n} \mathbf{v}_i\mathbf{v}_i^T\right)^{-1} \sum_{i=1}^{n}(I - \mathbf{v}_i\mathbf{v}_i^T)(P_i + R^T T) \\
= \ & RC + R\left(nI - \sum_{i=1}^{n} \mathbf{v}_i\mathbf{v}_i^T\right)^{-1} \sum_{i=1}^{n}(I - \mathbf{v}_i\mathbf{v}_i^T)R^T T \\
= \ & RC + T.
\end{aligned}$$

□

4.2.3 Method of registration

Now we consider how to register the two sets of lines. Let $\{\mathcal{L}_i(P_i, \mathbf{v}_i)\}_{i=1}^n$ and $\{\mathcal{L}'_i(Q_i, \mathbf{u}_i)\}_{i=1}^n$ be two sets of lines. It is assumed that there exist an unknown rotation R and translation T such that

$$\begin{aligned} \mathbf{u}_i &= R\mathbf{v}_i + \mu_i, \\ Q_i &= R(P_i + \alpha_i \mathbf{v}_i) + T + \epsilon_i, \end{aligned} \tag{4.2.12}$$

$i = 1, 2, \cdots, n$. where $\{\alpha_i\}_{i=1}^n$ are real numbers and $\{\mu_i\}_{i=1}^n$, $\{\epsilon_i\}_{i=1}^n$ are measurement errors.

Since the direction of a line is independent of translation during a rigid transformation, the rotation can be found directly from $\{\mathbf{v}_i\}_{i=1}^n$ and $\{\mathbf{u}_i\}_{i=1}^n$ by maximizing

$$\sum_{i=1}^n \mathbf{u}_i \cdot R\mathbf{v}_i,$$

which can be further rewritten as

$$R \cdot \sum_{i=1}^n \mathbf{u}_i \mathbf{v}_i^T. \tag{4.2.13}$$

The rotation R can thus be estimated by one of the four closed form solutions provided in Section 3.2 from the matrix

$$A = \sum_{i=1}^n \mathbf{u}_i \mathbf{v}_i^T.$$

Once the rotation has been found, the translation can be found using Proposition 4.2.3 as

$$T = C' - RC, \tag{4.2.14}$$

where C and C' are the centres of the line sets $\{\mathcal{L}_i\}_{i=1}^n$ and $\{\mathcal{L}'_i\}_{i=1}^n$ respectively.

As can be seen from the method of registration, the rotation matrix R is estimated from matrix A by maximizing the matrix scalar product $R \cdot A$. Since the matrix A is obtained from the corresponding directions of $\mathbf{v}_i$ and $\mathbf{u}_i$, changing the direction of $\mathbf{v}_i$ or $\mathbf{u}_i$ but not both will change the sign of matrix $\mathbf{u}_i\mathbf{v}_i^T$ and therefore change the matrix A. To obtain the correct rotation R, the correct correspondences in directions between the two line sets must be known in advance. When the directions of the two sets of straight lines are not known, the correspondences in directions for the two line sets need to be specified by using a carefully designed frame. Note that in some cases, the orientation of an object cannot be determined by two undirected lines or three orthogonal undirected lines, as in this case we have several ways in which to match the two undirected line sets. For example, for two undirected lines, their rotation by an angle π will have no apparent effect when the rotation axis is perpendicular to them and the extended axis of rotation passes through two lines. However, there are some situations where there will be only one way to match the two line sets. We will discuss this problem later in Chapter 8. Now suppose we can chose three corresponding lines in the frame that are not mutually orthogonal. Since there is only one possible matching in this case, a proper rotation matrix R can be found by comparing the matching errors for the rotation matrix that are obtained by using all possible correspondences for the three lines. After a proper rotation is estimated, this rotation is applied to each direction of a line in the first set and then compared with its corresponding direction. If the dot product of a rotated direction with its corresponding direction is negative, the direction of the corresponding line in the second set is changed to its opposite. This will synchronize the directions of the two line sets, since, with proper rotation, the angle between the rotated direction in the first data set and its corresponding direction in the second data set should be less than 90 degrees.

When only two or three lines in the line set are available, new pairs of corresponding directions can be constructed from non-parallel lines for each line set by calculating their vector product. This will make the rotation estimated more accurate, especially when matrix $(nI - \sum_{i=1}^{n} \mathbf{v}_i \mathbf{v}_i^T)$ is almost singular.

Algorithm

Let $\{\mathcal{L}_i(P_i, \mathbf{v}_i)\}_{i=1}^{n}$ and $\{\mathcal{L}'_i(Q_i, \mathbf{u}_i)\}_{i=1}^{n}$ be two sets of lines.

- Synchronize the directions of the two sets of lines.
- Find the centre for each line set: C, C'.
- From $\{\mathbf{v}_i\}_{i=1}^{n}$ and $\{\mathbf{u}_i\}_{i=1}^{n}$ calculate the matrix
$$A = \sum_{i=1}^{n} \mathbf{u}\mathbf{v}^T.$$
- Use the techniques provided in section 3.2 to estimate rotation R from matrix A which maximizes $R \cdot A$.
- Calculate translation T:
$$T = C' - RC.$$

4.3 Matching two sets of directed planes

When there are flat areas on the surface of an object, these areas can be approximated with planes. Thus differences between two data sets due to rigid transformation can be estimated by matching the corresponding planes.

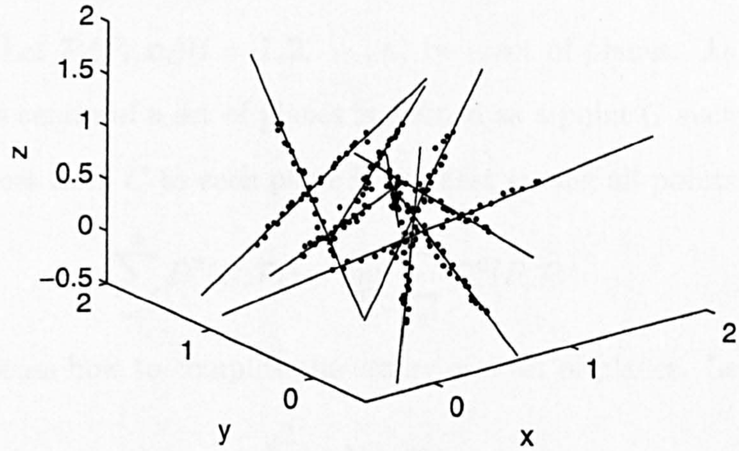

Figure 4.2: Estimate rigid transformation by line matching

4.3.1 The distance between a point and a plane

There are several ways to represent a plane. One way is to represent the plane with a point and a direction (called the normal of the plane). Let P_0 be a point on the plane and the unit vector $\mathbf{n}$ be the normal of the plane. Then for any point P on the plane, we have

$$(P - P_0) \cdot \mathbf{n} = 0. \tag{4.3.1}$$

A plane is denoted by $\mathcal{P}(P_0, \mathbf{n})$.

Let Q be a point, and let $\mathcal{P}(P_0, \mathbf{n})$ be a plane. The distance between the point and the plane can be represented as

$$D(Q, \mathcal{P}) = \|(Q - P_0) \cdot \mathbf{n}\|. \tag{4.3.2}$$

4.3.2 The centre of a set of planes

As with line matching, to register two sets of directed planes, we need to estimate their centres. Let $\mathcal{P}_i(P_i, \mathbf{n}_i)(i = 1, 2, \cdots, n)$ be a set of planes. As in the case of a set of lines, the centre of a set of planes is defined as a point C such that the sum of squared distances from C to each plane is smallest among all points in space, i.e.,

$$\sum_{i=1}^{n} D^2(C, \mathcal{P}_i) = \min_P \sum_{i=1}^{n} D^2(P, \mathcal{P}_i). \tag{4.3.3}$$

Now we discuss how to compute the centre of a set of planes. Let

$$\begin{aligned} S(C) &= \sum_{i=1}^{n} D^2(C, \mathcal{P}_i) \\ &= \sum_{i=1}^{n} \|(C - P_i) \cdot \mathbf{n}_i\|^2 \\ &= \sum_{i=1}^{n} (C - P_i)^T \mathbf{n}_i \mathbf{n}_i^T (C - P_i). \end{aligned} \tag{4.3.4}$$

Set

$$\frac{dS}{dC} = 2 \sum_{i=1}^{n} \mathbf{n}_i \mathbf{n}_i^T (C - P_i) = 0,$$

we have

$$\sum_{i=1}^{n} \mathbf{n}_i \mathbf{n}_i^T C = \sum_{i=1}^{n} \mathbf{n}_i \mathbf{n}_i^T P_i. \tag{4.3.5}$$

From proposition 4.6.2 in Appendix C, if there exist three linearly independent plane normals, sum $\sum_{i=1}^{n} \mathbf{n}_i \mathbf{n}_i^T$ will be nonsingular, and the centre of the set of planes will be

$$C = \left(\sum_{i=1}^{n} \mathbf{n}_i \mathbf{n}_i^T \right)^{-1} \sum_{i=1}^{n} \mathbf{n}_i \mathbf{n}_i^T P_i. \tag{4.3.6}$$

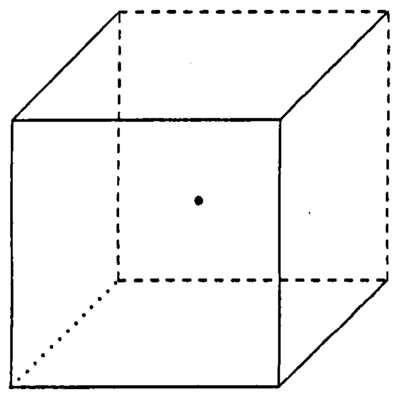

Figure 4.3: The centre of six planes

As with the centre for a set of lines, the centre of a set planes has the following properties.

Proposition 4.3.1. *The centre of a set of planes does not depend on the representation of planes.*

Proposition 4.3.2. *Let R and T be the rotation and translation. If C is the centre of plane set $\{\mathcal{P}_i(P_i, \mathrm{n}_i)\}_{i=1}^n$, then the centre of planes $\{\mathcal{P}(RP_i + T, R\mathrm{n}_i)\}_{i=1}^n$ will be $RC + T$.*

4.3.3 Registration of two sets of directed planes

The algorithm to register two sets of planes will be similar to the algorithm to register two sets of straight lines and will be described only briefly. Unlike line matching, we need at least three pairs of directed planes to determine the rigid transformation. First, the rotation R can be estimated directly from the corresponding normals with the techniques provided in section 3.2. Then the translation can be found as $T = C' - RC$, where C and C' are the centres of the two plane sets.

4.4 Matching two sets of geometric primitives

In practice, a frame can be designed to combine fiducial markers, fiducial lines and fiducial planes. In this section, we discuss how to estimate the rigid transformation when frame features are a mixture of points, lines and planes.

4.4.1 The centre of a set of geometric primitives

Let

$$\{P_i\}_{i=1}^{l} \bigcup \{\mathcal{L}(P'_j, \mathbf{v}_j)\}_{j=1}^{m} \bigcup \{\mathcal{P}(P''_k, \mathbf{n}_k)\}_{k=1}^{n}$$

be a set of geometric primitives consisting of points $\{P_i\}_{i=1}^{l}$, straight lines $\{\mathcal{L}(P'_j, \mathbf{v}_j)\}_{j=1}^{m}$ and planes $\{\mathcal{P}(P''_k, \mathbf{n}_k\}_{k=1}^{n}$. The centre of the set of geometric primitives is defined as a point C such that the sum of squared distances from C to each of these geometric primitive is least. Let

$$\begin{aligned} S(P) &= \sum_{i=1}^{l} \|P - P_i\|^2 + \sum_{j=1}^{m} D^2(P, \mathcal{L}_j) + \sum_{k=1}^{n} D^2(P, \mathcal{P}_k) && (4.4.1) \\ &= \sum_{i=1}^{l} \|P - P_i\|^2 \\ &+ \sum_{j=1}^{m} (P - P'_j)^T (I - \mathbf{v}_j \mathbf{v}_j^T)(P - P'_j) \\ &+ \sum_{k=1}^{n} (P - P''_k)^T \mathbf{n}_k \mathbf{n}_k^T (P - P''_k). && (4.4.2) \end{aligned}$$

Then the centre C should satisfy

$$S(C) = \min_P S(P).$$

As in the previous sections, we can show that the centre C will be

$$C = W^{-1}\left(\sum_{i=1}^{l} P_i + \sum_{j=1}^{m} (I - \mathbf{v}_j \mathbf{v}_j^T) P'_j + \sum_{k=1}^{n} \mathbf{n}_k \mathbf{n}_k^T P''_k\right) \qquad (4.4.3)$$

when matrix

$$W = lI + \sum_{j=1}^{m}(I - \mathbf{v}_j\mathbf{v}_j^T) + \sum_{k=1}^{n}\mathbf{n}_k\mathbf{n}_k^T \tag{4.4.4}$$

is nonsingular.

Now we discuss the circumstances in which the matrix W will be nonsingular.

- $l > 0$. In this case, matrix

  $$W = lI + \sum_{j=1}^{m}(I - \mathbf{v}_j\mathbf{v}_j^T) + \sum_{k=1}^{n}\mathbf{n}_k\mathbf{n}_k^T$$

  will be nonsingular as

  $$\sum_{j=1}^{m}(I - \mathbf{v}_j\mathbf{v}_j^T) + \sum_{k=1}^{n}\mathbf{n}_k\mathbf{n}_k^T$$

  is semi-positive definite.

- $l = 0$. This means that the set of primitives contains only lines and planes. In this case, matrix

  $$W = \sum_{j=1}^{m}(I - \mathbf{v}_j\mathbf{v}_j^T) + \sum_{k=1}^{n}\mathbf{n}_k\mathbf{n}_k^T.$$

  Note that both the matrix $\sum_{j=1}^{m}(I - \mathbf{v}_j\mathbf{v}_j^T)$ and the matrix $\sum_{k=1}^{n}\mathbf{n}_k\mathbf{n}_k^T$ are semi-positive definite. Thus, from Proposition 4.2.1 and Proposition 4.6.1 in Appendix C, matrix W cannot be singular when there are at least two linearly independent line directions or there are three linearly independent plane normals. Thus we know that the centre of this set of geometric primitives is uniquely defined if and only if one of the following three conditions is satisfied.

  1. There exist at least two lines that are not parallel to each other.
  2. There exist three linearly independent plane normals.

3. There exist a line and a plane such that the line is not parallel to the plane.

When one of these conditions is satisfied, the centre will be uniquely determined. The sufficiency of the first two conditions is obvious. For the third condition, it is also very direct. Let matrix $V = (\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_m)$ be formed with line directions $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_m$. When $rank(V) = 1$, matrix $\sum_{j=1}^{m}(I - \mathbf{v}_j\mathbf{v}_j^T)$ will be $m(I - \mathbf{v}_1\mathbf{v}_1^T)$. Let

$$U = I - \mathbf{v}_1\mathbf{v}_1^T$$

and let $\mathbf{v}_1 = (\alpha, \beta, \gamma)^T$. When replacing one of columns of U with a plane normal $\mathbf{n}$, it can be shown with simple algebra that $det(\mathbf{n}, \mathbf{u}_2, \mathbf{u}_3) = \alpha(\mathbf{v}_1 \cdot \mathbf{n})$, $det(\mathbf{u}_1, \mathbf{n}, \mathbf{u}_3) = \beta(\mathbf{v}_1 \cdot \mathbf{n})$, and $det(\mathbf{u}_1, \mathbf{u}_2, \mathbf{n}) = \gamma(\mathbf{v}_1 \cdot \mathbf{n})$, where $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ are the first, second and third column of matrix U. This means that whenever the direction $\mathbf{v}_1$ is not perpendicular to $\mathbf{n}$, $rank(U, \mathbf{n}) = 3$ since $\mathbf{n} \neq 0$. This confirms the sufficiency of condition 3.

4.4.2 Method of matching two sets of geometric primitives

Let

$$\{X_i\}_{i=1}^{l} \bigcup \{\mathcal{L}(Y_j, \mathbf{v}_j)\}_{j=1}^{m} \bigcup \{\mathcal{P}(Z_k, \mathbf{n}_k)\}_{k=1}^{n}$$

be a data set consisting of points, lines and planes, and let

$$\{X_i'\}_{i=1}^{l} \bigcup \{\mathcal{L}(Y_j', \mathbf{v}_j')\}_{j=1}^{m} \bigcup \{\mathcal{P}(Z_k', \mathbf{n}_k')\}_{k=1}^{n}$$

be their rigid transformation. The procedure to register the two sets of geometric primitives is similar to that of registering two corresponding directed line sets or two corresponding directed plane sets. First, the rotation can be found from the correspondences between the points, the directions of lines and the normals of planes. We

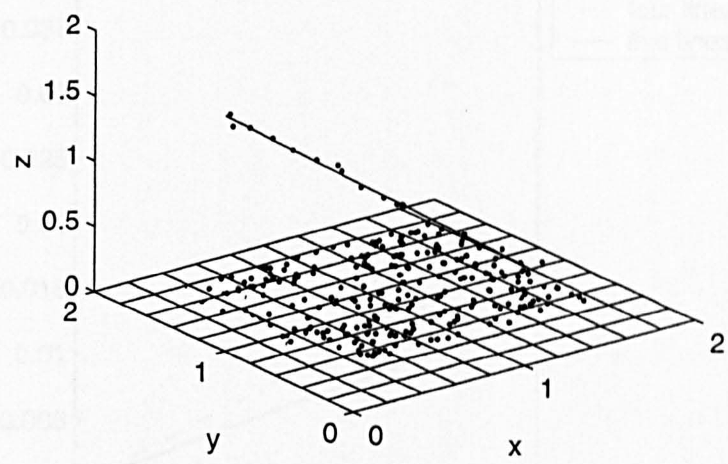

Figure 4.4: Estimate rigid transformation by line and plane matching

first compute the centres C and C' of the two sets of geometric primitives according to (4.4.3). Then, we calculate the correlation matrix

$$A = \sum_{i=1}^{l}(X_i' - C')(X_i - C)^T + \sum_{j=1}^{m} \mathbf{v}_j' \mathbf{v}_j{}^T + \sum_{k=1}^{n} \mathbf{n}_k' \mathbf{n}_k{}^T.$$

The rotation R can then be estimated by maximizing the scalar product $R \cdot A$ with the techniques provided in Section 3.2. The translation is calculated as $T = C' - RC$.

It should be noted that when only a line and a plane are available and the line is perpendicular to the plane, the rotation can not be determined properly, as in this case the normal of the plane is the same as the direction of the line.

4.5 Experimental results

In this section some tests on the line matching algorithm are given to demonstrate the performance of the algorithm. Due to the limitation of space, the results of testing

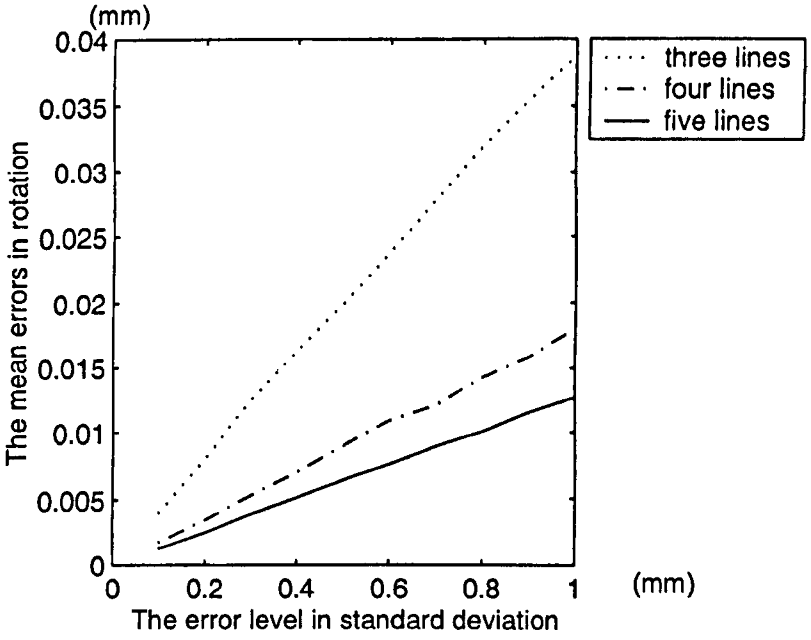

Figure 4.5: Errors in rotation against the noise level for line matching algorithm

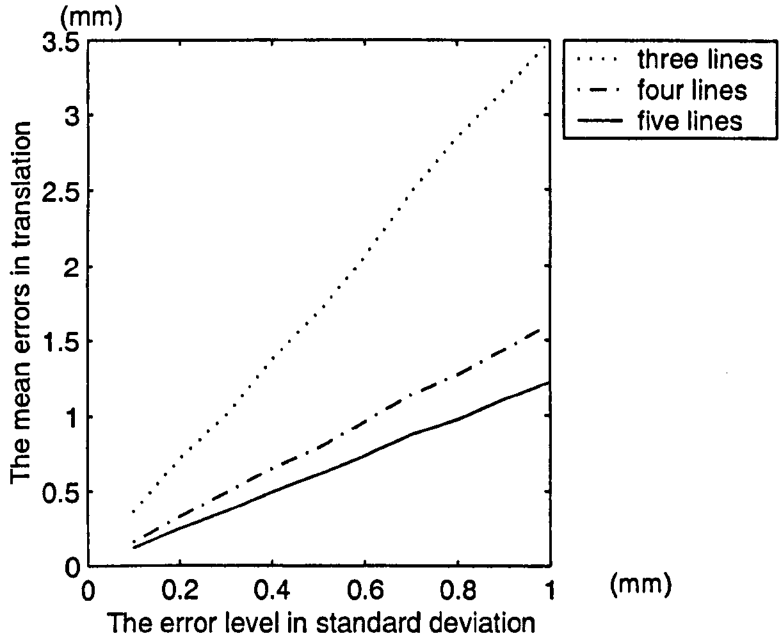

Figure 4.6: Errors in translation against the noise level for line matching algorithm

the general primitive matching algorithm will not be discussed here. In testing the line matching algorithm, three line data sets are used. The first line data set consists three lines which are generated with random positions and random orientations. The second and the third line data sets contain four lines and five lines respectively which are also randomly generated. Each line is about $200mm$ long. For each line data set, two groups of points are sampled from each line corresponding to different positions and different orientations of the line set. The number of points sampled from each line range from 10 to 20. Then the noise conforming to the normal distribution with standard deviation ranging from 0.1(mm) to 1(mm) is added to the three components of each sampled points. Straight lines are then fitted with the given data sets line by line. The direction correspondence for each pair of corresponding lines is determined by the order in which the points are sampled. Figure 4.5 shows the mean errors in rotation, measured by the Frobenius norm of the difference matrix between the estimated rotation and the true rotation over 5000 experiments, against different levels of standard deviation. As can be seen from the figure, the overall error increases nearly linearly with the increase of error level. The figure also shows that with an increase in the numbers of lines the error in rotation declines. Figure 4.6 shows the mean errors in translation over 5000 experiments against different levels of standard deviation. As can be seen, it has similar graph to the error in rotation.

The number of points sampled from each line also affects the matching accuracy, as shown in Figures 4.7 and 4.8, where three lines are considered. The greater the number of points used, the greater the accuracy in estimates of individual lines, and hence the greater the accuracy in the estimate of the transformation. Experiments on line distributions are also given and it is shown that the translation error will be

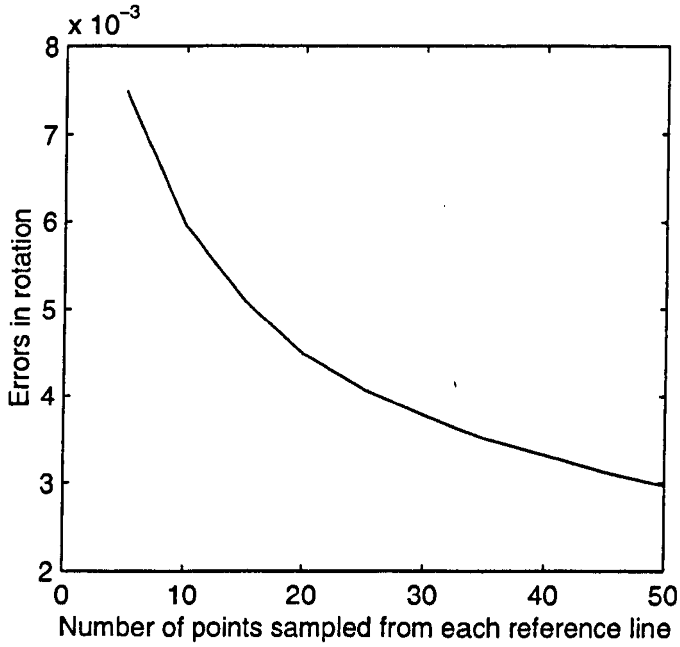

Figure 4.7: The errors in rotation against the number of points sampled from each line for line matching algorithm

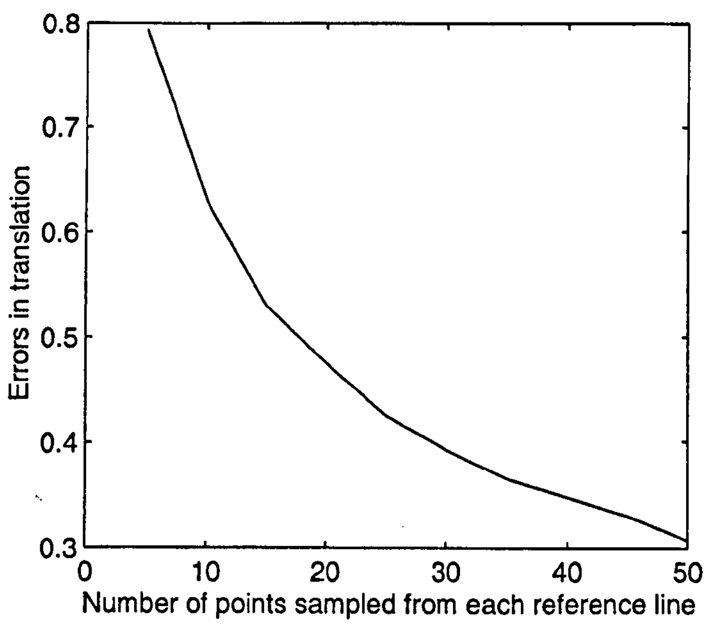

Figure 4.8: The errors in translation against the number of points sampled from each line for line matching algorithm

affected significantly by the line orientations. Well separated line orientations tend to give more accurate estimates.

To summarize, experiments have shown that when more then 4 lines are used, and when the positions and orientations of these lines are well separated, the line matching algorithm will give a very accurate estimate of the actual rigid transformation, if enough points are sampled for each line.

4.6 Appendices

In the previous sections, we have given some methods for matching two sets of geometric primitives consisting of points, lines and planes. These methods are based on the fact that the lines and planes can be accurately estimated from a set of points. Traditionally, line fitting is mainly based on algebraic distance. In this section, a line fitting technique based on geometric distance is developed using an eigen-technique, which is more robust and more accurate. 3D straight line fitting can also be provided with the algorithm given by Taubin in [94], by solving generalized eigen systems. However, our algorithm is simpler than Taubin's.

Appendix A: Least-squares Line Fitting

Let $\{P_i\}_{i=1}^n$ be a set of points, and let $\mathcal{L}(P_0, \mathbf{v})$ be the line to be estimated, where P_0 is a point on the line, and $\mathbf{v}$ is a unit vector representing the direction of the line. For each P_i, the square of the distance from the point to the line is

$$(P_i - P_0)^T(I - \mathbf{v}\mathbf{v}^T)(P_i - P_0), \qquad i = 1, 2, \cdots, n$$

and we wish to estimate P_0 and $\mathbf{v}$ by minimizing the sum

$$S = \sum_{i=1}^{n} (P_i - P_0)^T (I - \mathbf{v}\mathbf{v}^T)(P_i - P_0). \tag{4.6.1}$$

We first discuss how to estimate P_0. Setting $dS/dP_0 = 0$, we obtain

$$n(I - \mathbf{v}\mathbf{v}^T)P_0 = (I - \mathbf{v}\mathbf{v}^T)\sum_{i=1}^{n} P_i. \tag{4.6.2}$$

The solution for P_0 is not unique as matrix $I - \mathbf{v}\mathbf{v}^T$ is singular. An obvious choice for P_0 is the centre of gravity of the data set $\{P_i\}$. Let

$$\hat{P}_0 = \frac{1}{n}\sum_{i=1}^{n} P_i.$$

We will use $\hat{P}_0$ as the estimate of P_0.

Substituting $\hat{P}_0$ for P_0 in (4.6.1) and noting that

$$(P_i - \hat{P}_0)^T (I - \mathbf{v}\mathbf{v}^T)(P_i - \hat{P}_0) = (P_i - \hat{P}_0)^T (P_i - \hat{P}_0) - \mathbf{v}^T (P_i - \hat{P}_0)(P_i - \hat{P}_0)^T \mathbf{v},$$

it follows from equation (4.6.1) that

$$S = w - \mathbf{v}^T W \mathbf{v}, \tag{4.6.3}$$

where

$$w = \sum_{i=1}^{n} (P_i - \hat{P}_0)^T (P_i - \hat{P}_0)$$

is a real number and

$$W = \sum_{i=1}^{n} (P_i - \hat{P}_0)(P_i - \hat{P}_0)^T$$

is a 3×3 matrix.

It is evident that minimizing(4.6.3) is equivalent to maximizing

$$\mathbf{v}^T W \mathbf{v} \qquad \text{subject to} \qquad \|\mathbf{v}\| = 1. \tag{4.6.4}$$

The solution to 4.6.4 is well known as the unit eigenvector of W associated with the largest eigenvalue.

The algorithm to estimate the straight line from a set of points follows.

Algorithm for estimating the line

- P_0 is estimated as the centre of gravity of the data set:

$$\hat{P}_0 = \frac{1}{n}\sum_{i=1}^{n} P_i.$$

- Compute the matrix W:

$$W = \sum_{i=1}^{n}(P_i - \hat{P}_0)(P_i - \hat{P}_0)^T.$$

- The direction of the line can be found as the unit eigenvector of matrix W associated with the largest eigenvalue.

Appendix B: Least-squares plane Fitting

Fitting a plane from a set of points is much easier than fitting a line as for a plane $\mathcal{P}$, the Euclidean distance from a point P to $\mathcal{P}$ is the same as the algebraic distance. Therefore, for a given data set, $\{P_i(x_i, y_i, z_i)\}_{i=1}^{n}$, the plane can be fitted directly by minimizing

$$\sum_{i=1}^{n}(ax_i + by_i + cz_i - d)^2,$$

as a function of a, b, c, d, subject to a constraint, such as $a^2 + b^2 + c^2 + d^2 = 1$.

Here we provide another way of estimating a plane from a set of points by estimating a point that the plane passes through and the normal of the plane. This way of estimating a plane, though it is not as direct in computing as the first one, fits our geometric intuition better. As can be seen from the following discussion, the point that the plane passes through can be chosen as the centroid $\hat{P}_0$ of the data set, and the normal $\mathbf{n}$ of the plane can be estimated by minimizing the sum of squares of all scalar product of vector $\mathbf{n}$ and vector $P_i - \hat{P}_0$, which is supposed to be perpendicular to the plane normal. In fact let $\{P_i\}_{i=1}^n$ be a set of points, and let $\mathcal{P}(P_0, \mathbf{n})$ be the plane to be estimated, where P_0 is a point on the plane, and $\mathbf{n}$ is a unit vector represented the normal of the plane. For each P_i, the squared distance from the point to the plane can be put in the form

$$(P_i - P_0)^T \mathbf{n}\mathbf{n}^T (P_i - P_0), \qquad i = 1, 2, \cdots, n$$

and we want to estimate P_0 and $\mathbf{n}$ by minimizing the sum

$$S = \sum_{i=1}^{n} (P_i - P_0)^T \mathbf{n}\mathbf{n}^T (P_i - P_0). \tag{4.6.5}$$

By setting $dS/dP_0 = 0$, we obtain

$$n\mathbf{n}\mathbf{n}^T P_0 = \mathbf{n}\mathbf{n}^T \sum_{i=1}^{n} P_i. \tag{4.6.6}$$

The solution for P_0 is not unique as matrix $\mathbf{n}\mathbf{n}^T$ is singular. An obvious choice for P_0 is the centre of gravity of the data set $\{P_i\}_{i=1}^n$. Let

$$\hat{P}_0 = \frac{1}{n} \sum_{i=1}^{n} P_i.$$

We will use $\hat{P}_0$ as the estimate of P_0.

As

$$(P_i - P_0)^T \mathbf{n}\mathbf{n}^T (P_i - P_0) = \mathbf{n}^T (P_i - P_0)(P_i - P_0)^T \mathbf{n},$$

it follows from equation (4.6.5) that minimizing S subject to $\|\mathbf{n}\| = 1$ is equivalent to minimizing the quadratic form

$$S = \mathbf{n}^T W \mathbf{n}, \tag{4.6.7}$$

where

$$W = \sum_{i=1}^{n} (P_i - \hat{P}_0)(P_i - \hat{P}_0)^T.$$

The solution to 4.6.7 is the unit eigenvector of W associated with the smallest eigenvalue.

The algorithm to estimate the plane from a set of points follows.

Algorithm for estimating the plane

- P_0 is estimated as the centre of gravity of the data set:

$$\hat{P}_0 = \frac{1}{n} \sum_{i=1}^{n} P_i.$$

- Compute the matrix W:

$$W = \sum_{i=1}^{n} (P_i - \hat{P}_0)(P_i - \hat{P}_0)^T.$$

- The normal $\hat{\mathbf{n}}$ of the plane can be found as the unit eigenvector of matrix W associated with the smallest eigenvalue.

Above two methods for estimating a plane from a set of points are equivalent in accuracy as they are based on the same optimization criterion.

Appendix C: Some mathematical results

In this part, we provide two fundamental results in matrix algebra that have been used in our proof given in Sections from 4.2 to 4.4

Proposition 4.6.1. *Let A be an $n \times m$ matrix, then $A^T A$ is nonsingular if and only if A has linearly independent columns.*

Proof. Let $A = (\alpha_1, \alpha_2, \cdots, \alpha_m)$, where $\alpha_j = (a_{1j}, a_{2j}, \cdots, a_{nj})^T$ represents an n-dimensional vector$(j = 1, 2, \cdots, m)$. Then

$$A^T A = (A^T\alpha_1, A^T\alpha_2, \cdots, A^T\alpha_m).$$

Therefore, $det(A^T A) = 0$ if and only if there exist λ_1, λ_2, $\cdots$, λ_m, $\sum_{j=1}^{m} \lambda_j^2 > 0$, such that

$$\lambda_1 A^T\alpha_1 + \lambda_2 A^T\alpha_2 + \cdots + \lambda_m A^T\alpha_m = A^T(\lambda_1\alpha_1 + \lambda_2\alpha_2 + \cdots + \lambda_m\alpha_m) = 0.$$

This is equivalent to

$$(\lambda_1\alpha_1 + \lambda_2\alpha_2 + \cdots + \lambda_m\alpha_m, \alpha_2, \cdots, \alpha_m)^T(\lambda_1\alpha_1 + \lambda_2\alpha_2 + \cdots + \lambda_m\alpha_m) = 0,$$

or equivalently

$$\lambda_1\alpha_1 + \lambda_2\alpha_2 + \cdots + \lambda_m\alpha_m = 0.$$

That is, the columns of matrix A are not linearly independent. □

Proposition 4.6.2. $rank(A^T A) = rank(A)$.

Proof. We need only to show that $rank(A^T A) \geq rank(A)$. Let the rank of A is k. Without loss of generality, we assume that the first k columns of A, denoted by A_1, are linearly independent. Let A_2 represent the remaining columns of A, then

$$A^TA = \begin{pmatrix} A_1^T \\ A_2^T \end{pmatrix} (A_1, A_2) = \begin{pmatrix} A_1^TA_1 & A_1^TA_2 \\ A_2^TA_1 & A_2^TA_2 \end{pmatrix}.$$

According to Lemma 4.6.1, $A_1^TA_1$ is non-singular. Thus we have $rank(A^TA) \geq k = rank(A)$. $\square$

Chapter 5

The iterative closest line segment registration and the iterative closest triangle patch registration

5.1 Introduction

In the previous two chapters, we investigated some registration algorithms assuming that some links between the data sets are known, like point to point correspondence or directed line to directed line correspondence. Registration techniques like these are usually referred to as fiducial marker registration, which are efficient and accurate. However, to collect the reference data sets, external fiducial markers or fiducial frames have to be implanted or fitted to the area where surgery will be carried out. This brings about further trauma to the patient on the one hand and will put additional burden onto the surgeons by requiring them to set up the environment for collecting data from the markers on the other.

In this chapter we will consider how to compute the rigid transformation that link two given data sets without using fiducial markers or fiducial frames. In this case, the only information available is that the two data sets are collected from the same surface of a rigid object. Often, one data set (called the model data) has far more points than the other one(intra-operative data from the actual patient). As no exact correspondence information is known about the two data sets, non-landmark registration techniques are required to match the two data sets. These are much more complicated than the landmark registration techniques due to the nature of the problem. In this case, an iterative optimization procedure is unavoidable for data matching. Therefore, non-landmark registration techniques are slower, less efficient and less accurate in general compared with landmark registration.

Current non-landmark registration techniques can be classified roughly either as surface matching or as volume matching. The idea of volume matching will not be considered in this thesis for the following reasons. Firstly, volume data are computationally expensive as the size of the data set can often be as large as hundreds of megabytes. Secondly, as the intraoperative data are assumed to be sampled from the surface of a bone, the data position within a bone will not actually relate to the intraoperative data collected. Thirdly, volume data always contains too much detailed information that might not be an advantage for our registration problem. It should be noted that we only have very few points sampled from the surface of the surgical object for intraoperative data. If the surface of the bone is not smooth enough, the rigid transformation that matches the intraoperative data with the volume data might not be unique and therefore it might not always be possible to estimate the actual orientation correctly.

In this thesis, we will follow the idea of surface matching to investigate non-landmark registration techniques. Amongst all surface matching algorithms, the ICP algorithm [10] has been one of the most popular techniques. The main advantage of the ICP algorithm is that it always converges. However, in many cases, it does not converge to the expected transformation if a good initial estimate is not available. In this chapter, we first generalise the reference line segment registration technique presented in [43] from equal length line segments to general line segments. Then we develop reference triangle patch registration by extending the concept of line segment to triangle patches. The iterative closest line segment (ICL) registration algorithm and the iterative triangle patch (ICT) registration algorithms are developed along similar lines to the conventional iterative closest point (ICP) registration algorithm, and are shown to be more likely to converge to the true rigid transformation than the ICP algorithm.

5.2 Closed-form Line Segment Registration

Definition 5.2.1. Let P_1, $P_2 \in \mathbb{R}^3$ be two points. The ordered pair $[P_1, P_2]$ is called a line segment in $\mathbb{R}^3$. The set of all line segments on $\mathbb{R}^3$ is denoted by $\mathbf{L}$.

Definition 5.2.2. Let $\mathbb{L} \in \mathbf{L}$ be a line segment in $\mathbb{R}^3$, and let F be a transformation on space $\mathbb{R}^3$. Then F can be extended to be a line segment transformation by defining

$$F[P_1, P_2] = [FP_1, FP_2]. \tag{5.2.1}$$

$[FP_1, FP_2]$ is called the transformation of the line segment $\mathbb{L}$. For translation, we will write $T[P_1, P_2] = [P_1 + T, P_2 + T]$ more naturally as $[P_1, P_2] + T$.

Similarly, the meanings of other operations on $\mathbb{R}^3$ can also be extended to include operations on line segments.

Definition 5.2.3. Let $\mathbb{L}_1 = [P_1, P_2], \mathbb{L}_2 = [Q_1, Q_2] \in \mathbf{L}$ be two line segments, and a, b two real numbers. We define

$$a\mathbb{L}_1 + b\mathbb{L}_2 = [aP_1 + bQ_1, aP_2 + bQ_2]. \tag{5.2.2}$$

It should be noted that this definition is different from the set operation obtained with the conventional extension principle.

Let $\mathbb{L}_1 = [P_1, P_2], \mathbb{L}_2 = [Q_1, Q_2] \in \mathbf{L}$ be two line segments. Geometrically, $\mathbb{L}_1$ and $\mathbb{L}_2$ can be represented as functions in the form: $f_1(\lambda) = P_1 + \lambda(P_2 - P_1)$, and $f_2(\lambda) = Q_1 + \lambda(Q_2 - Q_1)$ respectively, where $0 \leq \lambda \leq 1$. If λ is incremented by $d\lambda$, then f_1 and f_2 are incremented by $(P_2 - P_1)d\lambda$ and $(Q_2 - Q_1)d\lambda$ respectively. The distance between these two line elements can be approximated by the area of the trapezium that has height $\|f_1(\lambda) - f_2(\lambda)\|^2$ with top-edge and bottom edge defined by $(P_2 - P_1)d\lambda$ and $(Q_2 - Q_1)d\lambda$ approximately, i.e.,

$$\frac{1}{2}(\|P_2 - P_1\| + \|Q_2 - Q_1\|)\|f_1(\lambda) - f_2(\lambda)\|^2 d\lambda.$$

We choose to use $\|f_1(\lambda) - f_2(\lambda)\|^2$ rather than $\|f_1(\lambda) - f_2(\lambda)\|$ to measure the distance between points $f_1(\lambda)$ and $f_2(\lambda)$ only for convenience of computation. The distance between the two line segments can thus be described by the following integration:

$$\begin{aligned} & \frac{l_1 + l_2}{2} \int_0^1 \|f_1(\lambda) - f_2(\lambda)\|^2 \, d\lambda \\ = \; & \frac{l_1 + l_2}{6} \left(\|P_1 - Q_1\|^2 + \|P_2 - Q_2\|^2 + (P_1 - Q_1) \cdot (P_2 - Q_2)\right), \end{aligned} \tag{5.2.3}$$

where $l_1 = \|P_2 - P_1\|$, and $l_2 = \|Q_2 - Q_1\|$.

Definition 5.2.4. Let $\mathbb{L}_1 = [P_1, P_2], \mathbb{L}_2 = [Q_1, Q_2]$ be two line segments. The distance between the two line segments is defined as (5.2.3) and is denoted by $D(\mathbb{L}_1, \mathbb{L}_2)$.

It should be noted that the value of the above integration depends on the corresponding relations between the ends of the two line segments. Thus, the distance between two line segments defined above is direction dependent.

In [43], the distance between line segments has been defined for the case where the lengths of the line segments are equal. We will show that our definition is more general.

Proposition 5.2.1. *Let* $\mathbb{L}_1 = [P_1, P_2], \mathbb{L}_2 = [Q_1, Q_2] \in \mathbf{L}$ *be two line segments. Then*

$$D(\mathbb{L}_1, \mathbb{L}_2) = \frac{l_1 + l_2}{2}\left(\|O_1 - O_2\|^2 + \frac{l_1 l_2}{12}\|V_1 - V_2\|^2 + \frac{1}{12}(l_1 - l_2)^2\right), \qquad (5.2.4)$$

where $O_1 = (P_1 + P_2)/2, O_2 = (Q_1 + Q_2)/2$ *are the centres of the two line segments, the unit vectors* V_1, V_2 *are their directions and* l_1, l_2 *their lengths.*

The geometric meaning of the measure is clear. The first term of equation (5.2.4) measures the difference between the two line segments in position, the second term measures the difference in direction, and the third term measures the difference in length.

Proof.

$$\|O_1 - O_2\|^2 = \frac{1}{4}\left(\|P_1 - Q_1\|^2 + \|P_2 - Q_2\|^2 + 2(P_1 - Q_1)\cdot(P_2 - Q_2)\right),$$

$$l_1 l_2\|V_1 - V_2\|^2 = 2l_1 l_2 - 2(P_2 - P_1)\cdot(Q_2 - Q_1),$$

$$\begin{aligned} l_1 l_2\|V_1 - V_2\|^2 \quad + \quad (l_1 - l_2)^2 = \\ \|P_1 - Q_1\|^2 + \|P_2 - Q_2\|^2 - 2(P_1 - Q_1)\cdot(P_2 - Q_2), \end{aligned}$$

and so

$$\begin{aligned}
& \frac{l_1+l_2}{2}\left(\|O_1-O_2\|^2+\frac{l_1 l_2}{12}\|V_1-V_2|^2+\frac{1}{12}(l_1-l_2)^2\right) \\
= & \frac{l_1+l_2}{6}\left(\|P_1-Q_1\|^2+\|P_2-Q_2\|^2+(P_1-Q_1)\cdot(P_2-Q_2)\right).
\end{aligned}$$

□

When $l_1 = l_2 = l$, it follows from (5.2.4) that

$$\begin{aligned}
D(\mathbb{L}_1, \mathbb{L}_2) &= l\|O_1-O_2\|^2+\frac{l^3}{12}\|V_1-V_2\|^2 \\
&= l\|O_1-O_2\|^2+\frac{l^3}{6}(1-V_1\cdot V_2).
\end{aligned}$$

This is the definition given in [43].

Though (5.2.4) is more direct than (5.2.3), we will mainly use (5.2.3) as it is easier to compute from points.

Next, we will establish a line segment registration algorithm. Let

$$\{[P_n, P_n']\}_{n=1}^N, \quad \{[Q_n, Q_n']\}_{n=1}^N$$

be two sets of line segments. Suppose that there exists a rotation R and a translation T such that

$$[Q_n, Q_n'] = R[P_n, P_n'] + T + [\varepsilon_n, \varepsilon_n'], \qquad n = 1, 2, \cdots, N$$

where $[\varepsilon_n, \varepsilon_n'], n = 1, 2, \cdots, N$, are experimental errors. We estimate R and T by minimizing

$$\begin{aligned}
\Sigma &= \sum_{n=1}^N D(\ R[P_n, P_n'] + T,\ [Q_n, Q_n']\ ) \\
&= \sum_{n=1}^N \frac{l_{1_n}+l_{2_n}}{6}(\ \|Q_n - RP_n - T\|^2 + \|Q_n' - RP_n' - T\|^2 \\
&\qquad + (Q_n - RP_n - T)\cdot(Q_n' - RP_n' - T)\ ),
\end{aligned} \tag{5.2.5}$$

where l_{1_n}, l_{2_n} are the lengths of $[P_n, P'_n]$ and $[Q_n, Q'_n]$ respectively.

For a minimum value, $\frac{\partial\Sigma}{\partial T} = 0$. It follows immediately that the optimal translation should be chosen as

$$T = \bar{Q} - R\bar{P}, \tag{5.2.6}$$

where

$$\bar{P} = \sum_{n=1}^{N} w_n \frac{P_n + P'_n}{2}, \qquad \bar{Q} = \sum_{n=1}^{N} w_n \frac{Q_n + Q'_n}{2},$$
$$w = \sum_{n=1}^{N} (l_{1_n} + l_{2_n}), \quad \text{and} \quad w_n = \frac{l_{1_n} + l_{2_n}}{w}. \tag{5.2.7}$$

Let

$$[\bar{P}_n, \bar{P}'_n] = [P_n, P'_n] - \bar{P}, \qquad [\bar{Q}_n, \bar{Q}'_n] = [Q_n, Q'_n] - \bar{Q},$$

$n = 1, 2, \cdots, N$. Substituting (5.2.6) into (5.2.5), we have

$$\Sigma = \sum_{n=1}^{N} \frac{l_{1_n} + l_{2_n}}{6} (\; \|\bar{Q}_n - R\bar{P}_n\|^2 + \|\bar{Q}'_n - R\bar{P}'_n\|^2$$
$$+ (\bar{Q}_n - R\bar{P}_n) \cdot (\bar{Q}'_n - R\bar{P}'_n) \;). \tag{5.2.8}$$

This sum can be further written as

$$\Sigma = \Delta - R \cdot \sum_{n=1}^{N} \frac{l_{1_n} + l_{2_n}}{6} (\, 2\bar{Q}_n \bar{P}_n^T + 2\bar{Q}'_n \bar{P}'^T_n + \bar{Q}_n \bar{P}'^T_n + \bar{Q}'_n \bar{P}_n^T \,), \tag{5.2.9}$$

where

$$\Delta = \sum_{n=1}^{N} \frac{l_{1_n} + l_{2_n}}{6} (\, \|\bar{Q}_n\|^2 + \|\bar{P}_n\|^2 + \|\bar{Q}'_n\|^2 + \|\bar{P}'_n\|^2 + \bar{Q}_n \cdot \bar{Q}'_n + \bar{P}_n \cdot \bar{P}'_n \,).$$

is a constant independent of rotation R. Therefore minimizing (5.2.8) is equivalent to maximizing

$$\Gamma = R \cdot A, \tag{5.2.10}$$

where the matrix

$$A = \sum_{n=1}^{N} \frac{l_{1_n} + l_{2_n}}{6} (2\bar{Q}_n \bar{P}_n^T + 2\bar{Q}'_n \bar{P'}_n^T + \bar{Q}_n \bar{P'}_n^T + \bar{Q}'_n \bar{P}_n^T). \tag{5.2.11}$$

The rotation R that maximizes (5.2.10) can be found using any of the four closed form solutions presented in section 3.2.

5.3 Closed-Form Triangular patch Registration

In this section, the concept of a triangle patch is introduced. The contents of this section can then be treated like that of the previous section. Triangle patches can be seen as a generalization of the line segments introduced in the previous section. The main results of this section can be further generalized without difficulty to an ordered point set of any size.

Definition 5.3.1. Let P_1, P_2, $P_3 \in \mathbb{R}^3$ be three points. The ordered triple $[P_1, P_2, P_3]$is called a triangle patch in $\mathbb{R}^3$. The set of all triangle patches on $\mathbb{R}^3$ is denoted by $\mathbf{T}$.

Definition 5.3.2. Let $\mathbb{T} \in \mathbf{T}$ be a triangle patch in $\mathbb{R}^3$. F is a transformation on space $\mathbb{R}^3$. Then this transformation can be extended to apply to a triangle patch by means of the definition

$$F[P_1, P_2, P_3] = [FP_1, FP_2, FP_3]. \tag{5.3.1}$$

$[FP_1, FP_2, FP_3]$ is called the transformation of triangle patch $\mathbb{T}$. For translation, we also write

$$T[P_1, P_2, P_3] = [P_1 + T, P_2 + T, P_3 + T],$$

or more conveniently as

$$[P_1, P_2, P_3] + T.$$

Addition and scaling operations can also be defined on triangle patches as they are on line segments.

Definition 5.3.3. Let $\mathbb{T}_1 = [P_1, P_2, P_3], \mathbb{T}_2 = [Q_1, Q_2, Q_3] \in \mathbf{T}$ be two triangle patches, and a, b two real numbers. We define

$$a\mathbb{T}_1 + b\mathbb{T}_2 = [aP_1 + bQ_1, aP_2 + bQ_2, aP_3 + bQ_3]. \tag{5.3.2}$$

We now discuss the concept of distance between triangle patches to measure their closeness. Let $\mathbb{T}_1 = [P_1, P_2, P_3], \mathbb{T}_2 = [Q_1, Q_2, Q_3] \in \mathbf{T}$ be two triangle patches. Geometrically, they can be represented as functions

$$f_1(u, v) = P_1 + u(P_2 - P_1) + v(P_3 - P_1)$$

and

$$f_2(u, v) = Q_1 + u(Q_2 - Q_1) + v(Q_3 - Q_1)$$

respectively, where $0 \leq u, v \leq 1$, and $u + v \leq 1$. As with line segments, we define a prism with height $\|f_1(u,v) - f_2(u,v)\|^2$. The top surface is defined with triangle $\mathbb{T}_1$ and bottom is defined with triangle $\mathbb{T}_2$, and the volume of the prism is used to measure the closeness of the two triangles. Given increments du and dv to variable u and v, the corresponding increment in the volume of the prism can be approximated by

$$C\|f_1(u,v) - f_2(u,v)\|^2 du dv,$$

where

$$C = \frac{1}{2}(\ \|(P_2 - P_1) \times (P_3 - P_1)\| + \|(Q_2 - Q_1) \times (Q_3 - Q_1)\|\ )$$

and $\times$ denotes the vector product of two vectors. Thus the volume is

$$
\begin{aligned}
& \iint_{\mathcal{D}} C\|f_1(u,v) - f_2(u,v)\|^2 \, dudv \\
= \ & \frac{C}{12}(\|P_1 - Q_1\|^2 + \|P_2 - Q_2\|^2 + \|P_3 - Q_3\|^2 \\
& +(P_1 - Q_1)\cdot(P_2 - Q_2) + (P_1 - Q_1)\cdot(P_3 - Q_3) \\
& +(P_2 - Q_2)\cdot(P_3 - Q_3)\ ),
\end{aligned}
\tag{5.3.3}
$$

where $\mathcal{D} = \{(u,v) | 0 \le u+v \le 1, 0 \le u, v \le 1\}$.

Definition 5.3.4. Let $\mathbb{T}_1 = [P_1, P_2, P_3], \mathbb{T}_2 = [Q_1, Q_2, Q_3] \in \mathbf{T}$ be two triangle patches. The distance between the two triangle patches is defined as (5.3.3) and is denoted by $D(\mathbb{T}_1, \mathbb{T}_2)$

It should be noted that the distance between two triangle patches defined above depends on the ordering of their vertices. As with Proposition 5.2.1, we have

Proposition 5.3.1. *Let $\mathbb{T}_1 = [P_1, P_2, P_3], \mathbb{T}_2 = [Q_1, Q_2, Q_3] \in \mathbf{T}$ be two triangle patches. Then*

$$
\begin{aligned}
D(\mathbb{T}_1, \mathbb{T}_2) \ = \ & \frac{C}{2}\{\ \|O_1 - O_2\|^2 \\
& +\frac{1}{36}(\ l_1 l_1' \|V_1 - V_1'\|^2 + l_2 l_2' \|V_2 - V_2'\|^2 + l_3 l_3' \|V_3 - V_3'\|^2\ ) \\
& +\frac{1}{36}\ (\ (l_1 - l_1')^2 + (l_2 - l_2')^2 + (l_3 - l_3')^2\ )\ \},
\end{aligned}
\tag{5.3.4}
$$

where $O_1 = (P_1 + P_2 + P_3)/3, O_2 = (Q_1 + Q_2 + Q_3)/3$ denote the centres of the two triangles, unit vector $V_1, V_2, V_3, V_1', V_2', V_3'$ are the corresponding directions of edges, and $l_1, l_2, l_3, l_1', l_2', l_3'$ are corresponding lengths of the three edges.

The proof of this proposition is similar to that of Proposition 5.2.1.

The first term of equation (5.3.4) measures the difference between the two triangle patches in position, the following three terms measure the differences of the two triangle patches in orientation, and the last three terms measure the differences in size.

Now, we discuss a technique for triangle patch registration. Let

$$\{[P_n, P'_n, P''_n]\}_{n=1}^N, \qquad \{[Q_n, Q'_n, Q''_n]\}_{n=1}^N$$

be two sets of triangle patches. Suppose that there exist a rotation R and a translation T such that

$$[Q_n, Q'_n, Q''_n] = R[P_n, P'_n, P''_n] + T + [\varepsilon_n, \varepsilon'_n, \varepsilon''_n], \qquad n = 1, 2, \cdots, N$$

where $[\varepsilon_n, \varepsilon'_n, \varepsilon''_n], n = 1, 2, \cdots, N$, are experimental errors. We estimate R and T by minimizing

$$\begin{aligned}
\Sigma &= \sum_{n=1}^{N} D(R[P_n, P'_n, P''_n] + T,\ [Q_n, Q'_n, Q''_n]) \\
&= \sum_{n=1}^{N} \frac{C_n}{12}(\|Q - RP_n - T\|^2 + \|Q' - RP'_n - T\|^2 \\
&\quad + \|Q'' - RP''_n - T\|^2 + (Q - RP_n - T) \cdot (Q' - RP'_n - T) \\
&\quad + (Q - RP_n - T) \cdot (Q'' - RP''_n - T) \\
&\quad + (Q' - RP'_n - T) \cdot (Q'' - RP''_n - T)\ ),
\end{aligned} \tag{5.3.5}$$

where

$$C_n = \tfrac{1}{2}\left(\|(P'_n - P_n) \times (P''_n - P_n)\| + \|(Q'_n - Q_n) \times (Q''_n - Q_n)\|\right).$$

is an invariant under rigid transformation.

For a minimum value, $\frac{\partial \Sigma}{\partial T} = 0$. It follows immediately that the optimal translation is chosen as

$$T = \bar{Q} - R\bar{P}, \tag{5.3.6}$$

where

$$\bar{P} = \sum_{n=1}^{N} w_n \frac{P_n + P_n' + P_n''}{3}, \qquad \bar{Q} = \sum_{n=1}^{N} w_n \frac{Q_n + Q_n' + Q_n''}{3},$$
$$w = \sum_{n=1}^{N} C_n, \qquad w_n = \frac{C_n}{w}. \tag{5.3.7}$$

Let

$$[\bar{P}_n, \bar{P}_n', \bar{P}_n''] = [P_n, P_n', P_n''] - \bar{P},$$

$$[\bar{Q}_n, \bar{Q}_n', \bar{Q}_n''] = [Q_n, Q_n', Q_n''] - \bar{Q},$$

$n = 1, 2, \cdots, N$. Substituting (5.3.6) into (5.3.5) we have

$$\begin{aligned}\Sigma &= \sum_{n=1}^{N} \frac{C_n}{12}(\|\bar{Q}_n - R\bar{P}_n\|^2 + \|\bar{Q}_n' - R\bar{P}_n'\|^2 + \|\bar{Q}_n'' - R\bar{P}_n''\|^2 \\ &\quad + (\bar{Q}_n - R\bar{P}_n) \cdot (\bar{Q}_n' - R\bar{P}_n') + (\bar{Q}_n - R\bar{P}_n) \cdot (\bar{Q}_n'' - R\bar{P}_n'') \\ &\quad + (\bar{Q}_n' - R\bar{P}_n') \cdot (\bar{Q}_n'' - R\bar{P}_n'')).\end{aligned} \tag{5.3.8}$$

This sum can be further written as

$$\Sigma = \Delta - R \cdot B, \tag{5.3.9}$$

where Δ is a constant independent of rotation R and

$$\begin{aligned}B &= \sum_{n=1}^{N} \frac{C_n}{12}(2\bar{Q}_n'' \bar{P''}_n^T + 2\bar{Q'}_n \bar{P'}_n^T + 2\bar{Q}_n \bar{P}_n^T \\ &\quad + \bar{Q}_n \bar{P'}_n^T + \bar{Q}_n \bar{P''}_n^T + \bar{Q'}_n \bar{P}_n^T + \bar{Q'}_n \bar{P''}_n^T + \bar{Q''}_n \bar{P}_n^T + \bar{Q''}_n \bar{P'}_n^T).\end{aligned}$$

Therefore minimizing (5.3.8) is equivalent to maximizing

$$R \cdot B. \tag{5.3.10}$$

The rotation R that maximizes (5.3.10) can be estimated using any of the four closed-form solutions presented in section 3.2.

5.4 The iterative closest line segment and the iterative closest triangle patch Registrations

In this section, the iterative closest line segment registration algorithm(ICL) and the iterative closest triangle patch registration algorithm(ICT) are presented. As the ICT algorithm is similar to the ICL algorithm in principle, it will just be discussed briefly.

Let $\mathcal{P} = \{P_n\}_{n=1}^{N}$, $\mathcal{Q} = \{Q_m\}_{m=1}^{M}$ be two sets of points sampled from the same surgical surface of the patient's bone intraoperatively and preoperatively. It is assumed that no point to point correspondence relations are known between the two data sets and that M is much larger than N. In this case, the data set $\mathcal{Q}$ should be large enough to represent the surface of the object realistically, otherwise, the estimated rotation and translation might not be what we expect. The ICL algorithm works by searching for the closest line segments in $\mathcal{Q}$ for each line segments in $\mathcal{P}$.

In searching for pair of closest line segments, a dynamically weighted distance is used to measure the closeness of two line segments. Note that the length of a line segment is left unchanged by rotation and translation. For a given line segment in $\mathcal{P}$, its closest line segment in $\mathcal{Q}$ should have a similar length. If two line segments are significantly different in length, then one cannot be the transformation of the other. With this fact in mind, we could modify the distance between line segments by increasing the weight on the length difference to filter out the unlikely pairs.

Therefore, at the beginning, we can set the weight on the length difference between two line segments to be very large such that whether two line segments are close mainly depends on whether they have similar lengths. The weight on length difference is then gradually minimized according to the matching error in each iteration.

In order to take advantage of the length of the line segments, the error in matching in the k^{th} step of the iteration is used as the weight for the differences in length, i.e., in the k^{th} step, the following definition of distance between line segments is used in searching for the closest line segments in our ICL algorithm:

$$D_k(\mathbb{L}_1, \mathbb{L}_2) = D(\mathbb{L}_1, \mathbb{L}_2) + e_{k-1}(l_1 - l_2)^2, \tag{5.4.1}$$

where the distance D is defined by (5.2.3) and e_{k-1} is the error sum in the $(k-1)^{th}$ step, and l_1, l_2 are the lengths of $\mathbb{L}_1$ and $\mathbb{L}_2$ respectively. Since the lengths of the line segments are invariant under rotation and translation, the estimated rotation and translation based on distance D_k and distance D are the same. To enhance the efficiency of the ICL algorithm, it is proposed to sort the line segments of the intra-operative data according to their lengths so that longer line segments are compared first. The reason for doing this is that the orientation of the object is mainly determined by these longer edges.

Let $\mathbb{L} = [P, P']$ be a line segment in data set $\mathcal{P}$, and $[\mathcal{Q}(P), \mathcal{Q}(P')]$ denote its closest line segment in data set $\mathcal{Q}$. Once the closest line segments for all elements in $\mathcal{P}$ have been found, the line segment registration method given in section 5.2 is used on the data sets $\{[P_i, P_j] | i < j; i, j = 1, 2, \cdots, N\}$ and $\{[\mathcal{Q}(P_i), \mathcal{Q}(P_j)] | i < j; i, j = 1, 2, \cdots, N\}$. The estimated rotation and translation are then applied to $\mathcal{P}$. Then the updated $\mathcal{P}$ are used as the intra-operative data. This procedure is repeated until the difference between two consecutive error sums is smaller than the given tolerance. This procedure is always convergent under appropriate scaling. Let $\mathcal{P}^{(k)}$ be the point set after updating $\mathcal{P}$ k times. For each line segment $[P_i^{(k)}, P_j^{(k)}]$ in $\mathcal{P}^{(k)}$, the closest line segment $[\mathcal{Q}(P_i^{(k)}), \mathcal{Q}(P_j^{(k)})] \in \mathcal{Q}$ is defined as the line segment that satisfies

$$D_k\left([P_i^{(k)}, P_j^{(k)}],\ [\mathcal{Q}(P_i^{(k)}), \mathcal{Q}(P_j^{(k)})]\right) = \min_{Q,Q' \in \mathcal{Q}} D_k\left([P_i^{(k)}, P_j^{(k)}],\ [Q, Q']\right) .$$

Algorithm 5.4.1. 1. Sort the line segments in data set $\mathcal{P}$ in such a way that the longer line segments are considered first in computing the closest line segments. In practice, if the number of points in $\mathcal{P}$ is N, we can just use the first N longest line segments to establish the registration. Let the first N longest line segments in data set $\mathcal{P}^{(k)}$ be $\mathbb{L}_1^{(k)}, \mathbb{L}_2^{(k)}, \cdots, \mathbb{L}_N^{(k)}$.

2. Initialize rotation, translation and e_0: R=I, T=0; e_0 should be initialized as large as possible;

3. For each line segments $\mathbb{L}_i^{(k)} \in \mathcal{P}^{(k)}$, find a line segment $\mathcal{Q}(\mathbb{L}_i^{(k)})$ in the pre-operative model such that the distance between line segments $\mathbb{L}_i^{(k)}$ and $\mathcal{Q}(\mathbb{L}_i^{(k)})$ is minimal in the sense of D_k defined in (5.4.1).

4. For line segments $\{\mathbb{L}_i^{(k)}\}_{i=1}^N$ and line segments $\{\mathcal{Q}(\mathbb{L}_i^{(k)})\}_{i=1}^N$, compute the rotation matrix R' and the translation T' using the line segment registration approach developed in section 5.2. Set $R = R'R$ and $T = R'T + T'$.

5. Calculate the error sum:

$$e_k = \sum_{i=1}^{N} D_k\left(\mathbb{L}_i^{(k)},\ \mathcal{Q}(\mathbb{L}_i^{(k)})\right).$$

If $|e_k - e_{k+1}|$ is less than the given tolerance, stop; otherwise, set

$$\mathbb{L}_i^{(k+1)} = R'\mathbb{L}_i^{(k)} + T', \qquad n = 1, 2, \cdots, N,$$

and repeat from step 3.

Now we show that this algorithm always converges under appropriate scaling. Let

$$e_k = \sum_{i=1}^{N} D_k\left(\mathbb{L}_i^{(k)},\ \mathcal{Q}(\mathbb{L}_i^{(k)})\right), \tag{5.4.2}$$

and let $R^{(k)}, T^{(k)}$ be the estimated rotation and translation from line segments $\{\mathbb{L}_i^{(k)}\}_{i=1}^N$ and line segments $\{\mathcal{Q}(\mathbb{L}_i^{(k)})\}_{i=1}^N$. Define

$$d_k = \sum_{i=1}^{N} D_k\left(\mathbb{L}_i^{(k+1)},\ \mathcal{Q}(\mathbb{L}_i^{(k)})\right). \tag{5.4.3}$$

Let *Rot* and Tr represent any rotation and any translation respectively. Since the rotation and translation estimates based on D_k and D are equal, according to the line segment registration algorithm,

$$\begin{aligned}
d_k &= \sum_{i=1}^{N} D_k\left(\mathbb{L}_i^{(k+1)},\ \mathcal{Q}(\mathbb{L}_i^{(k)})\right) \\
&= \sum_{i=1}^{N} D_k\left(R^{(k)}\mathbb{L}_i^{(k)} + T^{(k)},\ \mathcal{Q}(\mathbb{L}_i^{(k)})\right) \\
&= \min_{Rot,Tr} \sum_{i=1}^{N} D_k\left(Rot\mathbb{L}_i^{(k)} + Tr,\ \mathcal{Q}(\mathbb{L}_i^{(k)})\right) \\
&\le \sum_{i=1}^{N} D_k\left(\mathbb{L}_i^{(k)},\ \mathcal{Q}(\mathbb{L}_i^{(k)})\right) \\
&= e_k.
\end{aligned}$$

On the other hand, from the definition of closest line segments, we have

$$\begin{aligned} e_{k+1} &= \sum_{i=1}^{N} D_{k+1}\left(\mathbb{L}_i^{(k+1)},\ \mathcal{Q}(\mathbb{L}_i^{(k+1)})\right) \\ &\leq \sum_{i=1}^{N} D_{k+1}\left(\mathbb{L}_i^{(k+1)},\ \mathcal{Q}(\mathbb{L}_i^{(k)})\right), \end{aligned} \tag{5.4.4}$$

since $\mathcal{Q}(\mathbb{L}_i^{(k+1)})$ is the closest line segments of $\mathbb{L}_i^{(k+1)}$ in the sense of D_{k+1}. But

$$\begin{aligned} D_{k+1}\left(\mathbb{L}_i^{(k+1)},\ \mathcal{Q}(\mathbb{L}_i^{(k)})\right) &= D\left(\mathbb{L}_i^{(k+1)},\ \mathcal{Q}(\mathbb{L}_i^{(k)})\right) + e_k(\|\mathbb{L}_i^{(k+1)}\| - \|\mathcal{Q}(\mathbb{L}_i^{(k)})\|)^2 \\ &= D_k\left(\mathbb{L}_i^{(k+1)},\ \mathcal{Q}(\mathbb{L}_i^{(k)})\right) \\ &\quad +(e_k - e_{k-1})(\|\mathbb{L}_i^{(k+1)}\| - \|\mathcal{Q}(\mathbb{L}_i^{(k)})\|)^2, \end{aligned}$$

therefore,

$$\begin{aligned} e_{k+1} &\leq \sum_{i=1}^{N} D_k\left(\mathbb{L}_i^{(k+1)},\ \mathcal{Q}(\mathbb{L}_i^{(k)})\right) + (e_k - e_{k-1})\delta_k \\ &= d_k + (e_k - e_{k-1})\delta_k \\ &\leq e_k + (e_k - e_{k-1})\delta_k, \end{aligned} \tag{5.4.5}$$

where

$$\begin{aligned} \delta_k &= \sum_{i=1}^{N}(\|\mathbb{L}_i^{(k+1)}\| - \|\mathcal{Q}(\mathbb{L}_i^{(k)})\|)^2 \\ &= \sum_{i=1}^{N}(\|\mathbb{L}_i^{(1)}\| - \|\mathcal{Q}(\mathbb{L}_i^{(k)})\|)^2, \end{aligned}$$

since for all k, we have $\|\mathbb{L}_i^{(k)}\| = \|\mathbb{L}_i^{(1)}\|$. From the relation

$$e_{k+1} \leq e_k + (e_k - e_{k-1})\delta_k,$$

we see that if $e_k \leq e_{k-1}$, then $e_{k+1} \leq e_k$. Thus, if we could choose e_0 such that it is bigger than e_1 then the non-negative real number sequence $\{e_k\}$ will be non-increasing and bounded below, and so it must be convergent.

It can be seen directly that a necessary condition that $e_{k+1} \leq e_k$ is $\delta_k \leq 1.0$. In fact,

$$\begin{aligned} e_{k+1} &= \sum_{i=1}^{N} D\left(\mathbb{L}_i^{(k+1)},\ \mathcal{Q}(\mathbb{L}_i^{(k+1)})\right) + e_k\delta_{k+1} \\ &\geq e_k\delta_{k+1}. \end{aligned}$$

If $e_{k+1} \leq e_k$, then $e_k \geq e_k\delta_{k+1}$. Thus $\delta_{k+1} \leq 1$, since in general $e_k \neq 0$.

On the other hand, if $\delta_1 = 0$, e_1 will be independent to e_0 and e_0 can always be chosen to be larger than e_1, since in practice, e_1 will always be bounded. When δ_1 has an upper bound δ_0 less than 1.0, then from the inequality

$$e_1 \leq \sum_{i=1}^{N} D\left(\mathbb{L}_i^{(1)},\ \mathcal{Q}(\mathbb{L}_i^{(1)})\right) + e_0\delta_0,$$

it can be seen that $e_1 < e_0$ if $e_0 > \Delta/(1-\delta_0)$, where Δ is an upper bound for all possible sums

$$\sum_{i=1}^{N} D\left(\mathbb{L}_i^{(1)},\ \mathcal{Q}(\mathbb{L}_i^{(1)})\right)$$

relating to different initial values for e_0. Since δ_k is just a sum of differences of lengths of corresponding line segments, its upper bound should be very small if all line segments in $\mathcal{P}$ can be well approximated by corresponding line segments in $\mathcal{Q}$. Even when the intra-operative data are not good enough, an upper bound less than 1.0 can always be set to δ_1 uniformly by re-scaling the data sets such that δ_1 is less than one. Therefore, under appropriate scaling, the real positive number sequence $\{e_k\}$ will always be decreasing and thus convergent.

The iterative triangle patch algorithm(ICT) proceeds much the same way. The ICT algorithm can be obtained by replacing a line segment with a triangle patch in the above algorithm. As in the ICL algorithm, the following dynamic weighted distance is used in the ICT algorithm instead of (5.3.4).

$$D_k(\mathbb{T}_1, \mathbb{T}_2) = D(\mathbb{T}_1, \mathbb{T}_2) + e_{k-1}(d_1^2 + d_2^2 + d_3^2), \tag{5.4.6}$$

where e_k is the error sum in the k^{th} step defined in a similar way as (5.4.2) and d_1, d_2, d_3 are differences in lengths of three corresponding edges of the two triangle patches.

For each triangle patch in $\mathcal{P}$, the ICT algorithm will search for the closest triangle patches in the pre-operative model data and use the triangle patch registration method given in section 5.3 to compute the rotation and translation in each iteration. The ICT algorithm works more robustly than the ICP and the ICL algorithms, but it spends more time in computing the closest triangle patches.

5.5 Comparison of results

The iterative line segment registration algorithm and triangle patch registration algorithm have been tested and compared with the ICP algorithm. Three different geometric objects have been considered in our experiments: a set of space line segments, a space curve and a surface [see Figures 5.1, 5.2, 5.3]. Having sampled the first data set (corresponding to pre-operative data), the object is randomly transformed by a rotation and a translation and then the second data set (corresponding to intra-operative data) is sampled. The algorithms ICP, ICL and ICT are applied to the two data sets to estimate the transformation.

It is shown that when the second data set (intra-operative data) is just the transformation of a subset of the first data set, the estimated transformation obtained with the ICL and the ICT algorithm will be exactly the true transformation performed,

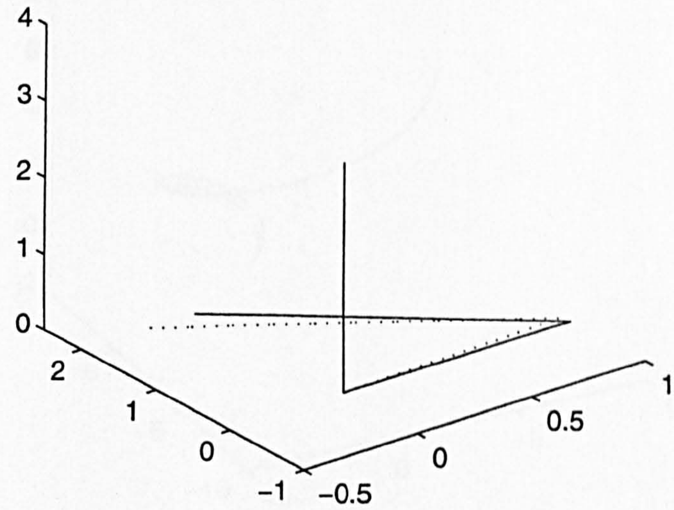

Figure 5.1: Line segments matching

and the number of iterations is just two or three in most cases. When the second data set is not the transformation of a subset of the first one, it takes more iterations to converge and the transformation estimated may not necessarily be very close to the true transformation, though it is close in most cases. To test the stability of the ICL and the ICT algorithms, data sets of different sizes are sampled for each object, with various orientations. The results of the experiment show that both the ICL and the ICT algorithms are much more stable than the ICP algorithm.

Obviously, the speed of convergence depends largely on the size and quality of the data used, but it is also affected by the initial value assigned to e_0. Too large an e_0 tends to result in a slower convergence, and too small an e_0 may be more likely to result in convergence to the wrong transformation. In our test, the ideal e_0 is chosen between 10^{10} to 10^{30}.

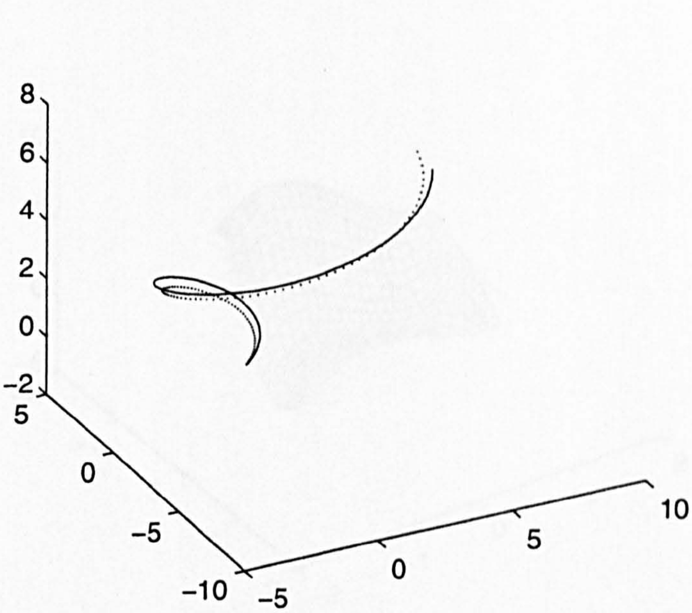

Figure 5.2: Curve matching

Much of the experiment has been done to compare the ICL and the ICT algorithms with the ICP algorithm. As we know, the ICP algorithm is very sensitive to the initial orientation of the object. Thus it is natural to ask whether the newly developed methods are any better. The first way to show such a stability for these algorithms is to compute the probability of success for a series of experiments. In the experiment, we say that a matching process is successful if both the error between the estimated rotation and the true rotation and the error between estimated translation and the true translation are less than the given threshold. Tables 5.1 to 5.3 show the percentage of successes over 500 runs of these three algorithms with different data sets using the ICP, the ICL and the ICT algorithms. The figures are obtained by setting the threshold to be 0.2 for the line segment object and curve object and 0.4 for the surface object. The closeness between two rotation matrices is defined as the Frobenius norm of their difference and the closeness of two translations is measured

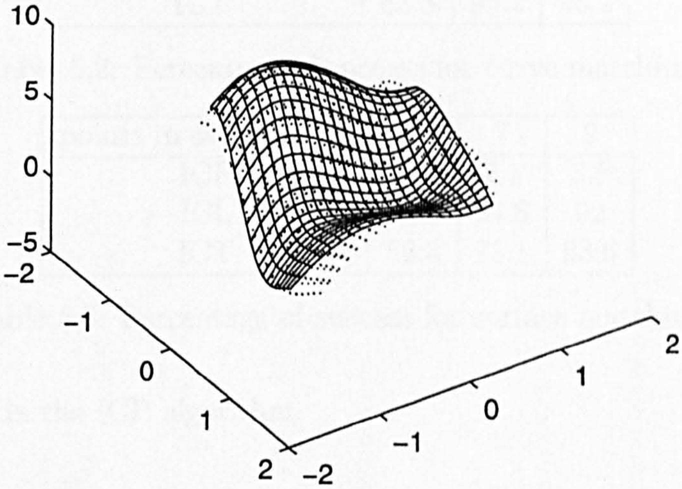

Figure 5.3: Surface matching

by the norm of the difference between the two vectors which define the translations. The number of points in the model data is in the range of 60 to 1000 and the number of points in the second data set is just between 4 and 6.

points in second data	4	5	6
ICP	37.4	35.1	14.3
ICL	78.8	91.8	99.6
ICT	100.0	100.0	100.0

Table 5.1: Percentage of success for line segments matching

It can be see from the figures in Table 5.1 to Table 5.3 that the ICL and the ICT algorithm are much less sensitive to the initial orientations of the object. In all cases, the rate of success is more than 70% for line segments and curve with the ICL and the ICT algorithms compared with less than 20% percent with the ICP algorithm. The figures in these tables also show that with the increase of number of points in the second data sets, the ICL and the ICT become more and more robust, while there is

points in second data	4	5	6
ICP	10.6	15.1	16.7
ICL	74.8	93.8	98.7
ICT	83.3	95.2	98.2

Table 5.2: Percentage of success for curve matching

points in second data	5	7	9
ICP	3.1	2.1	3.5
ICL	43.5	64.8	92
ICT	52.6	75.1	93.6

Table 5.3: Percentage of success for surface matching

no such tendency in the ICP algorithm.

The robustness of the ICL and ICT algorithms can also be shown by computing the average errors between true transformation and estimated transformation as well as errors in object matching. Tables 5.4 to 5.12 show that the average errors and the relevant standard deviations for both ICL and ICT algorithms are much smaller than those obtained with the ICP algorithm.

	mean			std		
	Rotation	translation	distance	Rotation	translation	distance
ICP	1.6626	0.8648	1.6755	1.2116	0.7507	1.3280
ICL	0.1253	0.1177	0.2219	0.4234	0.2752	0.2630
ICT	0.1271	0.1411	0.4071	0.0739	0.0536	0.1282

Table 5.4: Differences in rotation, translation and distance for line segments matching with 4 points in the test data set

The ICL and the ICT algorithms are designed for problems where only very few points in the intra-operative data set are used, normally between 4 and 10, while the points in pre-operative data can be huge so that the object surface can be fully

	mean			dev		
	Rotation	translation	distance	Rotation	translation	distance
ICP	1.6995	0.8022	1.9817	1.2215	0.7215	1.4866
ICL	0.2091	0.1675	0.3550	0.6382	0.4082	0.6115
ICT	0.1391	0.1497	0.5039	0.0544	0.0563	0.1650

Table 5.5: Differences in rotation, translation and distance for line segments matching with 5 points in the test data set

	mean			dev		
	Rotation	translation	distance	Rotation	translation	distance
ICP	2.0121	1.0921	3.0404	1.0776	0.7007	1.1353
ICL	0.1861	0.1484	1.0994	0.6016	0.3727	0.5039
ICT	0.1036	0.1388	1.3265	0.0639	0.0445	0.1370

Table 5.6: Differences in rotation, translation and distance for line segments matching with 6 points in the test data set

described by the data. Thus it is natural to ask whether the ICL and the ICT algorithms are feasible for this problem in practice as its computational complexity will be $O(NM^2)$ in searching for the closest line segments in the ICL algorithm and $O(NM^3)$ in searching for the closest triangle patches in the ICT algorithm. Generally speaking, the ICL and the ICT algorithms should not be directly applied to the data. Some preprocessing for pre-operative data is needed. For example, geometric invariants under rotation and translation can be considered to remove those unlikely pairs of line segments and triangle patches. Before applying the ICL algorithm, we could first select possible line segments in the pre-operative data by considering whether a line segment has a similar length to some line segment in the intra-operative data. Let P_1, P_2 be two points in the intra-operative data, and let their corresponding position in pre-operative space be Q_1, Q_2. If the maximum distance between the neighboring elements in the pre-operative data is δ, then the difference between $||P_2 - P1||$ and

	mean			dev		
	Rotation	translation	distance	Rotation	translation	distance
ICP	2.0372	5.1269	3.3219	1.1156	3.3631	2.0127
ICL	0.6733	1.6610	1.4082	1.0407	2.4570	1.5658
ICT	0.3409	0.7728	0.7065	0.8503	1.9574	1.5621

Table 5.7: Differences in rotation, translation and distance for curve matching with 4 points in the test data set

	mean			dev		
	Rotation	translation	distance	Rotation	translation	distance
ICP	1.9769	4.9529	5.0027	1.1702	3.3979	3.0511
ICL	0.4656	1.0212	1.3700	0.9080	2.3405	1.7400
ICT	0.0816	0.1876	0.7006	0.2118	0.4901	1.0160

Table 5.8: Differences in rotation, translation and distance for curve matching with 5 points in the test data set

$||Q_2 - Q_1||$ cannot be larger than 2δ. In this way, the number of line segments considered in the ICL will be greatly reduced. As the triangle patch algorithm provides more geometric invariants, more information can be used to select the possible triangle patches used in the ICT algorithm. This not only solves the problem of the feasibility of using the ICL algorithm and the ICT algorithm, but also increases their robustness.

As far as the computing time is concerned, it depends not only on the size of the data sets, but also on their quality. The total computation time consists of the time used in selecting the possible geometrical structure and the time used to estimate the transformation based on the selected geometrical structure. Increasing the number of points in matching data sets will only increase the time used for selecting the possible geometrical structure, but not necessarily the iteration times, and thus not necessarily the overall computation time. When the number of points in the model data set is not too large, the time used mainly depends on the convergence speed.

	mean			dev		
	Rotation	translation	distance	Rotation	translation	distance
ICP	1.2347	3.7532	4.1150	1.1352	3.4012	2.5323
ICL	0.1366	0.2655	0.8782	0.3423	0.9151	1.1480
ICT	0.0751	0.1666	0.8626	0.0814	0.2128	0.9346

Table 5.9: Differences in rotation, translation and distance for curve matching with 6 points in the test data set

	mean			dev		
	Rotation	translation	distance	Rotation	translation	distance
ICP	2.1157	2.5736	4.0252	0.8368	1.5330	1.3005
ICL	0.6150	1.3876	2.9965	0.2207	0.3765	0.6095
ICT	0.6751	0.9171	0.9749	0.8347	1.3943	0.1733

Table 5.10: Differences in rotation, translation and distance for surface matching with 5 points in the test data set

But for a large model data set, the computation speed will be determined mainly by the time used for selecting possible geometrical structures. As for the ICT algorithm, the computation time really depends on the value of the threshold for selecting the possible triangle patches. A big threshold may result in thousands of triangle patches being selected and it may take hours to finish the matching process. The number of triangle patches selected for an appropriate threshold should around 512 times the number of triangle patches selected from the second data sets. Figures 5.4 and 5.5 show the computation times in using the ICL and the ICT algorithms to match two data sets from a curve, where N represent the number of points in the second data set. The code is written in C++ and is run under Microsoft Windows NT with a CELERON 400MHz processor. As can be seen from Figure 5.4, the ICL algorithm uses less than a minute to match two data sets with the number of points in the model data ranging from 100 to 1000. As far as the ICT algorithm is concerned, its speed is also not as slow as expected as we can see from Figure 5.5. However, if we

	mean			dev		
	Rotation	translation	distance	Rotation	translation	distance
ICP	2.2706	2.9904	3.4057	0.8760	1.9573	1.3736
ICL	0.5996	0.6705	1.7966	0.8221	0.2151	0.6559
ICT	0.3100	0.5402	0.9695	0.6925	1.1445	0.3149

Table 5.11: Differences in rotation, translation and distance for surface matching with 7 points in the test data set

	mean			dev		
	Rotation	translation	distance	Rotation	translation	distance
ICP	2.3007	2.9269	4.5968	1.8298	3.0929	1.5027
ICL	0.1116	0.1506	2.0263	0.1184	0.0640	0.5962
ICT	0.2065	0.2761	2.8030	0.1009	0.0822	1.0202

Table 5.12: Differences in rotation, translation and distance for surface matching with 9 points in the test data set

ignore the selecting procedure, it does take a few hours to establish the match when the number of points in the model data is larger than 700. Again, it is worth to note that the computation time does not necessarily monotonically increase with the increase of the number of point in the model data, though it does in general situation. As can be seen from Figure 5.4, when the number of points in the second data set is eight, the computation time is much shorter when the number of point in the model data is 400. Same phenomenon can be observed from the figure when the number of points in the second data set is four, where shorter computation time is used when the number of points in the model data is 400, or 700, or 1000. The phenomenon can also be observed from Figure 5.5 for the ICT algorithm, where the number of points in the second data set is eight and the number of points in the model data set is 800. This phenomenon occurs when the second data set happen to be the result of a rigid transformation of a subset of the model data set (as we have mentioned earlier) or when the second data set can be well approximated by a subset of the

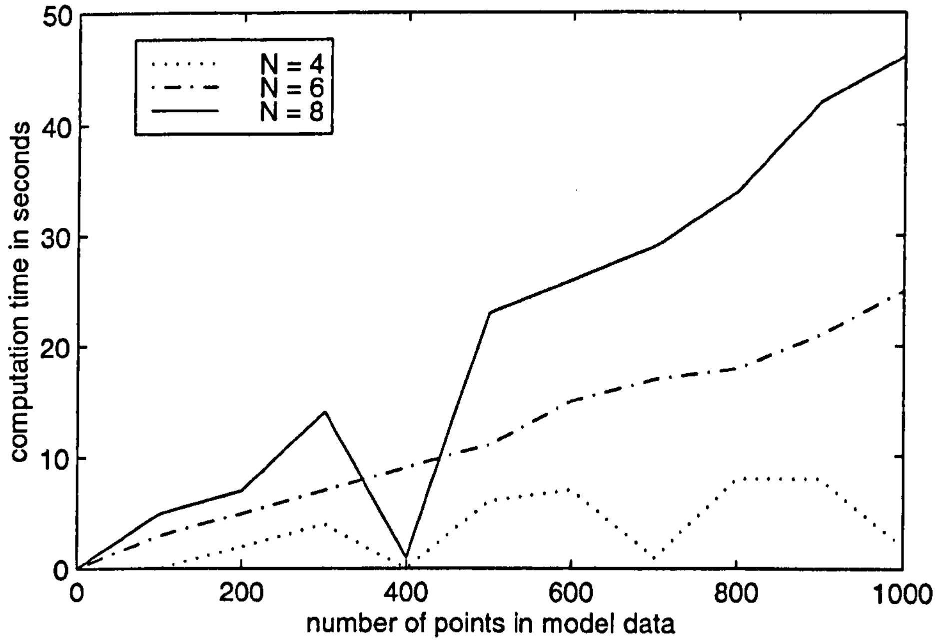

Figure 5.4: Computation time vs. the number of points in model data for the ICL algorithm

model data set with respect to a rigid transformation. The good thing about the ICL and ICT algorithms is that they can still give good estimates even if the model data set is a bit sparse, while the ICP cannot. Therefore, in some cases, we can first use the ICL or the ICT algorithm to find a good initial solution (one or two iterations) with a subset of model data. This initial solution can then be further tuned by the ICP algorithm. However, to fully investigate the performances of the ICL and ICT algorithms, more experiments need to be done with noisy data or with data from surfaces of some actual geometric objects sampled with various scanners. This will provide a more complete comparison with the ICP algorithm.

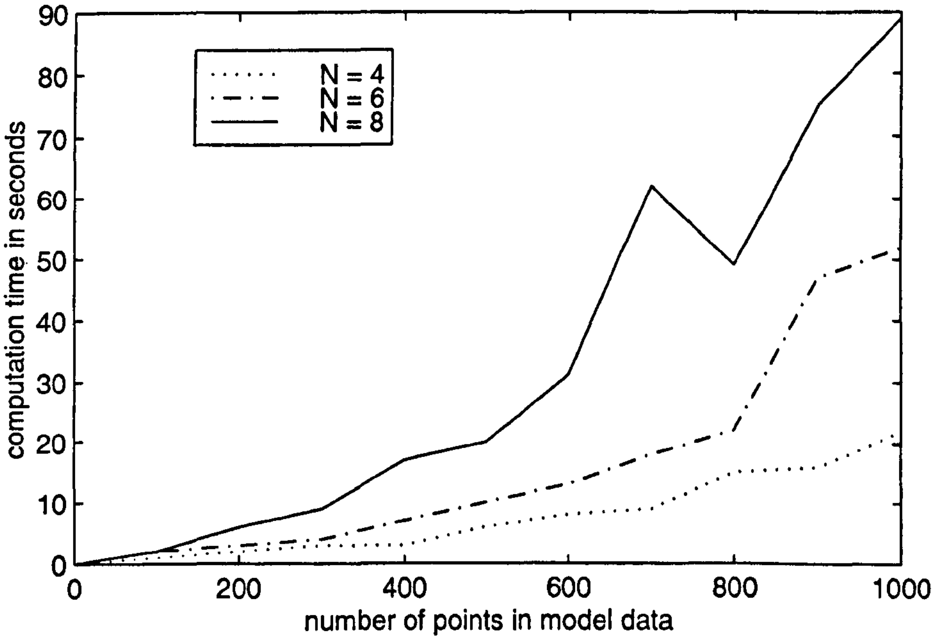

Figure 5.5: Computation time vs. the number of points in model data for the ICT algorithm

Chapter 6

Least squares ellipsoid fitting

6.1 Introduction

In the previous chapter, we developed the ICL and the ICT algorithms for surface data matching. They can be applied directly to scattered data sets when the preoperative data are fine enough to describe the surface of a human bone very well. However, when preoperative data are sparse or too noisy, they have to be fitted with a smooth surface in order to blur the gaps between the data points or to filter out some noise from the data. Basically, two kinds of shape can be fitted with a set of 3D points, either parametrically or implicitly. A parametric surface is a map S from $\mathbb{R}^2$ to $\mathbb{R}^3$ and can be expressed as $S(u,v) = (x(u,v), y(u,v), z(u,v))$. An implicit surface is the zero contour of a map f from $\mathbb{R}^3$ to $\mathbb{R}$, i.e., it can be expressed as $f(x,y,z) = 0$. Both types of fitting have their advantages and disadvantages. However, in computer graphics, research on data fitting has long been dominated by parametric shapes due to their highly desirable properties in drawing, tesselating, subdividing and bounding. Recently, fitting data with implicitly represented curves and surfaces has attracted increasing attention as implicitly represented shapes have advantages over parametric shapes in several ways. Firstly, they can tell directly whether a point is inside or outside the shape. Secondly, the surface normals can be easily computed. Thirdly, the most commonly used geometric shapes such as spheres, cylinders, ellipsoids can be represented very easily with implicit surfaces. Finally, the value of an implicit function at a point can be used to measure the distance from the point to the surface, which is the motivation of our work in the next three chapters.

In our registration problem, the type of surface chosen to fit the data depends on what kind of registration technique is adopted. If we use the ICP or the ICL algorithm, parametric surfaces are preferred as the computation of closest points on the preoperative model surface is easier for parametric surfaces than for implicit surfaces. However, when we follow the idea of 'Head and Hat' matching, an implicitly represented surface is much preferred as the value of the implicit function at a point can be used to approximate the distance from the point to the surface of the function represented. As we pointed out in Section 1.2, the conventional 'Head and Hat' surface matching algorithm [69] is computationally expensive since the distance from a point to the polyhedron surface constructed from the preoperative data is difficult to compute. However, when the shape is modeled implicitly, the distance from a point to the surface can be simply approximated with either the value of the function or the pseudo-Euclidean distance of the implicit surface. Furthermore, the gradient of the shape can be easily computed and thus the optimization procedure in searching for the rotation and translation can be carried out much more efficiently. In this thesis, we will focus on investigating implicit surface fitting as we want to develop an algorithm for our region to region matching following the idea of 'Head to Hat' matching, where to know approximately the distance from a point to the surface of the preoperative model is more important.

In this chapter, we will only consider how to fit 3D data with an implicitly represented ellipsoid. More general implicit surface fitting techniques will be discussed in the next chapter.

6.1.1 Why ellipsoid fitting ?

Fitting an ellipsoid to a set of 3D scattered points occurs in the area of pattern recognition, machine vision, 3D graphics and spatial data analysis. Ellipsoids, though rather restrictive for representing 3D shapes in general, are the only bounded and central quadrics that can provide information on the centre and orientation of an object. Ellipsoids have been used as an effective means for shape representation for quite some time[8] [70]. In the area of computer graphics visualization, ellipsoids can be fitted to a set of 3D points for the purpose of object segmentation [102]. In medicine, ellipsoids can be fitted to human organs for data visualization [84] or for better appreciation of the clinical outcome after operations [51] [89]. Ellipsoid fitting is also required in spatial data analysis such as geoid estimation, and geodetic datum acquisition [27]. Ellipsoid fitting can also serve as an initial step for many shape representation problems[19] [20] [21]. In the shape matching problem, when domain knowledge is available, ellipsoids can be fitted to different regions of interest of an object, so that a good initial matching can be established based on these ellipsoids. In all these cases, an ellipsoid is the preferred fit for a given data set.

6.1.2 A brief survey of ellipsoid fitting

The best-ellipsoid fit problem has long been investigated [71], where an ellipsoid with minimum volume surrounding the given data set is required. In [45], reconstructing an ellipsoid from its orthogonal silhouette projections has been considered. In this chapter, we consider how to fit a set of scatter points from the surface of a 3D object with an ellipsoid specifically. Fitting an algebraic surface with scatter 3D points has been discussed widely and some excellent work has been found in [94] [95] [93] [46] [47] [72]. However, most of these fitting techniques are not ellipsoid specific. Though the techniques for fitting bounded algebraic surfaces presented in [46] and [95] can be used to fit an ellipsoid, the fitting results are unsatisfactory when the data are noisy. In theory, the conditions that guarantee that a quadratic surface is an ellipsoid have been well investigated and explicitly stated in analytic geometry, and can be found in most analytic geometry textbooks. In fact, when its leading form is positive definite, the solution of the quadratic equation must be bounded and thus represents an ellipsoid [9]. Therefore, fitting an ellipsoid to a given data set can be performed directly in several ways. One way is to use multiple constraints to constrain the leading form of a fitted quadratic equation to be positive definite, which leads to a nonlinear optimization problem with multiple constraints. Another way is to parameterize the coefficients of the leading form as suggested in [46] and [95]. In this case, to guarantee that the fitted quadratic surface is an ellipsoid, we need only ensure that the corresponding coefficient matrix of the leading form of the quadratic equation is nonsingular, which turns out to be a nonlinear optimization problem with one constraint. As with all nonlinear optimization problems, the optimization procedures based on the above two methods often stop at a local minimum and cannot guarantee an optimal solution. Both of these approaches have the drawback that they are very sensitive to initial values. In this chapter, we first investigate whether there is a simple constraint that can determine whether a general quadratic in three variables represents an ellipsoid. As is well known, the discriminant which determines whether the 2D quadratic equation

$$ax^2 + 2bxy + cy^2 + 2dx + 2ey + f = 0$$

represents an ellipse is $ac - b^2 > 0$. However, to the best of our knowledge, no similar quadratic form exists for the 3D ellipsoid . In this chapter, we show that the discriminant for an ellipsoid is of similar form for a large group of ellipsoids. For the sake of conciseness, the three principal axes of an ellipsoid are described here as long, median, and short. Generally, a quadric surface is defined as the locus of points such that their coordinates satisfy the most general equation of the second degree in three variables, namely

$$ax^2 + by^2 + cz^2 + 2fyz + 2gxz + 2hxy + 2px + 2qy + 2rz + d = 0. \qquad (6.1.1)$$

Then it is well known that this equation represents either a central quadric or a paraboloid, including their degeneracies[90]. Let

$$\Delta = \begin{vmatrix} a & h & g \\ h & b & f \\ g & f & c \end{vmatrix}, \qquad (6.1.2)$$

$$I = a + b + c, \qquad (6.1.3)$$

and

$$J = ab + bc + ac - f^2 - g^2 - h^2. \qquad (6.1.4)$$

It is known that they are invariant under rotation and translation and that equation (6.1.1) represents an ellipsoid if and only if $J > 0$, and $I\Delta > 0$ [38].

It is shown further in this chapter that when

$$4J - I^2 > 0,$$

equation (6.1.1) must represent an ellipsoid. On the other hand, for an ellipsoid, when the length of the short axis is at least half of that of the long axis, then we must have $4J - I^2 > 0$. However, $4J - I^2 > 0$ is only a sufficient condition to guarantee that an equation of second degree in three variables represents an ellipsoid, but it is not necessary. Therefore the ellipsoids that satisfy the condition $4J - I^2 > 0$ are just a subset of the whole ellipsoid family. Generally, for any ellipsoid, there must exist a real number $\alpha \geq 4$ such that $\alpha J - I^2 > 0$. However, when $\alpha > 4$, the condition that $\alpha J - I^2 > 0$ cannot guarantee that a quadric surface is an ellipsoid. A simple search procedure is then suggested for fitting an ellipsoid in general. In most cases, the fitting will just be a one step fitting except in some extreme cases, and thus it is almost a direct fitting technique. In section 2, we first induce the constraint $4J - I^2 > 0$ and discuss some of its properties. Our fitting algorithm is described in section 3. In section 4, some experimental results are provided to show the efficiency and robustness of the algorithm.

6.2 The discriminant of the ellipsoid

Equation (6.1.1) can be rigidly transformed into the standard form

$$Ax^2 + By^2 + Cz^2 + Px + Qy + Rz + D = 0, \qquad (6.2.1)$$

where A, B, C are the roots of the characteristic equation

$$u^3 - Iu^2 + Ju - \Delta = 0. \tag{6.2.2}$$

Lemma 6.2.1. *$I^2 \geq 3J$, and $I^2 = 3J$ if and only if equation (6.1.1) represents a sphere.*

Proof. As both I and J are invariants, we need only consider the inequality for the standardized ellipsoid (6.2.1). From Cauchy's inequality, we have

$$(AB + BC + AC)^2 \leq (A^2 + B^2 + C^2)^2,$$

and thus

$$AB + BC + AC \leq A^2 + B^2 + C^2.$$

This is equivalent to $I^2 \geq 3J$. Also from Cauchy's inequality, the above inequality becomes an equality if and only if $A = B = C$. □

To ensure that equation (6.2.1) always represents an ellipsoid, none of the three roots of the cubic equation (6.2.2) can be zero and all of them must have the same sign. To find the basic form of the discriminant, we first consider a special case when the cubic equation (6.2.2) has a double root, that is, the ellipsoid is considered to be a revolution of an ellipse. Let u_1, u_2 be the roots of equation (6.2.2) with u_1 a double root. According to Vieta's root theorem for the cubic equation, we have

$$\begin{cases} 2u_1 + u_2 & = & I \\ u_1(u_1 + 2u_2) & = & J. \end{cases}$$

It follows that

$$u_1 = \frac{1}{3}(I \pm \sqrt{\delta}), \qquad u_2 = \frac{1}{3}(I \mp 2\sqrt{\delta}), \tag{6.2.3}$$

where $\delta = I^2 - 3J \geq 0$(see Lemma 6.2.1)

To ensure that both u_1 and u_2 are positive, from (6.2.3) we must have either

$$4J - I^2 > 0 \tag{6.2.4}$$

or

$$J > 0. \tag{6.2.5}$$

Thus, when the cubic equation (6.2.2) has two positive roots with one of them being a double root, either inequality (6.2.4) or (6.2.5) must be satisfied. Obviously, $J > 0$ is implied in $4J - I^2 > 0$. Conversely, if cubic equation (6.2.2) has a double root, all its roots must have same signs when $4J - I^2 > 0$ (or equivalently, $|I| > 2\sqrt{\delta}$), as can be seen from (6.2.3).

In general, we have

Proposition 6.2.2. *For equation (6.1.1),*

1. *If (6.2.4) is satisfied, then (6.1.1) must represent an ellipsoid.*

2. *If (6.1.1) represents an ellipsoid, then J must be positive.*

Proof. (1) To show the first conclusion, we need only to show that it is correct for the standardized quadric surface (6.2.1), as $4J - I^2$ is a rigid transformation invariant. Now we show that the signs of A, B, C in equation (6.2.1) are all the same. Assuming that the signs are not the same, without loss of of generality, we take $A > 0$, $B > 0$, and $C = -|C| \leq 0$. Then $I = A + B - |C|$, $J = AB - A|C| - B|C|$ and

$$4J - I^2 = -(A - B)^2 - C^2 - 2A|C| - 2B|C| \leq 0.$$

This contradicts $4J - I^2 > 0$.

(2) This result is not new, and can be shown in many ways. In fact, when the cubic equation (6.2.2) has three positive roots, its derivative

$$3u^2 - 2Iu + J = 0$$

must have two positive roots and this is equivalent to $J > 0$. □

To sum up, we see that $4J - I^2 > 0$ is a sufficient condition to ensure that a general surface of second order is an ellipsoid but it is not a necessary condition. However, $J > 0$ is a necessary condition but not sufficient. More explicitly, an ellipsoid does not necessarily satisfy $4J - I^2 > 0$ but must satisfy $J > 0$.

Since for any ellipsoid, the corresponding value of J will always be positive and thus there always exists a real number $\alpha > 0$ such that $\alpha J - I^2 > 0$. However when $\alpha > 4$, a quadric surface that satisfies $\alpha J - I^2 > 0$ is not necessarily an ellipsoid.

Proposition 6.2.3. *The maximum value of $\alpha > 0$ for which a quadric surface can be an ellipsoid, when $\alpha J - I^2 > 0$, is $\alpha = 4$.*

Proof. To proof this we need only show that for any $\alpha > 4$, there always exists a quadric such that $\alpha J - I^2 > 0$ but that the quadric is not an ellipsoid. Let $\alpha = 4 + \epsilon$, where $\epsilon > 0$. Then for any $\gamma > 0$, consider the equation:

$$x^2 + y^2 - \gamma z^2 = 1. \tag{6.2.6}$$

For this equation, $I = 2 - \gamma$, $J = 1 - 2\gamma$, and

$$(4 + \epsilon)J - I^2 = \epsilon - 2(2 + \epsilon)\gamma - \gamma^2.$$

It is obvious that $\epsilon - 2(2 + \epsilon)\gamma - \gamma^2$ will be positive when γ is sufficiently small. This shows that $\alpha J - I^2 > 0$ cannot ensure that the given equation is an ellipsoid whenever $\alpha > 4$. □

At the end of the section, we investigate the geometric meaning of the invariant $4J - I^2$ for an ellipsoid. Define $\rho = \frac{4J-I^2}{\{A^2+B^2+C^2\}}$, where $A, B, C > 0$ are the roots of (6.2.2). Then ρ is an invariant under rotation and translation. It can be shown that $|\rho| \leq 1$, and $\rho = 1$ if and only if $A = B = C$ when the equation (6.1.1) defines a sphere. Further, It can be observed that $\rho > -1$, and when one of the roots tends to infinity or when two of the roots tend to zero, the value of ρ tends to -1. In this case the corresponding ellipsoid will be flat shaped. Thus, we can see that the value of ρ can be used to measure the roundness of an ellipsoid. The bigger the value of ρ, the more nearly spherical the quadric. Conversely, the smaller the ρ is, the flatter or the longer and thinner the ellipsoid is. The following proposition states precisely when an ellipsoid satisfies $4J - I^2 > 0$.

Proposition 6.2.4. *For an ellipsoid*

$$Ax^2 + By^2 + Cz^2 + D = 0,$$

if we assume that $A \geq B \geq C > 0$, *then*

1. *If* $B > \frac{1}{4}A, C > \frac{1}{4}A$, *then* $4J - I^2 > 0$;
2. *If* $B \leq \frac{1}{4}A, C \leq \frac{1}{4}A$, *then* $4J - I^2 \leq 0$;
3. *If* $B \geq \frac{1}{4}A, C < \frac{1}{4}A$, *we can write* $C = (\frac{1}{4} - c)A$, $B = (\frac{1}{4} + \alpha c)A$ *for some positive number* $0 < c < \frac{1}{4}$ *and some number* $\alpha, 0 \leq \alpha \leq \frac{3}{4c}$. *Then* $4J - I^2 > 0$ *if and only if*
$$f(\alpha) = \frac{2(\alpha - 1)}{(\alpha + 1)^2} > c$$

Proof. As the sign of $4J - I^2$ will not be changed if we multiply equation (6.2.1) by a positive number, for simplicity, we assume without loss of generality that $A = 1$, $0 < B \leq 1$, and $0 < C \leq 1$.

1. If $B > \frac{1}{4}, C > \frac{1}{4}$, let $B = \frac{1}{4} + b, C = \frac{1}{4} + c$ for $b, c \in (0, \frac{3}{4}]$, then

$$\begin{aligned} 4J - I^2 &= 2(B+C) - (B-C)^2 - 1 \\ &= 2(b+c) - (b-c)^2 \geq 2(b+c) - |b-c| > 0, \end{aligned}$$

   as $b, c \in (0, \frac{3}{4}]$.

2. If $B \leq \frac{1}{4}, C \leq \frac{1}{4}$, it is obvious that $4J - I^2 = 2(B+C) - (B-C)^2 - 1 \leq 0$.

3. If $B \geq \frac{1}{4}, C < \frac{1}{4}$, Let $C = \frac{1}{4} - c$, $B = \frac{1}{4} + \alpha c$ for some c, $\frac{1}{4} > c > 0$ and some $\alpha, 0 \leq \alpha \leq \frac{3}{4c}$, then

$$4J - I^2 = 2(B+C) - (B-C)^2 - 1 = 2(\alpha - 1)c - (1+\alpha)^2 c^2,$$

   and $4J - I^2 > 0$ is thus equivalent to $f(\alpha) = \frac{2(\alpha-1)}{(\alpha+1)^2} > c$. Note that the function f has a maximum value of $\frac{1}{4}$ when $\alpha = 3$. Thus, for any value of c between 0 and 1/4, there always exists an α such that $f(\alpha) > c$. In fact, let $\alpha_1 = \frac{1-c-\sqrt{1-4c}}{c}$, then $1 < \alpha_1 < 3$ for $c \in (0, \frac{1}{4})$, and for any $\alpha \in (\alpha_1, 3]$, we have $f(\alpha) > c$.

□

In the proof of the third conclusion, let $B_1 = (\frac{1}{4} + \alpha_1 c)A$, and let $B = (\frac{1}{4} + \alpha c)A$ for $\alpha \in (\alpha_1, 3]$, then $C \leq B_1 \leq B \leq A$. It is direct that $4J > I^2$, for any $c \in (0, \frac{1}{4})$. This means that no matter how great the length of the long axis of an ellipsoid is, the corresponding ellipsoid satisfies $4J - I^2 > 0$ if only the length of the median axis is short enough. This, plus the first conclusion of the above property, means that the only kinds of ellipsoid that cannot be characterized by $4J - I^2 > 0$ are those that are very slender and very flat.

6.3 The ellipsoid fitting method

As we do not have one simple overall discriminant to determine whether a fitted equation is an ellipsoid in general, it is difficult to give a direct fitting as in the 2D case. However, when the fitted data are from a fairly spherical ellipsoid, direct fitting is possible using the same principle as in the 2D case[33]. When the data are from a very flat or a very slender surface of an object, we can specifically fit an ellipsoid with a simple search procedure. In this section, we first discuss the direct fitting for data from a fairly spherical surface. Then, we discuss how to fit an ellipsoid to data from a general surface.

6.3.1 Direct least squares ellipsoid fitting

In this section, we first discuss how to fit data to a quadratic surface (6.1.1) under the constraint $kJ > I^2$, where k is positive. When $k = 4$, the fitted shape will be an ellipsoid. This fitting strategy is quite similar to the work presented in [33] for fitting an ellipse in the 2D case. Let $\{p_i(x_i, y_i, z_i)\}_{i=1}^{n}$ be the set of points to which an ellipsoid needs to be fitted. For each point $p_i(x_i, y_i, z_i)$, let

$$\mathbf{X}_i = (x_i^2, y_i^2, z_i^2, 2y_iz_i, 2x_iz_i, 2x_iy_i, 2x_i, 2y_i, 2z_i, 1)^T.$$

For a given equation (6.1.1), let

$$\mathbf{v} = (a, b, c, f, g, h, p, q, r, d)^T,$$

our least squares fitting problem based on algebraic distance with the constraint

$$kJ - I^2 > 0$$

can be formulated as:

$$min\|D^T\mathbf{v}\|^2 \qquad \text{subject to} \qquad kJ - I^2 = 1, \tag{6.3.1}$$

where D is the design matrix of size $10 \times n$ defined as $D = (\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_n)$. If C is the 10×10 matrix defined below as:

$$C = \begin{pmatrix} -1 & \frac{k}{2}-1 & \frac{k}{2}-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{k}{2}-1 & -1 & \frac{k}{2}-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{k}{2}-1 & \frac{k}{2}-1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -k & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -k & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -k & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{6.3.2}$$

$kJ - I^2 = 1$ can be written as $\mathbf{v}^T C\mathbf{v} = 1$ and the constraint minimization problem (6.3.1) becomes that of solving a set of equations using Lagrange multipliers:

$$\begin{cases} DD^T\mathbf{v} = & \lambda C\mathbf{v} \\ \mathbf{v}^T C\mathbf{v} = & 1. \end{cases} \tag{6.3.3}$$

Note that matrix C has eigenvalues $\{k-3, -\frac{k}{2}, -\frac{k}{2}, -k, -k, -k, 0, 0, 0, 0\}$. Using the same inference as given in [33], we state that equation (6.3.3) has only one solution when DD^T is positive definite and $k > 3$, which is the general eigenvector associated with the unique positive eigenvalue $k-3$ of the general eigenvalue system $DD^T\mathbf{v} = \lambda C\mathbf{v}$.

Note that $c_{ij} = 0$ when $i > 6$ or $j > 6$ for the elements of matrix C. We write

$$DD^T = \begin{pmatrix} S_{11} & S_{12} \\ S_{12}^T & S_{22} \end{pmatrix}, \qquad \mathbf{v} = \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{pmatrix}, \tag{6.3.4}$$

where matrices S_{11}, S_{12}, S_{22} are of size 6×6, 6×4, 4×4 and vector $\mathbf{v}_1$, $\mathbf{v}_2$ are of size 6 and 4. Let C_1 be the top left 6×6 matrix of C, then the eigen system (6.3.3) becomes

$$(S_{11} - \lambda C_1)\mathbf{v}_1 + S_{12}\mathbf{v}_2 = 0 \tag{6.3.5}$$

$$S_{12}^T\mathbf{v}_1 + S_{22}\mathbf{v}_2 = 0 \tag{6.3.6}$$

When S is positive definite, S_{22} will be nonsingular and from (6.3.6),

$$\mathbf{v}_2 = -S_{22}^{-1}S_{12}^T\mathbf{v}_1.$$

Substituting this equation for $\mathbf{v}_2$ in equation (6.3.5), we obtain the general eigen system:

$$(S_{11} - S_{12}S_{22}^{-1}S_{12}^T)\mathbf{v}_1 = \lambda C_1\mathbf{v}_1. \tag{6.3.7}$$

If $S_{11} - S_{12}S_{22}^{-1}S_{12}^T$ is positive definite, let $\mathbf{u}_1$ be the eigenvector associated with the only positive eigenvalue of the general eigen system(6.3.7), and let $\mathbf{u}_2 = -S_{22}^{-1}S_{12}^T\mathbf{u}_1$, then $\mathbf{u} = (\mathbf{u}_1^T, \mathbf{u}_2^T)^T$ will be the eigenvector associated with the only positive eigenvalue for the general eigen system (6.3.3). The advantage of this way of computing is that the matrices involved in the generalized eigen system (6.3.7) are smaller and thus give better accuracy.

Remark 7. *1. When $k > 3$, the matrix C_1 is nonsingular. Solving the general eigen system (6.3.7) becomes solving the following ordinary eigen system*

$$C_1^{-1}(S_{11} - S_{12}S_{22}^{-1}S_{12}^T)\mathbf{v}_1 = \lambda\mathbf{v}_1 \tag{6.3.8}$$

2. *When DD^T is positive definite, it can be shown immediately that $S_{11}-S_{12}S_{22}^{-1}S_{12}^T$ is also positive definite [2]. If $(\lambda, \mathbf{v})$ is the solution of the general eigen system (6.3.7), then $\mathbf{v}^T(S_{11} - S_{12}S_{22}^{-1}S_{12}^T)\mathbf{v} > 0$. Therefore, we must have*

$$\lambda \mathbf{v}^T C_1 \mathbf{v} > 0.$$

On the other hand, if $\mathbf{v}$ satisfies the constraint that $\mathbf{v}^T C_1 \mathbf{v} = 1 \geq 0$, the eigenvalue λ corresponding to the solution must be positive and thus the solution is unique.

3. *Direct spherical fitting. From Lemma 6.2.1, a sphere can be characterized by $3J = I^2$. When $k = 3$, C_1 will be singular. It can be shown that the reciprocal eigen system of (6.3.7)*

$$(S_{11} - S_{12}S_{22}^{-1}S_{12}^T)^{-1}C_1\mathbf{v}_1 = \lambda \mathbf{v}_1 \tag{6.3.9}$$

will have only one non-negative eigenvalue 0 when $S_{11} - S_{12}S_{22}^{-1}S_{12}^T$ is positive definite, and the vector $\mathbf{v}_1$ corresponding to the nontrivial solution for problem (6.3.1) will be the eigenvector of eigen system (6.3.9) associated with eigenvalue 0.

4. *When S_{22} is almost singular, S_{22}^{-1} can be replaced with its generalized inverse $S_{22}^{\dagger}$ and the corresponding solution to $\mathbf{v}_2$ for (6.3.6) can be replaced with $-S_{22}^{\dagger}S_{12}^T\mathbf{v}_1$, which has the following properties [62]:*

   (a) It is the least squares solution of $S_{22}\mathbf{v}_2 = -S_{12}^T\mathbf{v}_1$ when there is no solution.

   (b) It is the unique solution when there is but one solution.

   (c) It is the minimum norm solution when there are an infinite number of solutions.

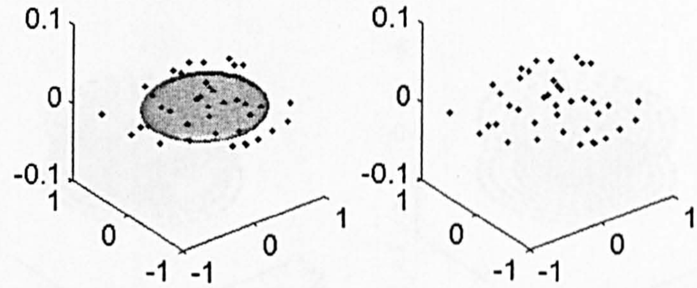

Figure 6.1: Fitting an ellipsoid with planar points

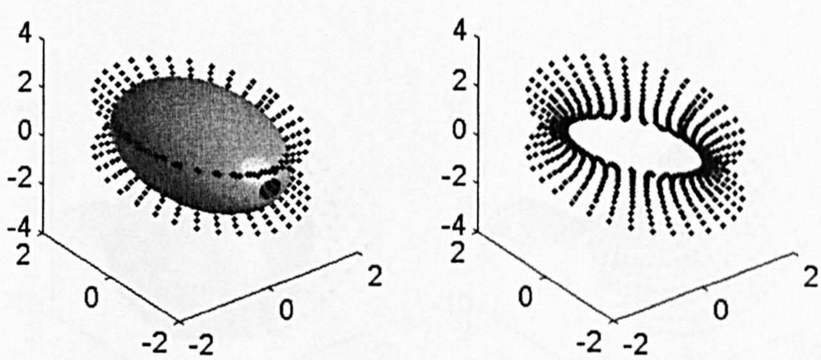

Figure 6.2: Fitting an ellipsoid with points from a hyperboloid of single sheet

As already pointed out, $4J - I^2 > 0$ is just a sufficient condition to confirm that a quadric is an ellipsoid, but it is not a necessary condition. Thus, the above ellipsoid fitting achieved by setting $k = 4$ is 'best' just for those ellipsoids that satisfies $4J - I^2 > 0$, which is only a subset of the whole ellipsoid family. In practice, when data are from a roughly spherical ellipsoid, the algorithm achieves the fit in only one step. However, when data cannot be well represented by a nearly spherical surface, above one step ellipsoid fitting based on $k = 4$ is not enough. In the next section, we will give a method to cope with the problem using a simple searching procedure to find an appropriate value of k.

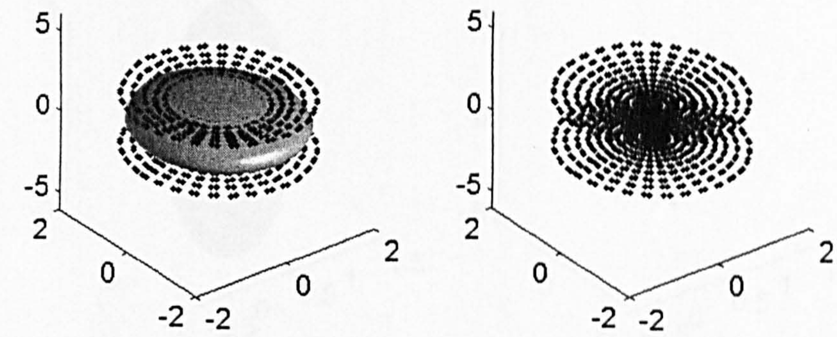

Figure 6.3: Fitting an ellipsoid with points from a hyperboloid of double sheets

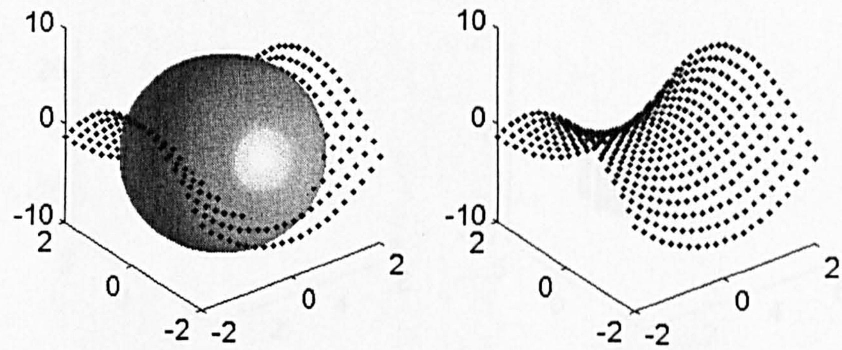

Figure 6.4: Fitting an ellipsoid with points from a hyperbolic paraboloid

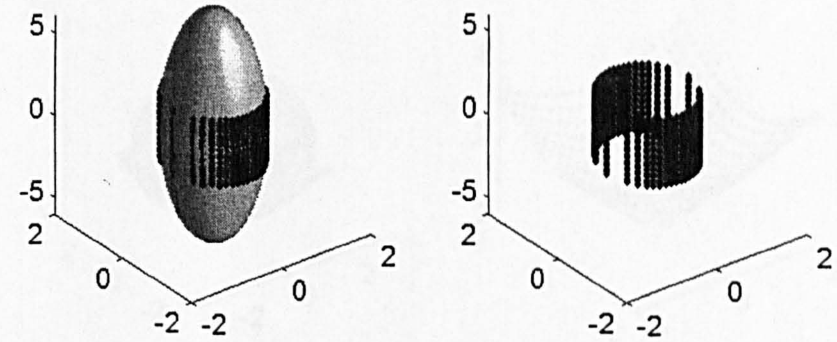

Figure 6.5: Fitting an ellipsoid with points from an elliptic cylinder

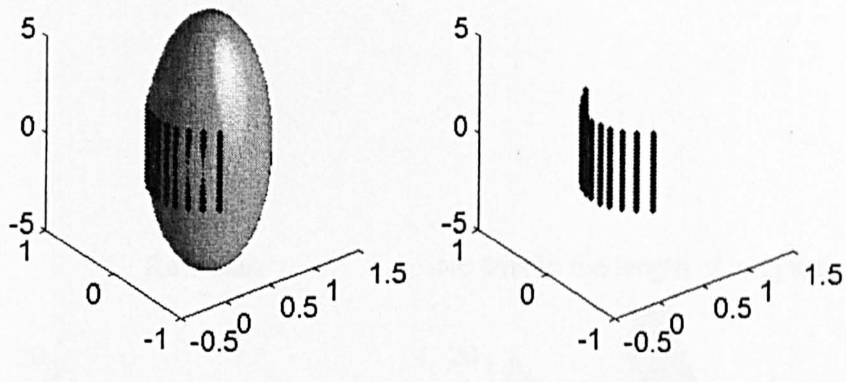

Figure 6.6: Fitting an ellipsoid with points from a parabolic cylinder

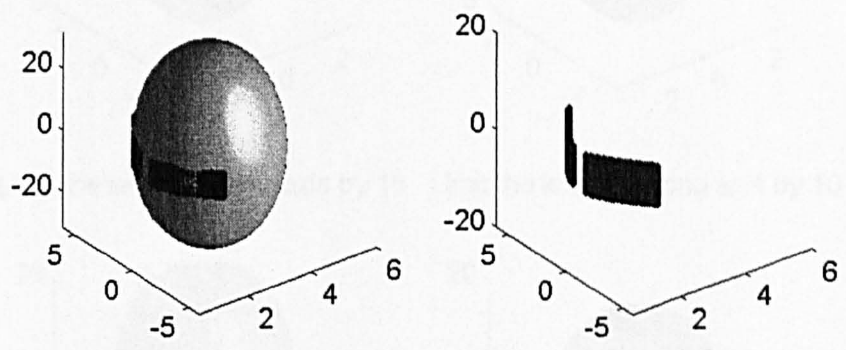

Figure 6.7: Fitting an ellipsoid with points from a hyperbolic cylinder

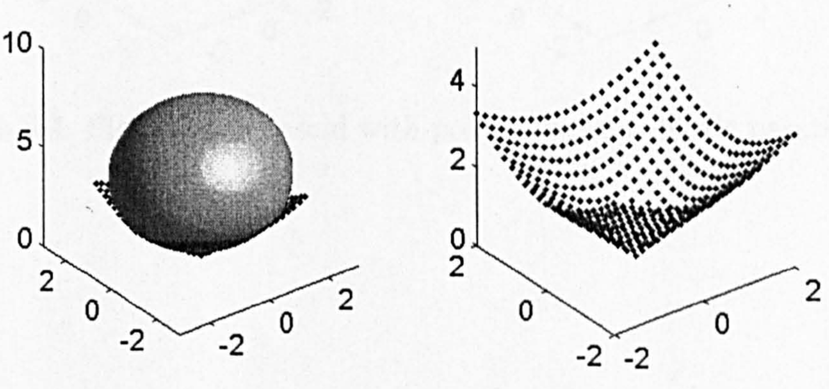

Figure 6.8: Fitting an ellipsoid with points from a cone

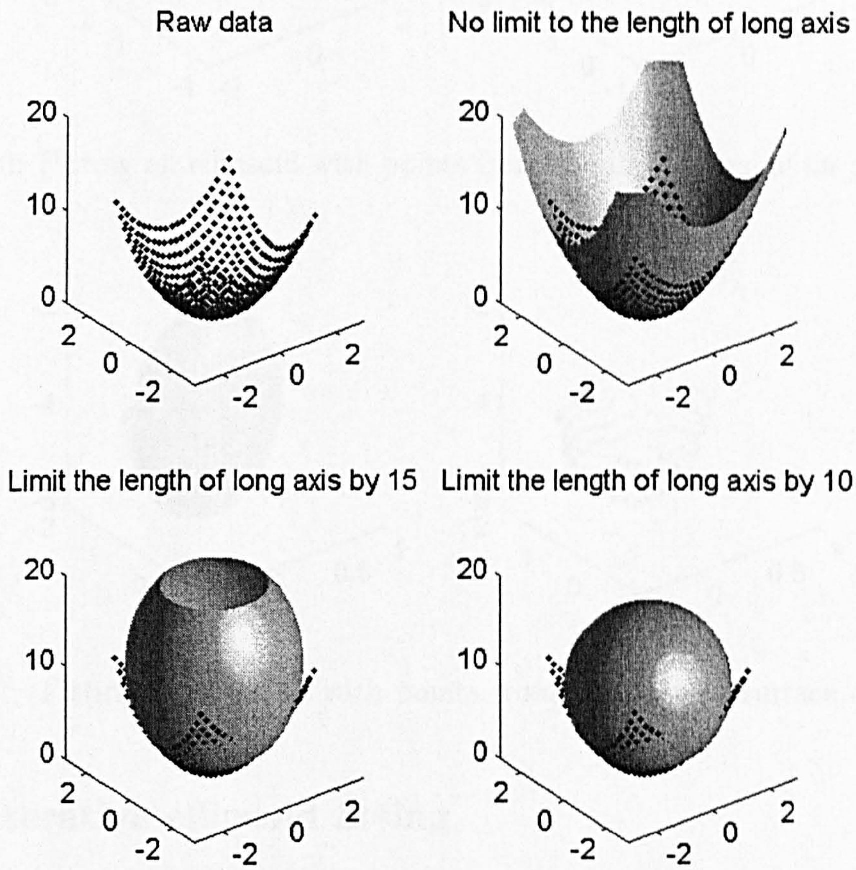

Figure 6.9: Fitting an ellipsoid with points from an elliptic paraboloid .

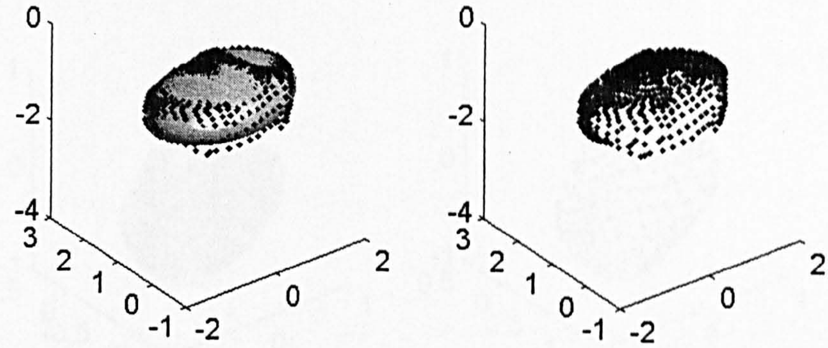

Figure 6.10: Fitting an ellipsoid with points from the top surface of an actual tibia

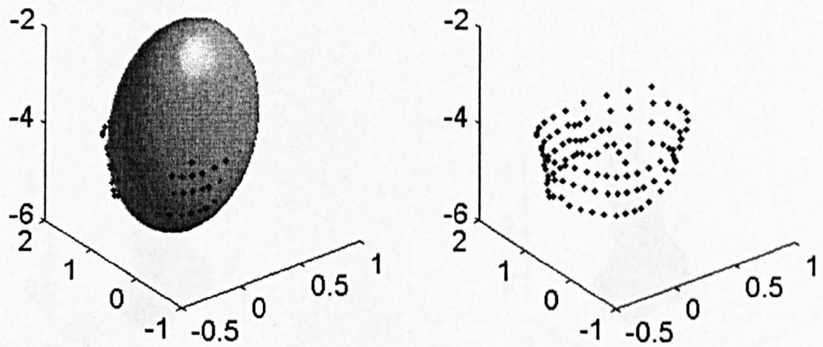

Figure 6.11: Fitting an ellipsoid with points from the bottom surface of an actual tibia

6.3.2 Iterative ellipsoid fitting

When data cannot be well described by an ellipsoid with the property $4J - I^2 > 0$, a large fitting error may be encountered. In this case we need to enlarge the family of ellipsoids by choosing a real number $k > 4$ and use $kJ - I^2 = 1$ to constrain the fitting. The problem is that the fitted equation under the constraint $kJ - I^2 > 0$ might not be an ellipsoid when $k > 4$. Note that when a real number $k' > k$, a quadric satisfying $kJ > I^2$ must also satisfy $k'J > I^2$. Thus, the fitting based on

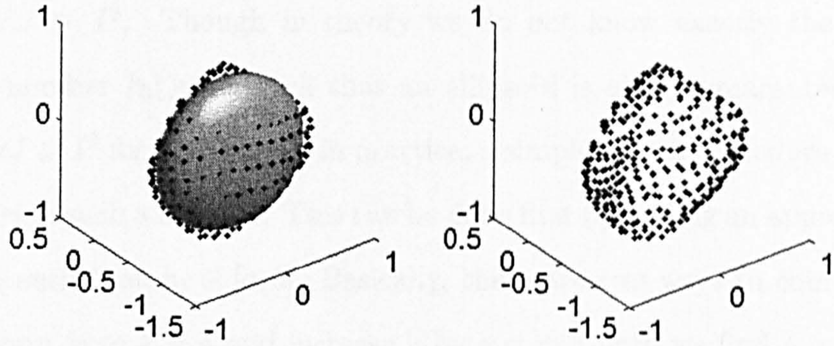

Figure 6.12: Fitting an ellipsoid with points from the surface of an actual patella

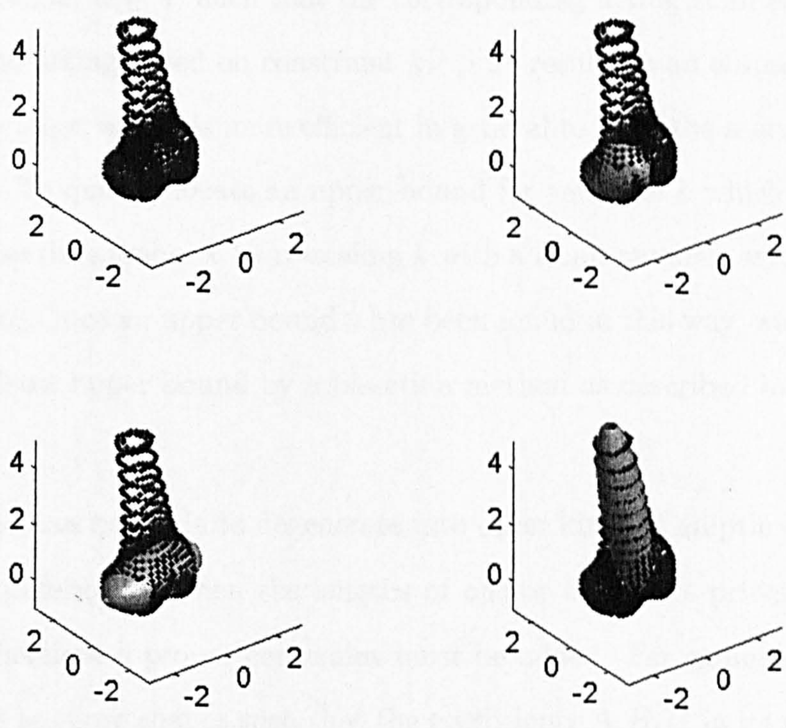

Figure 6.13: Fitting an ellipsoid to different parts of an actual femur

the constraint $k'J - I^2 = 1$ will be preferable to the fitting based on $kJ - I^2 = 1$ if the fitted quadric surface is an ellipsoid, since there are more ellipsoids satisfying condition $k'J > I^2$. Though in theory we do not know exactly the location for the largest number $k_0(\geq 4)$, such that an ellipsoid is always guaranteed under the constraint $kJ > I^2$ for $4 \leq k < k_0$, in practice, a simple search procedure can be easily devised to trace such a number. This can be done first by finding an appropriate range $[a, b]$ for k_0, such that $k_0 \in [a, b]$. Basically, there are two ways to compute it. One way is to begin from $k = 4$ and increase k by a step δ until we find a number b such that the fitting with constraint $bJ > I^2$ will no longer result in an ellipsoid or b is large enough such that almost all ellipsoids satisfy the condition $bJ > I^2$. The other way is to begin with a number b as large as possible and decrease it by a step δ until we get to a number $a(\geq 4)$ such that the corresponding fitting is an ellipsoid. As in most cases, the fitting based on constraint $kJ > I^2$ results in an ellipsoid even when k is extremely large, and it is more efficient in general to start the search from a very large number. To quickly locate an upper bound for values of k which are plausible, we can decrease the number k by re-scaling k with a small number, say, let the new k be $k/2$ or $k/10$. Once an upper bound b has been found in this way, we could further compute the least upper bound by a bisection method as described in the following algorithm.

An ellipsoid can be made to degenerate into other kinds of elliptic quadrics, such as an elliptic paraboloid, when the lengths of one or two of its principal axes tend to infinity. Therefore a proper constraint must be added. For example, an ellipsoid can be defined as those shapes such that the coefficients A, B, C in its standard form given in (6.2.1) are all positive and larger than a preset positive number ϵ. Once such a constraint is given, $4J > I^2$ might not guarantee a quadric to be an ellipsoid in the sense of our restricted definition of an ellipsoid. But $\alpha J > I^2$ for $\alpha > 3$ will certainly suffice. The algorithm is described as follows:

Algorithm 6.3.1. Iterative ellipsoid specific fitting

1. Set k to a very large positive number b, say $b = 10^8$.

2. Use direct ellipsoid fitting method stated in section 6.3.1 to find the solution for equation:

$$DD^T\mathbf{v} = \lambda\mathbf{v} \qquad \text{subject to} \qquad \mathbf{v}^T C\mathbf{v} \;=\; 1$$

   where $\mathbf{v}C\mathbf{v} = 1$ corresponding to $kJ - I^2 = 1$.

3. If the fitting is an ellipsoid, STOP; else

   (a) While the fitting is not an ellipsoid and $k \geq 3$, replace k by $k/2$ and fit the data with the constraint $kJ - T^2 = 1$.

   (b) Set $a = max(k, 3), b = 2k$.

   (c) set k=(a+b)/2 and fit the data with the constraint $kJ > I^2$. If the fitted shape is an ellipsoid set a=k; else set $b = k$.

   (d) If $|a - b|$ is less than a preset tolerance, STOP, otherwise go to 3c.

The number of iterations depends on the data. When the data can be described well by an ellipsoid, it is just a one step fitting. In the worst case for a data set containing about 1000 points, it just take less than a second to find the optimal solution. For a data set that cannot be properly described by an ellipsoid, we have a trade off between approximation accuracy and the size of the fitted ellipsoid. For

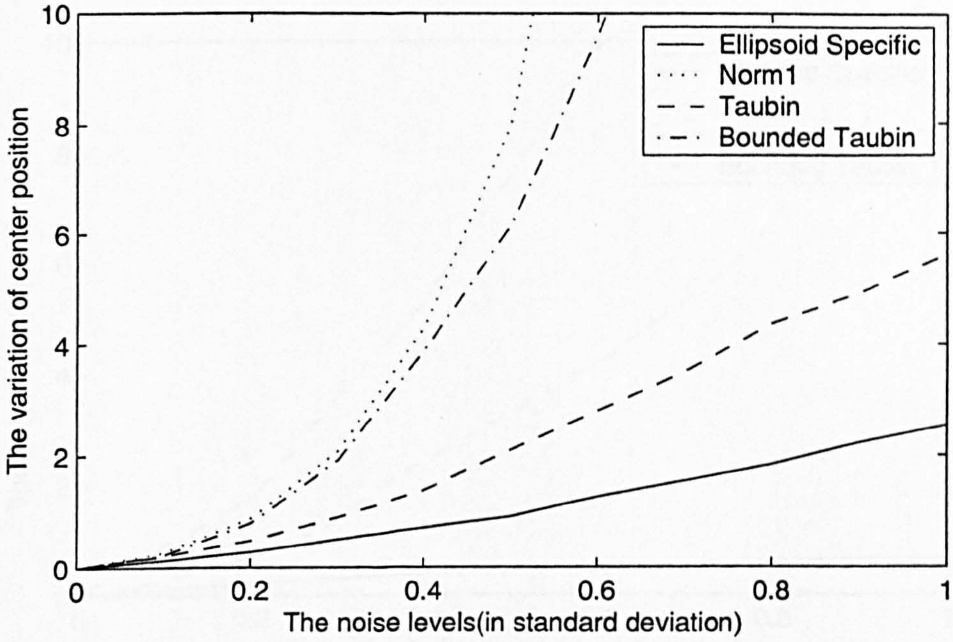

Figure 6.14: The variation of centre position vs. noise level

example, when the data is precisely from an elliptic paraboloid (see Figure 6.9), if we want the fitting to be as accurate as possible, the fitted ellipsoid can be very big, which may not be what we expect in some cases. To limit the size within an appropriate range, we can discard those ellipsoids whose size is larger than given bounds. Figure 6.9 shows the fitting results for the same data set with different limitations on the length of the long axis. The above algorithm has been tested with data from various kinds of shape such as the hyperboloid, paraboloid, cone, cylinder, and an actual tibia, patella, and femur. The fitted results are quite satisfactory.

6.4 Experimental results

In this section we demonstrate the robustness of our fitting method by comparing it with other least-squares fitting approaches. For the sake of simplicity, the fitting

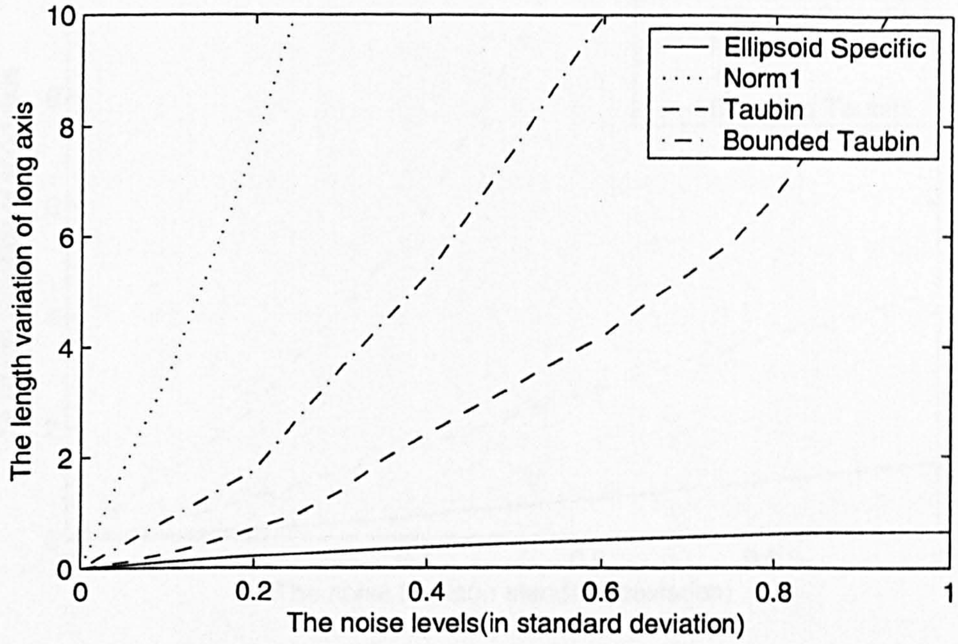

Figure 6.15: The length error in the long axis vs. noise level

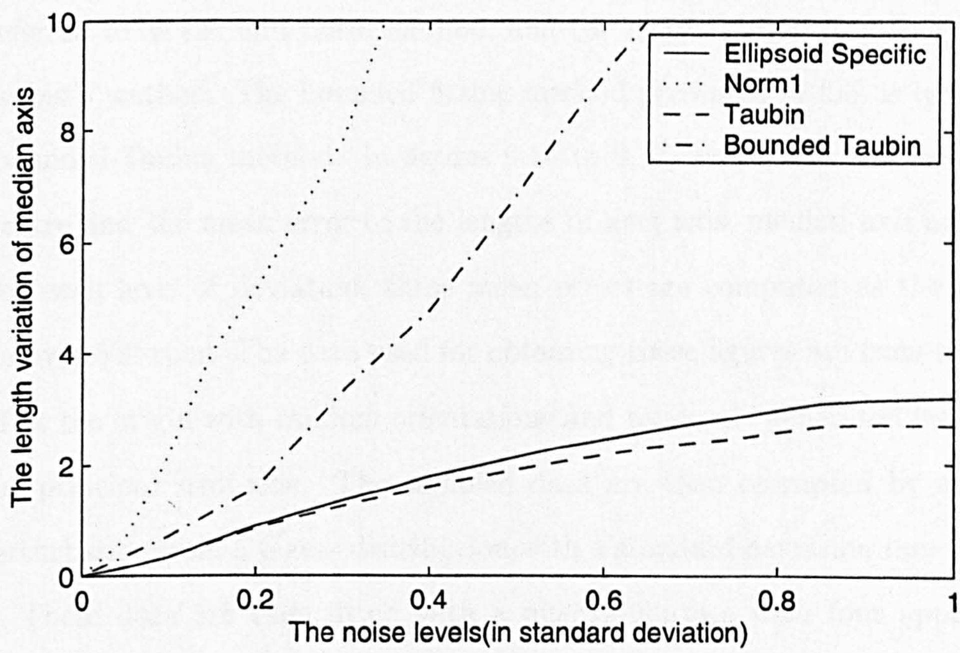

Figure 6.16: The length error in the median axis vs. noise level

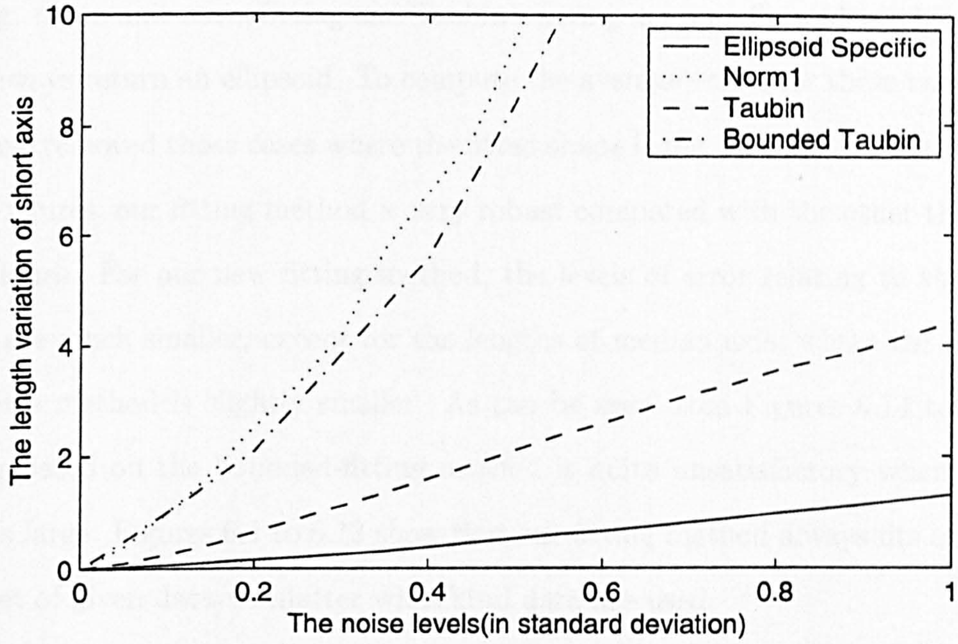

Figure 6.17: The length error in the short axis vs. noise level

algorithm using the constraint that the sum of squares of the estimated coefficients is one is referred to as the unit norm method, and the method given in [94] is referred to as Taubin's method. The bounded fitting method presented in [95] is referred to as the bounded-Taubin method. In figures 6.14 to 6.17, we present the mean error of the centre and the mean error of the lengths of long axis, median axis and short axis. For each level of deviation, these mean errors are computed as the average variation over 500 runs. The data used for obtaining these figures are from ellipsoids centered at the origin with random orientations and randomly generated lengths for the three principal semi-axis. The sampled data are then corrupted by adding a small perturbation from a Gauss distribution with a standard deviation ranging from 0 to 1. These data are then fitted with a quadric surface with four approaches: our ellipsoid specific fitting, unit norm fitting, Taubin's fitting and bounded-Taubin fitting. Since unit norm fitting and Taubin's fitting are not ellipsoid specific, they will not always return an ellipsoid. To compute the average errors for these two methods, we have removed those cases where the fitted shape is not an ellipsoid. As is shown in these figures, our fitting method is very robust compared with the other three fitting techniques. For our new fitting method, the levels of error relating to the levels of noise are much smaller, except for the lengths of median axis, where the error from Taubin's method is slightly smaller. As can be seen from Figures 6.14 to 6.17, the fitting based on the bounded-fitting method is quite unsatisfactory when the error level is large. Figures 6.1 to 6.13 show that our fitting method always fits an ellipsoid to a set of given data no matter what kind data are used.

6.5 Summary

In this chapter, we have developed an algorithm for fitting an ellipsoid. We showed that when data are from a surface that is not very slender or very flat, the given method is just a one step direct fitting. For data from a general shape, or when the data are from a surface that cannot be well described by an ellipsoid, the data can be fitted with an ellipsoid by a simple searching procedure starting with a very large number k. In most cases, this will also be a one-step fitting except for some extreme instances, for example, when the data are precisely from a quadric surface other than an ellipsoid. Usually, it takes less that a second to complete the fitting for data containing about 1000 points. The experiment presented in section 6.4 demonstrates that the method is very robust in the presence of noise.

Chapter 7

Constructive Implicit Fitting

7.1 Introduction

In the previous chapter, we investigated how to fit a cloud of points with an ellipsoid. However, ellipsoid surfaces are too simple to represent general shapes. In this chapter, we will further explore how to fit a cloud of points with a more general implicit shape.

By far the most common fitting techniques are concerned only with algebraic curves and surfaces. These represent a geometric shape by a single algebraic equation, because of its simple form and good algebraic properties. The problem is that higher degree algebraic curves and surfaces are often unbounded and multiple sheeted, while lower degree algebraic curves and surfaces are too simple to represent general shapes. In this chapter, we present a constructive method for fitting an implicit function to a set of scattered points by using the gate functions, which are smooth functions from $\mathbb{R}^n$ to the interval $[0, 1]$ which take values close to 1 on some simply connected region $D \subset \mathbb{R}^n$ and take value close to 0 outside D. With this fitting technique, the data are first partitioned by means of geometric primitives into small data sets such that each subset of the data can be well described by a low degree algebraic surface. These simple algebraic shapes are first confined locally with gate functions corresponding to the geometric primitives which partition the data. They are summed together to obtain the overall fitting. More precisely, the general form of this type of implicit shapes can be put in the following way:

$$\sum_{i=1}^{n} g_i(P) f_i(P) = 0,$$

where each $f_i(P)$ is a low degree polynomial and $g_i(P)$ is the gate function that confines $f_i(P)$. Examples are included to demonstrate that the fitting is very satisfactory.

In most cases, implicit surfaces are used to construct solid geometrical objects. In this situation, the only requirement is that the constructed objects should be similar to a known object in shape but not necessarily an approximation to the actual surface of the given object. Thus the shapes can be created rather arbitrarily. They don't have to be exactly the same as the known objects to be simulated. However, in many applications a very accurate implicit shape needs to be reconstructed from a cloud of points sampled from the surface of a given object. Fitting an implicit surface to a set of points proves to be a difficult task. The implicit functions fitted are very unpredictable in shape and very uncontrollable in size, even for shapes represented by very simple algebraic equations, such as the cubic algebraic surfaces. Secondly, it is very likely that the fitted shape might not well represent the actual shape from which the data have been sampled. It is not difficult to fit a shape such that every data point is very close to it, but some parts of the fitted shape might not be close to any data points. In addition, the shape fitted might be much more complex than the actual shape that the data represents. The fitted shape might be unbounded and multiple sheeted. It may have holes or knots. Savchenko et al. in [82] have given a brief survey on the implicit fitting problem and present an approach to interpolate scattered points, a method which is computationally expensive when the number of data points is large. Furthermore, this method cannot guarantee that the zeros of a computed algebraic difference are all close to the given data set. Some other implicit fitting techniques either are mainly concerned with solid reconstruction, without considering how to represent the shape with an implicit function, or simply use a global optimization procedure to obtain a function representation for given data. This function representation is often far from accurate [58] [64] [66] [98]. Some other good work on implicit fitting can be found in [6] [13] [23] [46] [47] [79] [94] [95] [103].

One feasible way to overcome the difficulties of implicit fitting is to break the given data set into several subsets, such that the data of each subset can be well approximated with a simple implicit function. The overall shape can be obtained by combining these simple surfaces. The methods for combining a set of implicit shapes have often been referred to as shape blending. One way of blending a set of implicit shapes has been purely based on set theoretic operations. Generally, this technique cannot guarantee the expected smoothness in a composite shape, while its generalized smooth blending methods cannot often preserve the local properties of those locally fitted shapes very well [25] [34]. In this chapter, we present a constructive method to fit implicit curves and surfaces to a set of data points using gate functions. Several examples are given to demonstrate the effectiveness of the method. The advantage of this kind of fitting method is that any part of the fitted shape will be close to some points in the data set. In addition, the overall approximation accuracy can be attained for the data set if the data partitioning is fine enough. Furthermore, the fitted shape can always be bounded by confining the fitted shape with a bounded gate.

The basic idea of this technique is the use of gate functions. Let D be a region in real Euclidean space $\mathbb{R}^n$, and let

$$\chi_D(P) = \begin{cases} 1 & P \in D \\ 0 & P \notin D \end{cases} \tag{7.1.1}$$

be the characteristic function of region D. Let $f(P)$ be a function defined in $\mathbb{R}^n$. Then the function $\chi_D(P)f(P)$ has the following property:

$$\chi_D(P)f(P) = \begin{cases} f(P) & P \in D \\ 0 & P \notin D. \end{cases} \tag{7.1.2}$$

For each function $f(P)$, once it is multiplied by the function $\chi_D(P)$, the effect of function $f(P)$ outside the region D is diminished as the resulting function value will be zero, while the value of $f(P)$ inside the region D will not be changed. Thus, if a set of points can be well approximated by an implicit function $f(P) = 0$, then the equation

$$\chi_D(P)(f(P) + \delta) - \delta = 0,$$

where $\delta > 0$, will be exactly the original equation $f(P) = 0$ itself on region D and thus the original shape on region D is kept unchanged. However, the shape outside the region D has been removed as on the outside of region D, the value of function $\chi_D(P)(f(P) + \delta) - \delta = -\delta < 0$.

The problem with function $\chi_D(P)$ is that it is not continuous. The combined shape using this kind of function might be fragmented. To overcome the difficulty, smooth gate functions are constructed. As can be seen from the following section, a smooth gate function can be seen as a smooth approximation to the characteristic function of a set such as the one given in (7.1.1).

In this chapter, we first discuss how to construct smooth gate functions in one dimension. Then we investigate how to extend them to higher dimensions. This is followed by the applications of gate functions in establishing our constructive explicit and implicit curve and surface fitting method. Since the gate functions used in this paper are not polynomials, it should be noted that the implicit curves or surfaces fitted are no longer algebraic.

7.2 Gate functions

In this section, we first introduce the unit step function [106] (also called Heaviside's step function) and its smooth approximations. We then investigate how to construct general smooth gate functions with the smooth unit step functions.

Gate functions are usually defined on one dimensional real Euclidean space. However, this concept has been used with broader meanings in the following sections. More explicitly, we have following definition:

Definition 7.2.1. Function $g : \mathbb{R}^n \to [0,1]$ is called a gate function on real Euclidean space $\mathbb{R}^n$ if for each $\alpha \in [0,1]$, the α-level set

$$g_\alpha = \{P \in \mathbb{R}^n : g(P) \geq \alpha\}$$

is a simply connected set in $\mathbb{R}^n$.

In this section, we mainly discuss how to construct smooth gate functions used in our constructive implicit fitting method.

7.2.1 The unit step function and its smooth approximations

A discontinuous step function defined as

$$H(t) = \begin{cases} 0 & t < 0 \\ \frac{1}{2} & t = 0 \\ 1 & t > 0 \end{cases} \tag{7.2.1}$$

is called the unit step function and for any real number a, the step function $H(t-a)$ is called a unit step function at a and is denoted by $H_a(t)$.

Obviously, $H(t)$ is a gate function. The smooth gate functions that approximate a unit step function can be constructed in the same way as constructing what have been called "rising cutoff functions" by Wickerhauser in [108]. In wavelet theory, rising cutoff functions are used to construct localized trigonometric functions which are then combined into a library of orthonormal bases. It is found that this kind of function can be extended to higher dimensions to approximate 2D or 3D set characteristic functions smoothly.

In [108], a rising cutoff function has been defined as a complex function $r(t)$ with a real argument t that satisfies the conditions that:

$$|r(t)|^2 + |r(-t)|^2 = 1 \tag{7.2.2}$$

for all $t \in R$, and that

$$r(t) = \begin{cases} 0, & \text{if } t < -1 \\ 1, & \text{if } t > 1. \end{cases} \tag{7.2.3}$$

It can be seen that any function satisfying this condition must be of the form

$$r(t) = \sin\theta(t)\, e^{i\phi(t)}, \tag{7.2.4}$$

where $\theta(t)$ is a real function satisfying

$$\theta(t) + \theta(-t) = \pi/2, \qquad \theta(t) = \begin{cases} 0, & \text{if } t < -1 \\ \frac{\pi}{2}, & \text{if } t > 1, \end{cases} \tag{7.2.5}$$

and $\phi(t)$ is a real function satisfying

$$\phi(t) = \begin{cases} 2n\pi, & \text{if } t < -1 \\ 2m\pi, & \text{if } t > 1\,. \end{cases} \tag{7.2.6}$$

As can be seen from the definition, the rising cutoff function is not necessarily real and monotone. Thus, this type of function does not fully fit our purpose since what we need are just those functions satisfying definition 7.2.1. In this thesis, we are only interested in a special kind of rising cutoff function, called smooth unit step functions, which are real and non-decreasing. Therefore, smooth unit step function always take the form $r(t) = \sin(\theta(t))$, where $\theta(t)$ is non-decrease and satisfies equation (7.2.5).

Note that for any smooth unit step function $r(t)$, the function defined by $\theta(t) = \frac{\pi}{2}r^2(t)$ satisfies equation (7.2.5). Therefore, smooth unit step functions can be constructed iteratively. To construct the smooth unit step functions, we can first begin with a simple function $\theta_0(t)$ satisfying condition (7.2.5). Then for $k = 1, 2, \cdots$, the smooth unit step functions $r_k(t)$ can be defined iteratively in the following way:

$$\begin{array}{rcl} r_k(t) & = & \sin\theta_{k-1}(t) \\ \Uparrow & & \Downarrow \\ \theta_k(t) & = & \frac{\pi}{2}r_k^2(t) \end{array} \tag{7.2.7}$$

The interesting feature of the construction procedure is that the smoothness of the constructed smooth unit step function will be increased with the increase of the number of iterations when the initial $\theta_0(t)$ is continuous everywhere and is differentiable except at the points $t = 1$ and $t = -1$.

Here is an example of a set of smooth unit step functions constructed in this way.

Example 1. *[108] One of the simplest continuous θ-functions can be given by*

$$\theta_0(t) = \begin{cases} 0 & t < -1 \\ \frac{\pi}{4}(1+t)) & -1 \leq t \leq 1 \\ \frac{\pi}{2} & t > 1. \end{cases} \tag{7.2.8}$$

With this initial θ-function, a set of smooth unit step functions can be constructed according to (7.2.7).

$$r_1(t) \quad = \sin\theta_0(t) \quad = \begin{cases} 0 & t < -1 \\ sin(\frac{\pi}{4}(1+t)) & -1 \leq t \leq 1 \\ 1 & t > 1, \end{cases} \tag{7.2.9}$$

$$\begin{aligned} \theta_1(t) \quad &= \tfrac{\pi}{2} r_1^2(t) \quad = \begin{cases} 0 & t < -1 \\ \frac{\pi}{4}(1+\sin\frac{\pi}{2}t) & -1 \leq t \leq 1 \\ \frac{\pi}{2} & t > 1 \end{cases} \\ &= \theta_0(\sin\frac{\pi}{2}t), \\ r_2(t) \quad &= \sin\theta_1(t) \quad = \begin{cases} 0 & t < -1 \\ r_1(\sin\frac{\pi}{2}t) & -1 \leq t \leq 1 \\ 1 & t > 1, \end{cases} \end{aligned}$$

and in general, we have

$$r_{n+1}(t) = \begin{cases} 0 & t \leq -1 \\ r_n(\sin\frac{\pi}{2}t) & -1 < t < 1 \\ 1 & t \geq 1. \end{cases} \tag{7.2.10}$$

Obviously, $r_n(t)$, $(n = 1, 2, \cdots)$ constructed above are all non-decreasing. Furthermore, it can be shown that $r_n(t)$ has $2^{n-1} - 1$ vanishing derivatives at $t = 1$ and $t = -1$ for $n = 1, 2, \cdots$.

Definition 7.2.2. Let $r(t)$ be a smooth unit step function, and let a, ϵ be real numbers with $\epsilon > 0$. The real function $r(\frac{t-a}{\epsilon})$ is called a smooth unit function at point a with rising range $[a-\epsilon, a+\epsilon]$, and is denoted by $r_{a,\epsilon}(t)$.

It can be seen that smooth unit functions are always gate functions.

A special unit step function can be obtained when ϵ tends to ∞. Following is a example of this kind of function:

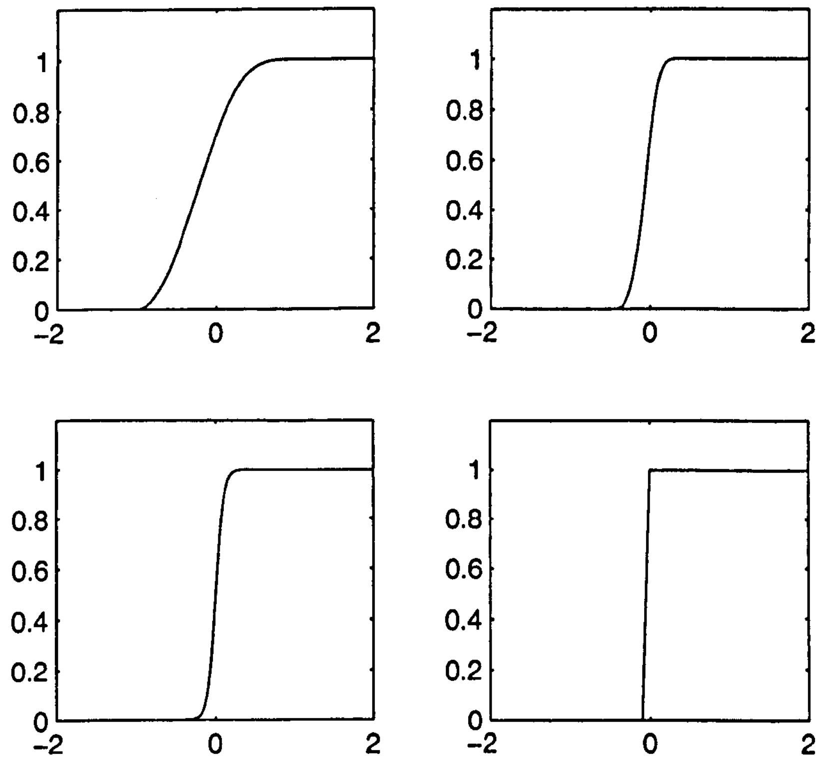

Figure 7.1: Top left: $r_1(t)$; Top right: $r_2(2.0t)$; Bottom right: $r_\infty(t)$ with $a = 10$; Bottom right: $r_\infty(t)$ with $a = 50$.

Example 2.

$$r_\infty(t) = \frac{1}{\sqrt{1+e^{-at}}}. \tag{7.2.11}$$

As can be seen directly, $r_\infty(t)$ is differentiable at any point and to any order.

Figure 7.1 shows some typical smooth unit functions.

7.2.2 Smooth one dimensional gate functions

As the only simply connected set on the real line $\mathbb{R}$ is an interval, the one dimensional gate functions have a very simple form.

Proposition 7.2.1. *For a real function $g : \mathbb{R} \to [0, 1]$, the following conditions are equivalent:*

(1) g is a gate function;

(2) For each $\alpha \in [0, 1]$, the α-level set g_α is a convex set;

(3) For each $\alpha \in [0, 1]$, the α-level set g_α is an interval;

(4) For $t_1, t_2 \in \mathbb{R}$ and any $\lambda \in [0, 1]$,

$$g(\lambda t_1 + (1 - \lambda)t_2) \geq \min(g(t_1), g(t_2))$$

Proof. The equivalence of condition (1), (2), (3) are evident as the interval is the only kind of simply connected set and convex set. The sufficiency of condition (4) is obvious since if g satisfies (4), g_α will be convex for any $\alpha \in [0, 1]$. Now we show the necessity of condition (4). For any $t_1, t_2 \in \mathbb{R}$, let $\alpha = \min(g(t_1), g(t_2))$. Then, $t_1, t_2 \in g_\alpha$. If g is a gate function, then g_α will be a simply connected set, thus the points between t_1 and t_2 must be in g_α. That is, for any λ, $\lambda t_1 + (1 - \lambda)t_2 \in g_\alpha$. In other words, $g(\lambda t_1 + (1 - \lambda)t_2) \geq \alpha = \min(g(t_1), g(t_2))$. □

Let $[a, b]$ be an interval of the real line $\mathbb{R}$. The interval can then be represented as its characteristic function $\chi_{[a,b]}(t)$, where $\chi_{[a,b]}$ is defined in such a way that

$$\chi_{[a,b]}(t) = \begin{cases} 1, & \text{if } t \in [a,b] \\ 0, & \text{otherwise}. \end{cases}$$

With the smooth unit step function, a smooth approximation to $\chi_{[a,b]}(t)$ can be obtained. Let $r(t)$ be a smooth unit step function at origin, then the interval can be approximated by $g_1(t) = r_{a,\epsilon}(t) - r_{b,\epsilon}(t)$ or by $g_2(t) = r^2_{a,\epsilon}(t) - r^2_{b,\epsilon}(t)$ if $a < b$ and ϵ is small enough such that $a + \epsilon < b - \epsilon$. As $r(t)$ and $r^2(t)$ are non-decreasing, both $g_1(t)$ and $g_2(t)$ are obviously gate functions. Figure 7.2 shows different degrees of approximation to the interval [1,4].

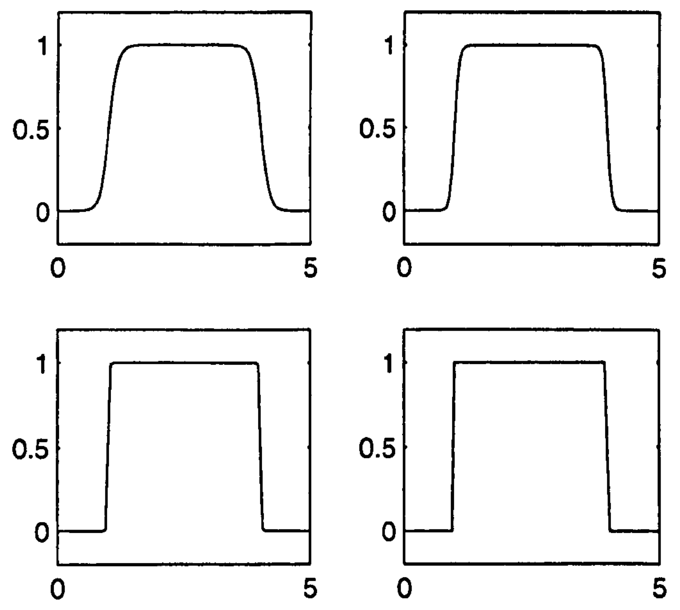

Figure 7.2: Smooth gate functions corresponding to interval $[1, 4]$ constructed from smooth unit step functions with different rising ranges

7.2.3 Gate functions in higher dimensions

The concept of the smooth gate function can be extended to higher dimensions. In the one dimensional case, the only geometrical object that can separate a real line into two simply connected parts is the point, and the only simply connected set is the interval. However, in 2D and 3D or even higher dimensions, it is much more complicated. For example, in the 2D case, there will be an infinite number of different types of geometric object that can separate a plane into two simply connected areas, and there will be a variety of different kinds of simply connected sets. In this section, we will discuss how to extend the smooth gate functions in one dimension to 2D smooth gate functions. The further extension to 3D or even higher dimensions will be similar.

Now we investigate how to construct 2D gate functions corresponding to a 2D geometric object. Let $\mathcal{O}$ be a 2D geometric object that separates the 2D plane into two parts $\mathcal{O}^-, \mathcal{O}^+$ and let $d(P, \mathcal{O})$ be the signed distance from point P to $\mathcal{O}$, i.e.,

$$d(P,\mathcal{O}) = \begin{cases} > 0 & \text{when } P \in \mathcal{O}^+, \\ = 0 & \text{when } P \in \mathcal{O} \\ < 0 & \text{otherwise.} \end{cases}$$

For any smooth unit step function $r(t)$, a function $g(P)$ can be defined in the following way

$$g(P) = r^2(d(P,\mathcal{O})).$$

If $\{P \in \mathbb{R}^2 : d(P,\mathcal{O}) \geq \delta\}$ for any $\delta \in \mathbb{R}$ is simply connected, then function $g(P)$ will be a 2D gate function having the property that the value of the function will be close to 1 for $P \in \mathcal{O}^+$ and close to 0 for $P \in \mathcal{O}^-$. From this fact, various 2D smooth gate functions can be constructed corresponding to some simple 2D geometric primitives.

We first consider how to construct 2D smooth functions corresponding to straight lines. 2D lines can be represented in several ways. In section 4.2, 2D lines have been represented by the direction of the line plus a point on the line. However, this representation is not the best one for calculating the signed distance from a point to a line. Note that in the 2D plane, the direction of a line and its normal are determined by each other. Instead of representing a line using its direction, we use the line normal to represent its orientation. The advantage of representing a 2D line in this way is that the signed distance from a point to the line can be given in extremely simple form.

Let $\mathcal{L}(P_0, \mathbf{n})$ be a straight line on the 2D plane, where P_0 represents a point on the line, and $\mathbf{n}$ the normal of the line, i.e., $\mathbf{n}$ is the unit vector perpendicular to the line (note that this representation is valid only for 2D lines). Then the signed distance from a point P to the line can be represented as

$$d(P, \mathcal{L}) = (P - P_0) \cdot \mathbf{n}. \tag{7.2.12}$$

With this signed distance, a smooth gate function in 2D generated by the straight line can be given in the form:

$$\mathcal{R}(\mathcal{L}) = r^2(d(P, \mathcal{L})), \tag{7.2.13}$$

where $r(t)$ is a smooth unit step function defined in the previous section. The shape of this function is shown in Figure 7.3.

As with one dimensional gate functions, simple gate functions can be combined to generate more complex gate functions. For example, polygonal regions can be approximated with smooth functions generated from 2D line gate functions. Figure

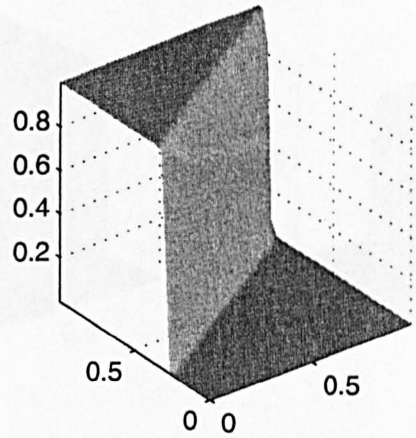

Figure 7.3: 2D gate function generated by a straight line

7.4 shows the shapes of some gates of this kind. The top-left figure corresponds to

$$f_1(P) = r_\infty^2(d(P, \mathcal{L}(P_0, \mathbf{e}_1))r_\infty^2(d(P, \mathcal{L}(P_0, \mathbf{e}_2)),$$

where $P_0 = (0.3, 0.3)$ and $\mathbf{e}_1 = (1, 0)$, $\mathbf{e}_2 = (0, 1)$. The top-right and bottom left figures are plotted for functions

$$f_2(P) = \mathcal{R}(\mathcal{L}_{21})\mathcal{R}(\mathcal{L}_{22})\mathcal{R}(\mathcal{L}_{23})$$

and

$$f_3(P) = \mathcal{R}(\mathcal{L}_{31})\mathcal{R}(\mathcal{L}_{32})\mathcal{R}(\mathcal{L}_{33})\mathcal{R}(\mathcal{L}_{34}),$$

where $\mathcal{L}_{ij}(i = 2, 3; j = 1, 2, 3, 4)$ are lines corresponding to the edges of the relevant polygons with inward pointing normals. The one at the bottom right is defined by $f_4(P) = f_3(P)(1 - f_1(P))$.

Smooth gate functions in two dimensions can also be generated by 2D curves. Generally, let $f(P) = 0$ be the implicit representation of a curve C such that $\{P \in \mathbb{R}^2 : f(P) \geq \delta\}$ is a simply connected set for any real number δ. Then for any smooth

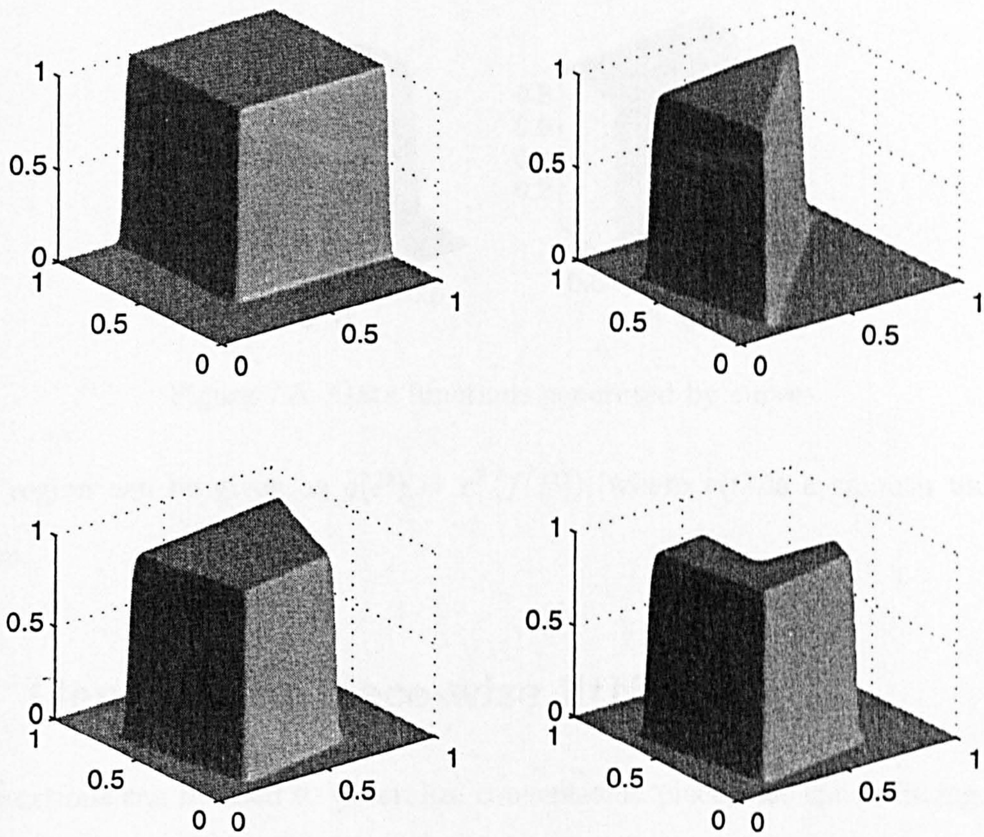

Figure 7.4: 2D gate functions generated by several straight lines

unit step function $r(t)$, the composite function $G(P) = r^2(f(P))$ leads to a curved 2D gate function, as illustrated in Figure 7.5.

In the above discussion, if we replace the 2D geometric object with a 3D geometric object, 3D smooth gate functions can be generated. Therefore, the generalization from 2D smooth gate functions to 3D smooth gate functions is direct, and need not be discussed in detail in this thesis.

In general, for a simply connected region $D \in \mathbb{R}^n$, if its boundary ∂D can be represented implicitly as $f(P) = 0$ such that $\{P \in \mathbb{R}^n : f(P) \geq \delta\}$ is a simply connected set for any real number δ, then the smooth gate function corresponding

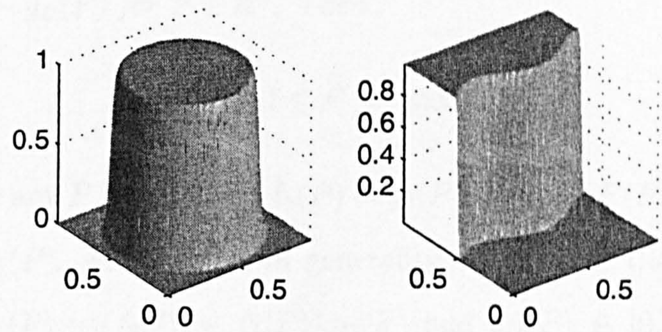

Figure 7.5: Gate functions generated by curves

to the region can be given as $g(P) = r^2(f(P))$, where $r(t)$ is a smooth unit step function.

7.3 Generalized piece-wise fitting

Gate functions can be used to generalize conventional piece-wise spline fitting. For a given data set, we could first group the data into several sub-groups, then for each subset of data, a simple curve or surface can be fitted. These piece-wise fitted curves or surfaces can then be combined to give an overall fitting for the whole data set. Compared with conventional piece-wise spline fitting, the fitted curves and surfaces can have a higher order of continuity. In addition, the fitting procedure is more direct. Our method for combining two functions is based on the following observations:

Proposition 7.3.1. *Let $f_1(P)$ and $f_2(P)$ be two smooth functions in space $\mathbb{R}^n$. Let $g_D(P)$ be a gate function corresponding to a region $D \subset \mathbb{R}^n$ and let*

$$F(P) = f_1(P)g_D(P) + f_2(P)g_D^c(P),$$

where $g^c(P) = 1 - g_D(P)$ *for* $P \in \mathbb{R}^n$. *Then*

$$min(f_1, f_2) \leq F \leq max(f_1, f_2).$$

Proof. In fact, for any $P \in \mathbb{R}^n$, when $f_1(P) = f_2(P)$, we have $F(P) = f_1(P) = f_2(P)$; when $f_1(P) \neq f_2(P)$, without loss of generality, we assume that $f_2(P) > f_1(P)$. Since $F(P) = f_2(P) - (f_2(P) - f_1(P))g_D(P)$ and $g_D(P) \in [0, 1]$, it follows that $f_1(P) \leq F(P) \leq f_2(P)$. □

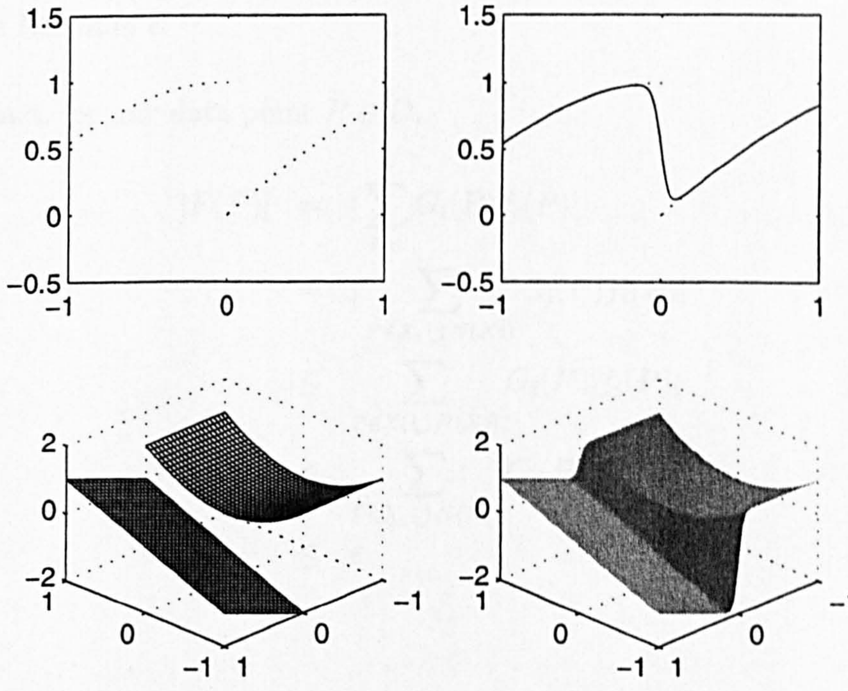

Figure 7.6: Two curves and surfaces are smoothly connected

Proposition 7.3.2. *Suppose* $f(P)$ *is a real function defined on the space* $\mathbb{R}^n$. *Let* D *be a data set sampled from the surface defined by the equation* $f(P) = 0$ *with added noise. Let* $X_1, X_2, \cdots, X_n$ *be a partitioning of space* $\mathbb{R}^n$, *such that on each subset* X_i *plus its neighborhood* $N(X_i)$, *an equation* $f_i(P) = 0$ *can be fitted with fitting* *error at each data point* $P \in (X_i \bigcup N(Xi)) \bigcap D$ *less than a given number* $\epsilon > 0$, *i.e.*, $|f_i(P)| \leq \epsilon$. *Let* $G_1, G_2, \cdots, G_n$ *be the gate functions corresponding to subsets* $X_1 X_2, \cdots, X_n$, *such that* $G_i = 0$ *outside* $X_i \bigcup N(Xi)$ *and*

$$G_1(P) + G_2(P) + \cdots + G_n(P) = 1.$$

Then, the fitting error at each data point for function

$$F(P) = \sum_{i=1}^{n} G_i(P) f_i(P)$$

will also be less than ϵ.

Proof. In fact, for any data point $P \in D$,

$$\begin{aligned} |F(P)| &= |\sum_{i=1}^{n} G_i(P) f_i(P)| \\ &= |\sum_{P \in X_i \bigcup N(Xi)} G_i(P) f_i(P)| \\ &\leq \sum_{P \in X_i \bigcup N(Xi)} G_i(P) |f_i(P)| \\ &\leq \sum_{P \in X_i \bigcup N(Xi)} G_i(P) \epsilon \\ &\leq \epsilon \end{aligned}$$

□

With property 7.3.1, different shapes can be combined smoothly. Fig 7.6 shows that curves and surfaces can be combined smoothly with gate functions. The most important part of this combination is that the original shapes are well preserved in their combined shape.

We notice that for any smooth unit step function $r(t)$, the monotonic properties of $r^2(t)$ and $r(t)$ are similar. Therefore, a smooth gate function can be constructed from either $r(t)$ or $r^2(t)$. However, $r^2(t)$ has property that $r^2(t) + r^2(-t) = 1$, which $r(t)$ does not have. If we connect two functions f_1 and f_2 at a point $t = a$ with $r^2_{a,\epsilon}(t)$ and $1 - r^2_{a,\epsilon}(t) = r^2_{-a,\epsilon}(-t)$ such that

$$f(t) = r^2_{a,\epsilon}(t)f_1(t) + r^2_{-a,\epsilon}(-t)f_2(t),$$

then $f(a) = \frac{f_1(a)+f_2(a)}{2}$. Thus, this combination is unbiased. However, if we combine f_1 and f_2 with $r_{a,\epsilon}(t)$ and $1 - r_{a,\epsilon}(t)$, it will be a biased connection as $r(0)$ might not be $\frac{1}{2}$ in general. In the following discussion, we always assume that the smooth gate functions are constructed from r^2.

7.3.1 Constructive explicit curve fitting

Suppose that a set of points $D = \{P_i(x_i, y_i)\}_{i=1}^n$ is generated from an unknown function like $y = f(x)$. When the form of the true function $f(x)$ is linear or quadratic, there is no difficulty in finding the optimal estimate for such a function. However, when the true function $f(x)$ is very complicated, it is not easy to estimate it directly. The solution is piecewise fitting, such as piecewise linear fitting and piecewise cubic spline fitting. However, to fit an extremely smooth curve to the given data seems difficult, for example fitting data with a curve of C^k-continuity for $k \geq 4$. In this section, we suggest a very flexible piecewise fitting technique using gate functions, which can be regarded as the generalization of conventional piecewise spline fitting. Using this technique, the given data can be fitted with a curve with any degree of smoothness. Suppose for all $\{x_i\}_{i=1}^m$ we have $a \leq x_i \leq b$. Randomly choose a set of points $a = a_0 < a_1 \cdots < a_n = b$ from the interval $[a, b]$. For each sub-interval $[a_i, a_{i+1}]$, let $g_i(x)$ be the corresponding gate function, and let $y = f_i(x)$ be the locally fitted functions based on data $\{P_i(x_i, y_i) : x_i \in [a_i, a_{i+1}]\}$, called the confined data

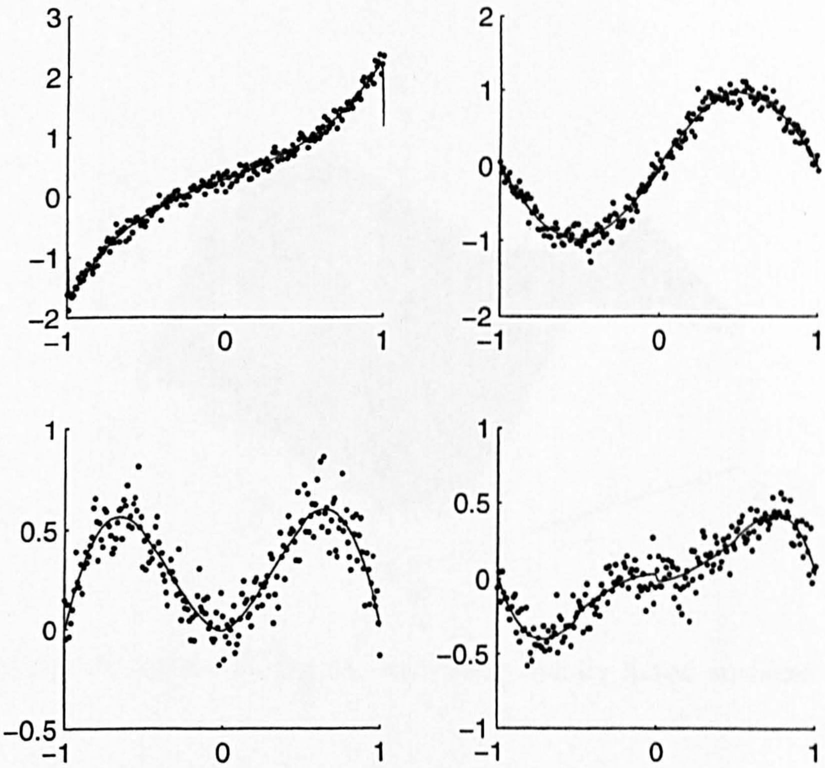

Figure 7.7: Explicit curve fitting by combining locally fitted curves with one dimensional smooth gate functions

of D on interval $[a_i, a_{i+1}]$, using any conventional fitting technique. Then, the overall fitting can be represented by

$$f(x) = \sum_{i=0}^{n-1} g_i(x) f_i(x). \tag{7.3.1}$$

The degree of smoothness of function $f(x)$ depends on the choice of the gate function $g_i(x)$ and the degree of smoothness of each locally fitted function. Generally, we could use $r_\infty^2(x)$ introduced previously, to construct $g_i(x)$, which is an infinitely differentiable function. In this case the degree of smoothness of the combined function $f(x)$ defined in (7.3.1) depends only on the smoothness for each locally fitted function $f_i(x)$.

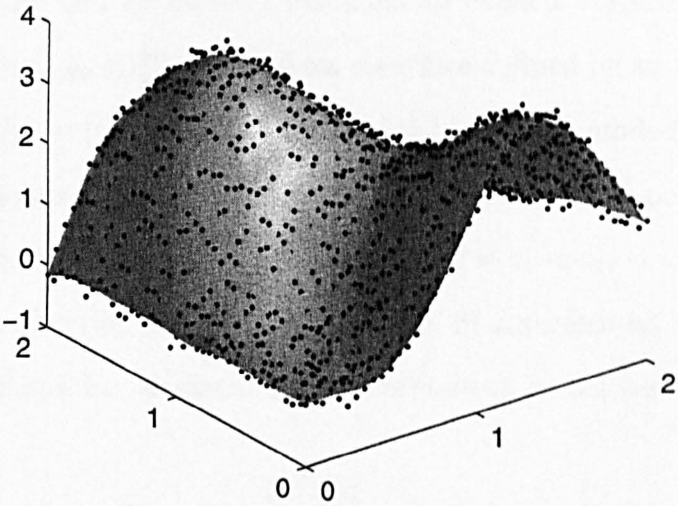

Figure 7.8: Explicit surface fitting by combining locally fitted surfaces with 2D gate functions

The only problem for this kind fitting is that the combined function defined in (7.3.1) may have too many fluctuations when data are too noisy. To avoid such a problem, we could slightly expand the data used in each sub-interval by including the data in their $\epsilon-$neighbor, i.e., use the data in interval $[a_i - \epsilon, a_{i+1} + \epsilon]$ to estimate the i^{th} function $f_i(x)$. Figure 7.7 shows some fitting results based on this strategy where each local data set is fitted with a polynomial of degree less than three. As can be seen from the figure, the fitting is quite satisfactory even when the data are extremely noisy. The two obvious advantages of this fitting technique are that the fitted curve can be extremely smooth on one hand and the fitting procedure is very simple on the other.

7.3.2 Constructive explicit surface fitting

The same procedure can be directly extended to explicit surface fitting. Suppose that data $D = \{P_i(x_i, y_i, z_i)\}_{i=1}^n$ come from a surface defined by an unknown function $z = f(x, y)$. Let $D_{xy} = \{(x_i, y_i) : (x_i, y_i, z_i) \in D\}$ be in the bounded area $[a, b] \times [c, d]$. Choose a set of points $\{u_i\}_{i=0}^m$ from interval $[a, b]$, and a set of points $\{v_j\}_{j=0}^k$ from interval $[c, d]$, such that $a = u_0 < u_1 \cdots < u_m = b$, $c = v_0 < v_1 \cdots < v_k = d$. For each subset of the data $D \cap ([u_i, u_{i+1}] \times [v_j, v_{j+1}] \times \mathbb{R})$, fit a surface $z_{ij} = f_{ij}(x, y)$ locally. Then the overall shape for the data can be represented by the sum

$$F(x, y) = \sum_{i=0}^{m-1} \sum_{j=0}^{k-1} g_{ij}(x) f_{ij}(x, y) \tag{7.3.2}$$

where $g_{ij}(x, y)$ is the gate function corresponding to rectangle $[u_i, u_{i+1}] \times [v_j, v_{j+1}]$. Figure 7.8 shows that a cloud of data (containing a great deal of noise) can be well approximated by a smooth function which is a combination of 64 quadratic patches.

As with explicit curve fitting, the degree of smoothness of the surface defined by (7.3.2) depends on the smoothness of both f_{ij} and g_{ij}. When r_∞^2 is used to define the gate functions g_{ij}, the smoothness of $F(x, y)$ will only depend on those $f_{ij}(x, y)$. In most cases, a lower degree polynomial will be fitted for each local data set and thus the fitted surface (7.3.2) has derivatives of all orders.

7.4 Piece-wise implicit fitting

In this section, we will describe how smooth gate functions can be used to establish a piece-wise implicit fitting. It has been pointed out that implicit fitting is a very difficult problem in that the shape fitted can be either over-fitted or under-fitted. A feasible method is to subdivide the complicated data set into smaller data sets such that each subset of the data can be well approximated with a simple implicitly represented geometric shape like a line, a quadric or a cubic surface. However, geometric shapes combined with conventional blending techniques do not usually preserve the locally fitted shapes. In addition, the combined shape might not have a high degree of smoothness. In this section, we propose a constructive implicit function fitting strategy using gate functions. The combined surface will not only be extremely smooth in general but can also be expressed in a neat form. The most important feature of the fitting is that the local properties of each locally fitted implicit shape are well preserved.

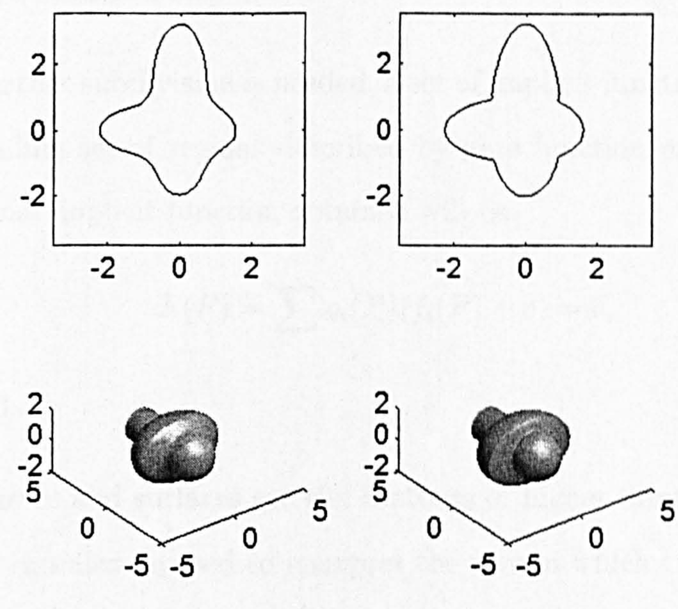

Figure 7.9: Implicit curves and implicit surfaces are smoothly combined using gate functions.

Suppose a data set $D = \{P_i\}_{i=1}^{m}$ is a set of points from a surface in space $\mathbb{R}^n$. Let A be a bounded area such that $D \subset A \subset \mathbb{R}^n$. We propose the following algorithm for piece-wise fitting:

Algorithm 7.4.1.

1. Divide the area A into two areas A_1, A_2 with a geometric primitive, such as a line (in 2D) or a plane (in 3D). Let $D_1 = A_1 \cap D$ and $D_2 = A_2 \cap D$. Fitting D_1 and D_2 with some simple implicit functions respectively.

2. Check the accuracy for each fitting. If the fitting in the existing area is acceptable, stop subdividing the corresponding region and keep the fitted implicit function. If the fitting in the existing region is not acceptble, further subdivide the region into two subregions. Set the current region to be A and current data to be D, and return to step 1.

3. When no further subdivision is needed, a set of implicit functions $f_i(P) = 0$ and a corresponding set of regions described by gate function $g_i(P)$ are obtained. Then the final implicit function obtained will be:

$$F(P) = \sum g_i(P)(f_i(P) + \delta) = \delta,$$

where $\delta > 0$.

As implicit curves and surfaces are the contours of higher dimensional functions, Proposition 7.3.1 can also be used to interpret the way in which two implicit shapes are combined. Figure 7.9 shows how smoothly two shapes can be combined with gate functions.

7.4.1 Piece-wise implicit curve fitting

At the beginning of this section we proposed a general constructive fitting procedure by partitioning the data space iteratively. However, the partition can be quite arbitrary. In practice, we could chose a specific method of partitioning in a way which exploits any special features of the data set.

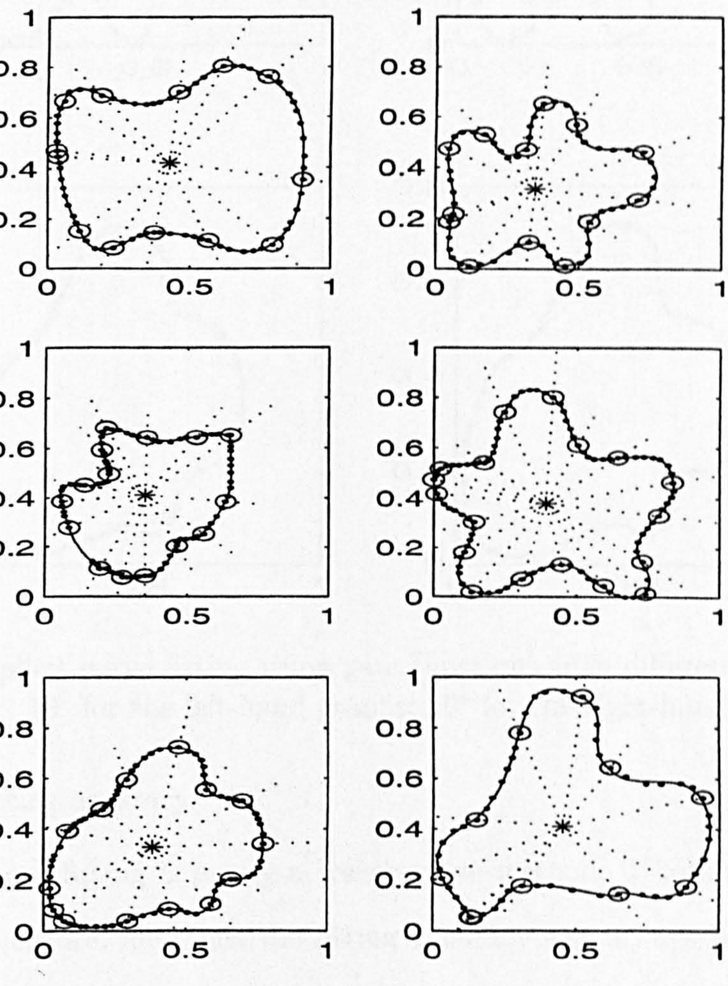

Figure 7.10: Implicit curve fitting using gate functions

Figure 7.10 is obtained by dividing the data area into different fan-shaped areas with local approximation accuracy 10^{-4}. Figure 7.11 compares the fitting results with

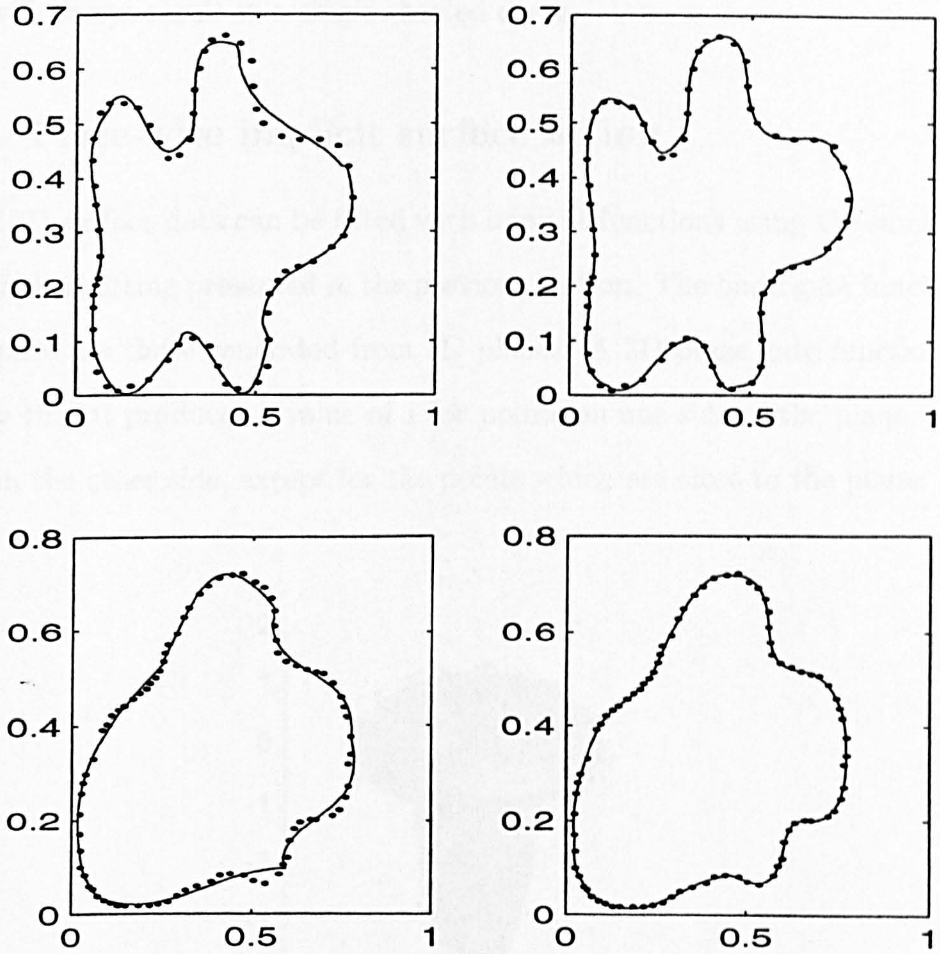

Figure 7.11: Implicit curve fitting using gate functions with different local approximation accuracy: 10^2 for the left-hand graphs; 10^4 for the right-hand graphs.

different local fitting accuracy.

As the proposed fitting is purely a constructive method, it does not rely on any optimization procedure, and thus the fitting accuracy can always be guaranteed if the partition is fine enough. With this method, very complex shapes can be fitted with implicit functions. To avoid multiple sheeted fitting, a single sheeted curve is preferable for each subset of the data. For example, we could fit a set of data with an implicit function which is obtained by rotating an explicit curve like $y - f(x) = 0$, which will always result in a single-sheeted curve.

7.4.2 Piece-wise implicit surface fitting

A set of 3D surface data can be fitted with implicit functions using the same strategy as 2D implicit fitting presented in the previous section. The basic gate functions used in 3D space are those generated from 3D planes. A 3D plane gate function has the property that it produces a value of 1 for points on one side of the plane, and 0 for points on the other side, except for the points which are close to the plane.

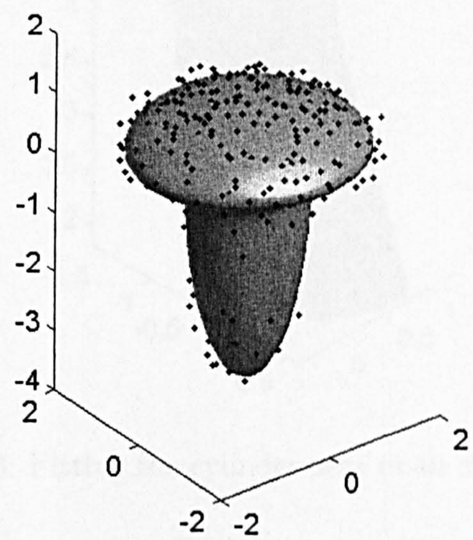

Figure 7.12: Example of constructive implicit surface fitting

As an example, Figure 7.12 shows how separately fitted implicit surfaces are combined to form a smooth surface. In this example, the data given are first grouped into two subsets corresponding to the upper and the lower parts of the object separated by a plane $\Gamma : (P - P_0) \cdot \mathbf{n} = 0$, where P_0 is a point on the plane and $\mathbf{n}$ is the normal to the plane. Let the gate functions $g_1(P) = r^2((P - P_0) \cdot \mathbf{n}))$, $g_2(P) = 1 - g_1(P)$, where $r(t)$ is a smooth unit step function at the origin. For each subset of the data, an implicit surface is fitted. Let $f_1(P) = 0$ and $f_2(P) = 0$ be the fitted implicit surfaces corresponding to the data subset on the positive side and the negative side of the plane respectively. Then the overall implicit fitting for the whole data set can be given by

$$g_1(P)(f_1(P) + 0.5) + g_2(P)(f_2(P) + 0.5) = 0.5.$$

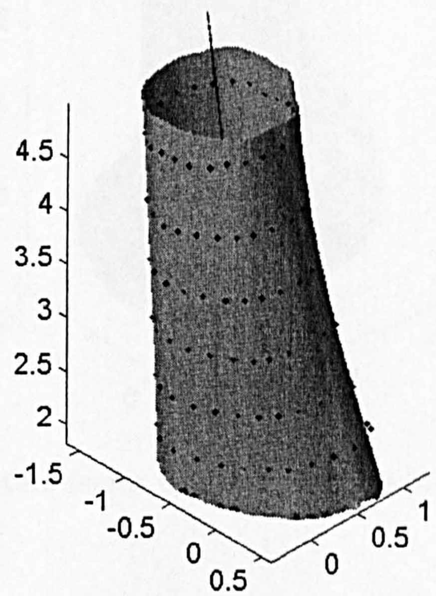

Figure 7.13: Fitting the cylinder part of an actual femur

Figures 7.13 to 7.16 are fitting results for data from an actual bone surface. The femur data have been first partitioned into two parts with a plane, corresponding to the top and bottom of the femur. The two subsets of the data are then fitted separately with our constructive implicit fitting technique. Figure 7.13 is the fitting result for the Femur shaft (the cylinder part of Femur). Figure 7.14 combines the fitted surface for the shaft with a second fitting for the femoral condyles. Each Local data set is fitted with a low degree implicit polynomial. In total, 80 gates are

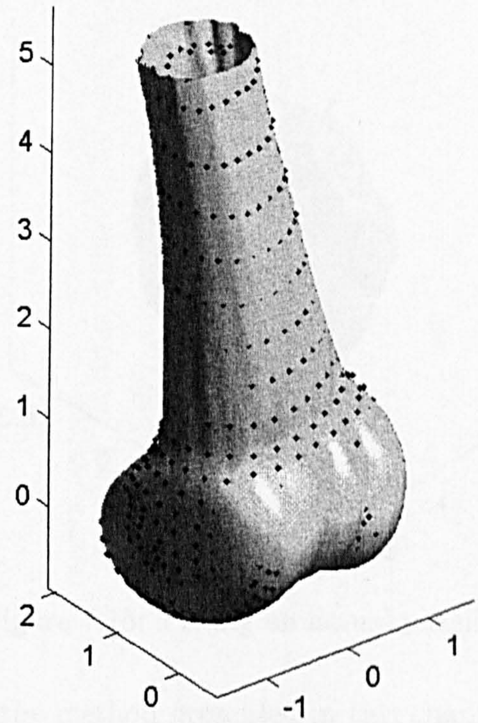

Figure 7.14: Fitting an actual femur by combining the locally fitted implicit shapes together

used. The patella data and tibia data are fitted similarly but with slightly different partitioning procedures.

7.4.3 Summary

To summarize, in this chapter a new technique for fitting complex shapes with implicit functions using gate functions has been developed. Some interesting applications on explicit and implicit curve and surface fitting using smooth unit step functions and general gate functions are also investigated.

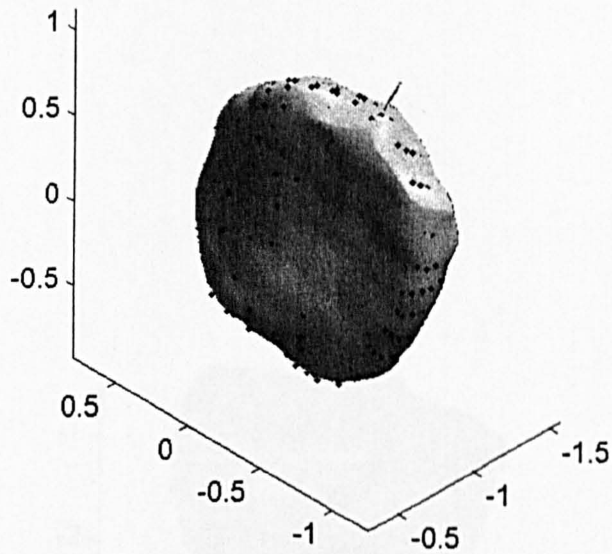

Figure 7.15: Fitting an actual patella

In the next chapter, the method presented in this chapter will be used to reconstruct implicit surfaces for the data from the surface of an actual bone. The estimated implicit function will then be used for developing non-landmark registration techniques.

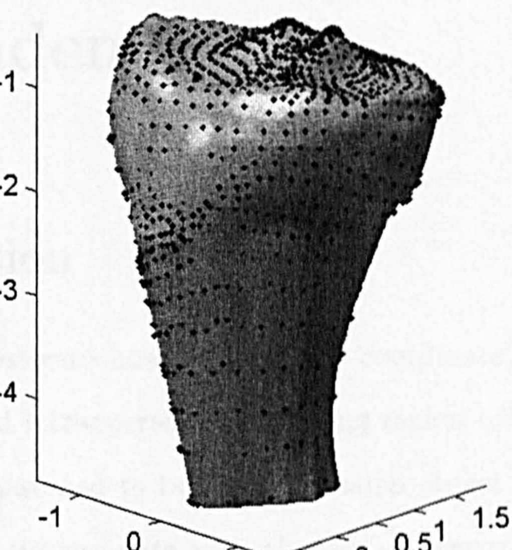

Figure 7.16: Fitting an actual tibia

Chapter 8

Coordinate System Alignment Using Region to Region Correspondence

8.1 Introduction

In this chapter, we investigate how to align two coordinate systems corresponding to preoperative data and intraoperative data using region to region correspondence. The two data sets are assumed to be from the same object and a region to region correspondence between the two data sets is known. However, we do not have precise information about point to point correspondences. In this registration procedure, it is assumed that several regions of interest are selected from the surface of a bone. These regions should be easy to identify by surgeons during surgery. For each of these regions, two sets of data points are sampled. One data set serves as the preoperative model data and the other serves as the intraoperative data. Each model data set is assumed to be dense so that the shape of its corresponding region can be precisely represented, while the intraoperative data set is assumed to be sparse.

The region to region matching strategy is basically an optimization procedure where a good initial estimate for the unknown rigid transformation can be obtained through the region to region correspondence. In searching for the best rigid transformation, each model data set is first modelled as an implicit function which serves as the distance function from a point to the region. Then the search becomes basically a function minimization problem.

Compared with general non-landmark registration techniques, the region to region match method has two obvious advantages. In image guided surgery, the matching between the preoperative data and the intraoperative data must be accurate. To avoid an optimization procedure stopping at a local minimum, a good initial guess is always necessary for any local minimization procedure to guarantee an accurate estimate. In addition, the model for each region can be much simpler than the model for the whole bone, and thus the search for the best rigid transformation by minimizing a sum of squared distance functions will be much quicker.

Formally, the region to region registration strategy can be stated as follows:

Let $\{A_i\}_{i=1}^{n}$ be n regions from the surface of a bone object. These regions should be well separated in order to provide a good initial guess for the unknown rigid transformation. Let C_l, C_r be the two coordinate systems in which regions $\{A_i\}_{i=1}^{n}$ are presented. Suppose the shape of each region A_i can be represented as an implicit function $f_i(P) = 0$ in C_r. Let $\{P_{ij}\}_{j=1}^{m_i}$ be a set of points from region $A_i (i = 1, 2, \cdots, n)$ in coordinate system C_l. We wish to estimate the rotation R and translation T that links the two coordinate systems by minimizing the sum

$$\sum_{i=1}^{n} \sum_{j=1}^{m_i} f_i^2(RP_{ij} + T). \qquad (8.1.1)$$

In sum (8.1.1), the function value $f_i(P)$ can also be replaced with a better approximation to the distance from a point to the surface as

$$\frac{f_i(P)}{\nabla f_i(P)}.$$

The organization of this chapter is as follows. We will first discuss how to compute the initial rotation parameters according to different ways of representing a rotation. This is followed by a discussion on how to compute the initial rigid transformation from the region to region correspondence. This is then followed by a discussion on when a rigid transformation can be uniquely determined by the locations and orientations of the regions considered. Finally, experimental results are given to demonstrate the efficiency and the stability of the algorithm.

8.2 The computation of the rotation parameters

Once an initial rotation matrix is obtained, the initial rotation parameters can then be calculated in one of the following ways depending on how the rotation matrix is parameterized.

8.2.1 Computing the Euler Angles

As have been seen in section 2.2.1, there are 12 different ways to represent a rotation matrix with Euler angles. Here we have chosen the form

$$\begin{aligned} R(\alpha,\beta,\gamma) &= R_z(\gamma)R_x(\beta)R_z(\alpha) = \\ & \begin{pmatrix} C_1C_3 - S_1C_2S_3 & S_1C_3 + C_1C_2S_3 & S_2S_3 \\ -C_1S_3 - S_1C_2C_3 & -S_1S_3 + C_1C_2C_3 & S_2C_3 \\ S_1S_2 & -C_1S_2 & C_2 \end{pmatrix}, \end{aligned} \tag{8.2.1}$$

where

$$C_1 = \cos\alpha, \qquad C_2 = \cos\beta, \qquad C_3 = \cos\gamma,$$

$$S_1 = \sin\alpha, \qquad S_2 = \sin\beta, \qquad S_3 = \sin\gamma,$$

and $\alpha, \gamma \in [0, 2\pi), \beta \in [0, \pi]$.

When

$$R(\alpha, \beta, \gamma) = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}, \tag{8.2.2}$$

from $\cos\beta = r_{33}$ and $\beta \in [0, \pi]$, we obtain $\beta = \arccos(r_{33})$. Note that for $\beta \in (0, \pi)$, $sin\beta > 0$, from

$$\sin\alpha \sin\beta = r_{31}, \qquad \cos\alpha \sin\beta = -r_{32}$$

we have

$$\alpha = \begin{cases} \arccos(-\frac{r_{32}}{\sin\beta}), & r_{31} \geq 0 \\ 2\pi - \arccos(-\frac{r_{32}}{\sin\beta}), & r_{31} < 0 \end{cases} \tag{8.2.3}$$

similarly,

$$\gamma = \begin{cases} \arccos(\frac{r_{23}}{\sin\beta}), & r_{13} \geq 0 \\ 2\pi - \arccos(\frac{r_{23}}{\sin\beta}), & r_{13} < 0 \end{cases} \tag{8.2.4}$$

If $\sin\beta = 0$, R will have the form:

$$\begin{pmatrix} \cos(\alpha \pm \gamma) & \sin(\alpha \pm \gamma) & 0 \\ \mp\sin(\alpha \pm \gamma) & \pm\cos(\alpha \pm \gamma) & 0 \\ 0 & 0 & \pm 1 \end{pmatrix}.$$

The three Euler angles can be chosen in the following way:

$$\alpha = \begin{cases} \arccos(r_{11}), & r_{12} \geq 0 \\ 2\pi - \arccos(r_{11}), & r_{12} < 0 \end{cases} \tag{8.2.5}$$
$$\beta = \arccos(r_{33});$$
$$\gamma = 0;$$

8.2.2 Computing the rotation axis and the rotation angle

Assume that the rotation matrix is represented as a vector $\mathbf{r} = (x, y, z)^T$. Let $\mathbf{n} = \mathbf{r}/||\mathbf{r}|| = (n_x, n_y, n_z)^T$, $\theta = ||\mathbf{r}||$, then $\mathbf{n}$ and θ will be the rotation axis and the rotation angle, and rotation matrix R can be represented in the form

$$R = \cos\theta I + (1 - \cos\theta)\mathbf{n}\mathbf{n}^T + \sin\theta N$$

where

$$N = \begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}. \tag{8.2.6}$$

Let

$$S_M = \cos\theta I + (1 - \cos\theta)\mathbf{n}\mathbf{n}^T \qquad S_K = \sin\theta N$$

then S_M is a symmetric matrix, S_K is a skew-symmetric matrix and $R = S_M + S_K$. Therefore,

$$R - R^T = 2S_K \tag{8.2.7}$$
$$R + R^T = 2S_M. \tag{8.2.8}$$

From equation (8.2.7), we have

$$\begin{aligned} n_x \sin\theta &= \frac{(r_{32} - r_{23})}{2} \\ n_y \sin\theta &= \frac{(r_{13} - r_{31})}{2} \\ n_z \sin\theta &= \frac{(r_{21} - r_{12})}{2} \end{aligned} \tag{8.2.9}$$

From the equation (8.2.8) we have $Tr(R) = Tr(S_M)$ which implies

$$\cos\theta = \frac{Tr(R) - 1}{2} \tag{8.2.10}$$

From (8.2.10), the rotation angle can always be computed. If $|\cos\theta| \neq 1$, $\sin\theta \neq 0$ and the rotation axis can be found directly from (8.2.9). If $|\cos\theta| = 1$, $\sin\theta = 0$. The rotation axis can not be determined from (8.2.9). In this case, the rotation matrix will be either $R = I$ or $R = 2\mathbf{n}\mathbf{n}^T - I$, where $\mathbf{n}$ is the rotation axis, and the rotation axis can be found directly from these relations. Note that (8.2.10) leads to two signs of $\sin\theta$. When we chose $\theta \in [0, \pi]$, $\sin\theta \geq 0$, and when we chose $\theta \in [\pi, 2\pi]$, $\sin\theta \leq 0$. When they are used, rotation by θ around $\mathbf{n}$ and rotation by $-\theta$ around $-\mathbf{n}$ are obtained respectively, they are the same rotation. When $\theta = 0$, $\mathbf{n}$ is undetermined, which is understandable, as in this case the rotation axis can be arbitrary.

8.2.3 Computing the unit quaternion

A given rotation matrix can also be represented by a quaternion, which can be represented as a vector in real four dimensional space with unit length. Let $\mathbf{q} = (s, l, m, n)^T$ be a quaternion. The corresponding rotation matrix is

$$\mathcal{Q} = \begin{pmatrix} s^2 + l^2 - m^2 - n^2 & 2(lm - sn) & 2(ln + sm) \\ 2(lm + sn) & s^2 - l^2 + m^2 - n^2 & 2(mn - sl) \\ 2(ln - sm) & 2(mn + sl) & s^2 - l^2 - m^2 + n^2 \end{pmatrix} \tag{8.2.11}$$

Now we consider how to extract the four components corresponding to the given rotation matrix R. First, from $Tr(Q) = Tr(R)$, we have

$$4s^2 - 1 = Tr(R) \tag{8.2.12}$$

On the other hand, from

$$Q - Q^T = R - R^T \tag{8.2.13}$$

we have

$$4sl = r_{32} - r_{23} \quad 4sm = r_{13} - r_{31} \quad 4sl = r_{21} - r_{12}.$$

If $s \neq 0$, the other three components can then be obtained in terms of s:

$$l = \frac{r_{32} - r_{23}}{4s} \qquad m = \frac{r_{13} - r_{31}}{4s} \qquad n = \frac{r_{21} - r_{12}}{4s} \tag{8.2.15}$$

From (8.2.12), s can be obtained either as $\frac{\sqrt{Tr(R)+1}}{2}$ or as $-\frac{\sqrt{Tr(R)+1}}{2}$, whichever of them is used, the rotation matrix R defined by the quaternion obtained will be same.

If $s = 0$, $Q = 2\mathbf{n}\mathbf{n}^T - I$, where $\mathbf{n} = (l, m, n)^T$ is a unit vector serving as the rotation axis. In this case, l, m, n can be directly computed from the relation $2\mathbf{n}\mathbf{n}^T - I = R$.

8.3 Computing the initial rigid transformation from the region to region correspondence

It has been pointed out that to obtain an accurate estimate by a local optimization procedure, a good initial guess is crucial. The region to region correspondence will undoubtedly provide very important information about the relevant relations between the two coordinate systems corresponding to the preoperative data and the intraoperative data. Though a point to point correspondence is not available, the geometric information from each region, like the region centre, can be used to estimate the initial rigid transformation. As can be seen, smaller regions provide information which is more precise.

In this section, we discuss how to compute the initial rigid transformation from the region to region correspondence.

8.3.1 Using the 'centre' of each region

To use the information from a region to region correspondence, the data from each region needs to be summarized to provide a good initial guess for the estimate of the rigid transformation that links the two groups of data sets. Depending on the geometric features of the surface of the object, the data in each region can be summarized in several ways.

Centre of gravity

When the size of each region is small and all the regions selected are well-separated, the centre of gravity of the data set for each region can be used to approximate the overall position of the region, and the initial guess for the rigid transformation for the region to region matching problem can be obtained immediately through point to point correspondences using the algorithms given in Chapter 3.

'Curved' data centre

The centre of gravity describes the centre of the data set well when the data come from a linear region like a straight line or a plane. However, it cannot describe well the centre position for data from a curved region. For example, when the data come from a sphere, the centre of gravity of the data will never be on the sphere when there are more than one point in the data. For data from a curved surface, we propose the concept of a 'curved' centre.

There are two obvious ways to define the concept of a 'curved' centre, with slightly different meanings. One is to define the curved centre as the point on the region which has minimum sum of squared distances from all other points in the data set. The second way is to define the curved centre as the point on the region such that it is closest to the centre of gravity of the data. In some cases, the two ways just define same point. For example, when the data is from a straight line or a plane. When surface S is known or can be easily estimated, we can follow the first idea to compute the 'curved' centre of the data set. But the second idea is much easier to implement in practice. Suppose $D = \{P_i\}_{i=1}^{n}$ is a set of points from a curved region, and its centre is $\bar{P}$. Then the curved centre can just be given approximately with the point $\widehat{P} \in D$ that is closest to $\bar{P}$.

Once the curved centre for data set from each region has been computed, the initial guess for rigid transformation can be obtained by reference mark registration techniques.

8.3.2 Describing each region as a simple geometric object

In computing a good initial guess for the unknown rigid transformation from the region to region correspondence, the geometry of regions provide more reliable information than the statistical properties of their corresponding data points. Though the centroid and 'curved' centroid can be used to describe the position of each region approximately when the data points are evenly distributed on the surface of the region, they might provide faulty information when data are not evenly distributed. Furthermore, for each region, the intraoperative data might be too sparse to provide any meaningful centre for its corresponding region. In all these cases, it would be better to model each set of preoperative data with a simple geometric shape like a plane or an ellipsoid. When there are enough points in the intraoperative data for each region, it can be fitted with a similar shape and the initial guess can be obtained by matching these two simple geometric shapes. However, the intraoperative data will be very sparse usually. In this case, the initial rigid transformation can be found by point to surface matching with a simple optimization procedure.

Computing the initial rigid transformation by modeling each region as a straight line or a plane

In this subsection, we discuss how to find the initial match when each region can be described by a straight line or a plane. We first consider line regions.

Some bones like tibia and fibula have sharp edges called interosseous margins on their surfaces. This anatomic feature can be approximately represented by a straight line. A line region is one of the most simple geometric features that can be identified on the surface of a bone. Let

$$\mathcal{L}_1(P_1, \mathrm{v}_1),\ \mathcal{L}_2(P_2, \mathrm{v}_2),\ \cdots,\ \mathcal{L}_n(P_n, \mathrm{v}_n).$$

be a set of lines on the surface of a rigid object O. Let

$$\bigcup_{i=1}^{n} \{Q_{ij} : j = 1, 2, \cdots, m_i\}.$$

be a set of points sampled from the rigidly transformed position of object O with $\{Q_{ij}\}_{j=1}^{m_i}$ lying on the i^{th} line region. From a practical point of view, when these lines can determine the position and orientation of the object, the best match can be found from the set of lines and their corresponding data sets.

Basically, there are two ways to estimate the initial rigid transformation that roughly transforms these groups of data sets onto their corresponding lines. The first method is to turn the undirected line match into a directed line match. If, for each region, there are at least two points in the intraoperative data set, a straight line can be fitted. Thus, matching a group of data points to a group of lines becomes matching two groups of undirected straight lines. If we fix the directions of lines corresponding to the preoperative data, then a rigid transformation can be computed using the method developed in section 4.2 for each set of selected line directions corresponding to the intraoperative data. The best initial rigid transformation estimated should be the one corresponding to the line directions that produce the smallest matching errors.

The second method is to use a minimization procedure. Let R and T be the rotation and translation to be estimated. The distance from $RQ_{ij}+T$ to line $\mathcal{L}_i(P_i, \mathbf{v}_i)$ can then be represented as

$$(RQ_{ij} + T - P_i)^T(I - \mathbf{v}_i\mathbf{v}_i^T)(RQ_{ij} + T - P_i).$$

The rotation and translation can be estimated by minimizing the sum of squares

$$S^2 = \sum_{i=1}^{n}\sum_{j=1}^{m_i}(RQ_{ij} + T - P_i)^T(I - \mathbf{v}_i\mathbf{v}_i^T)(RQ_{ij} + T - P_i). \tag{8.3.1}$$

Setting $\partial S^2/\partial T = 0$, the optimal solution for T can be found as

$$T = A^{-1}\sum_{i=1}^{n} m_i A_i(P_i - R\overline{Q}_i) \tag{8.3.2}$$

where $A_i = I - \mathbf{v}_i\mathbf{v}_i^T$, $A = \sum_i^n m_i A_i$, and

$$\overline{Q}_i = \frac{1}{m_i}\sum_{j=1}^{m_i} Q_{ij}$$

is the centre of the i^{th} group of data. From Appendix C in Chapter 4, A has an inverse whenever there exist at least two linearly independent line directions for the given lines.

For rotation matrix $R = (r_{ij})_{3\times 3}$, set vector

$$\widehat{R} = (r_{11}, r_{12}, r_{13}, r_{21}, r_{22}, r_{23}, r_{31}, r_{32}, r_{33})^T,$$

and for any vector $U = (u_1, u_2, u_3)^T$, set matrix

$$U^* = \begin{pmatrix} u_1 & u_2 & u_3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_1 & u_2 & u_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & u_1 & u_2 & u_3 \end{pmatrix}.$$

With these notations, the translation T in (8.3.2) can be further written as

$$T = \mathbb{P} - \mathbb{Q}^*\widehat{R} \tag{8.3.3}$$

where

$$\mathbb{P} = A^{-1}\sum_{i=1}^{n} m_i A_i P_i,$$

and

$$\mathbb{Q}^* = A^{-1}\sum_{i=1}^{n} m_i A_i \overline{Q}_i^*.$$

With (8.3.3), for each Q_{ij},

$$RQ_{ij} + T - P_i = (Q_{ij}^* - \mathbb{Q}^*)\widehat{R} - (P_i - \mathbb{P}).$$

For each i and each j, let $\overline{P}_i = P_i - \mathbb{P}$, $\overline{Q_{ij}^*} = Q_{ij}^* - \mathbb{Q}^*$, then

$$
\begin{aligned}
S^2 &= \sum_{i=1}^{n}\sum_{j=1}^{m_i}(\overline{Q}^*_{ij}\widehat{R} - \bar{P}_i)' A_i (\overline{Q}^*_{ij}\widehat{R} - \overline{P}_i) \\
&= \sum_{i=1}^{n} m_i \overline{P}'_i A_i \overline{P}_i - 2\sum_{i=1}^{n} \overline{P}'_i A_i \sum_{j=1}^{m_i} \overline{Q}^*_{ij}\widehat{R} \\
&+ \widehat{R}' \sum_{i=1}^{n}\sum_{j=1}^{m_i} \overline{Q}^{*\prime}_{ij} A_i \overline{Q}^*_{ij}\widehat{R}
\end{aligned}
\tag{8.3.4}
$$

To estimate the rotation, this sum needs to be further written as a function of rotation parameters, and a minimization procedure is then performed to estimate them.

When the region is represented as planes, the estimation procedure is roughly the same. When there are at least three linearly independent points in the intraoperative data for each region, a plane can be fitted and the problem of matching a group of data sets to a group of planes becomes the problem of matching two sets of undirected planes. If we fix the plane normals for one group of planes, then a rigid transformation can be computed using the method developed in section 4.3 for any selected normal directions of second group of planes. The best initial estimate for the rigid transformation should be the one that produces the smallest matching error. Another way to match the group of data sets to a group of planes is to use a minimization procedure. In the above minimization procedure for line regions, if we replace those matrices A_i with matrices $\mathbf{n}_i\mathbf{n}_i^T$, then we obtain a similar result.

Similarly, the initial rigid transformation can also be estimated by modeling the regions as a mixture of straight lines and planes.

Computing the initial rigid transformation by modeling each region as an ellipsoid

When regions represented by flat geometric objects cannot provide enough information for computing a good initial estimate for the unknown rigid transformation, a curved geometric object needs to be used to model the region. We choose to use an ellipsoid to represent a curved region because of its simplicity and its good geometric properties. When there are enough points in both the preoperative data and the intraoperative data for each region, an ellipsoid can be fitted to both of the two data sets. In this case, a rigid transformation can be computed using the centres and the orientations of the fitted ellipsoids. When too few intraoperative points are available, a minimization procedure is required to compute the initial rigid transformation, as in the case of modeling the regions as lines or planes.

8.4 Uniqueness of data matching

As can be seen, whether the true rigid transformation information can be retrieved depends on whether the data given can provide enough information to determine the unknown rigid transformation. When too few regions are selected or when the intraoperative data has too few sample points, or when the geometry of regions is too simple, it might not be possible to determine uniquely the rigid transformation that links the two groups of data sets. It is easy to show that when only two lines are considered, no matter how many points are sampled from the lines and how the two lines are positioned and oriented, it is impossible to determine the unknown rigid transformation purely from point-region matching, as in this case there are

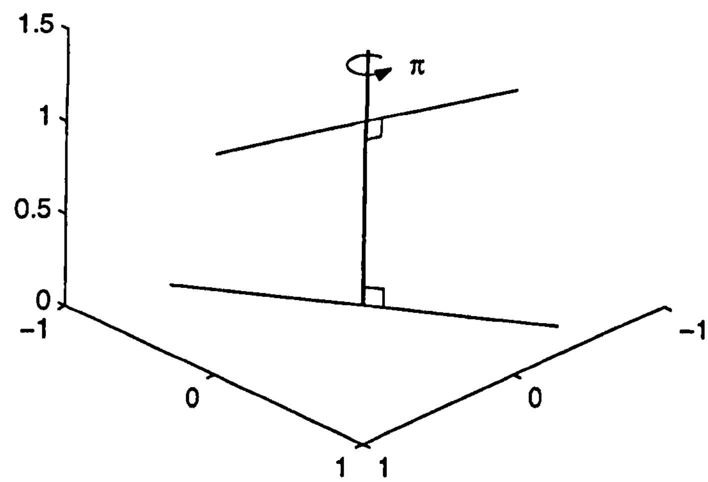

Figure 8.1: Two lines cannot determine the orientation of an object

always at least two rotations which rigidly transform the two sets of points onto their corresponding lines. Even in the case when more than two lines are used, there might be more than one way to match the given sets of points with their corresponding regions. In these cases, either the rotation or the translation cannot be estimated correctly. In this section, we briefly discuss how regions should be located and oriented, so that the rigid transformation can be determined uniquely.

8.4.1 Uniqueness for flat regions

We first consider the uniqueness for regions all represented as straight lines.

Proposition 8.4.1. *Let R_1, T_1, R_2, T_2 be two sets of rigid transformation such that they all transform two different points P_1, P_2 onto the line $\mathcal{L}(P, \mathbf{v})$. Let $\mathbf{n}$ and α be the rotation axis and angle of rotation $R_2 R_1^T$. Then $\mathbf{n} = \mathbf{v}$ or $\mathbf{n} \perp \mathbf{v}$ when $\alpha = \pi$.*

Proof. According to the assumption, $R_iP_1 + T_i$, $R_iP_2 + T_i \in \mathcal{L}(i = 1, 2)$. Thus there exist numbers $\lambda_i(i = 1, 2)$, $R_i(P_2 - P_1) = \lambda_i \mathbf{v}(\lambda_i \neq 0)$. From these relations, we have

$$R_2(P_2 - P_1) = \lambda_1 R_2 R_1^T \mathbf{v} = \lambda_2 \mathbf{v}.$$

This implies that $R_2R_1^T\mathbf{v} = \pm\mathbf{v}$, since $\|R_2R_1^T\mathbf{v}\| = \|\mathbf{v}\|$ and $\lambda_1 \neq 0, \lambda_2 \neq 0$. Let $R = R_2R_1^T$. If $R\mathbf{v} = \mathbf{v}$, the rotation axis will be $\mathbf{v}$. If $R\mathbf{v} = -\mathbf{v}$, the rotation axis must be perpendicular to $\mathbf{v}$, and at the same time the rotation angle must be π. In fact, let the rotation axis and the rotation angle be $\mathbf{n}$ and θ respectively. Then

$$R\mathbf{v} = cos\theta\mathbf{v} + (1 - cos\theta)(\mathbf{n} \cdot \mathbf{v})\mathbf{n} + sin\theta\mathbf{n} \times \mathbf{v} = -\mathbf{v}.$$

Since $\mathbf{v}$, $\mathbf{n}$ and $\mathbf{n} \times \mathbf{v}$ are linearly independent to each other, we must have $sin\theta = 0$, $cos\theta = -1$ and $\mathbf{n} \cdot \mathbf{v} = 0$. Therefore, $\mathbf{n}$ is perpendicular to $\mathbf{v}$ and $\theta = \pi$. □

From proposition 8.4.1, we have

Proposition 8.4.2. *Suppose that both rigid transformations R_1, T_1 and R_2, T_2 transform three groups of line points(with at least two points included in each group)onto three straight lines $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ respectively. We must have $R_1 = R_2$ and $T_1 = T_2$ if the following three condition hold:*

1. *The three straight lines do not have a common perpendicular line that intersects all three of the lines;*

2. *None of the three lines is the common perpendicular to the other two which also intersects the other two;*

3. *The three lines are not parallel to each other.*

Proof. Let the directions of the three lines be $\mathbf{v}_1$,$\mathbf{v}_2$,$\mathbf{v}_3$ respectively. The following observation should be noted. If two lines $\mathcal{L}_1$ and $\mathcal{L}_2$ are not parallel and do not intersect, they will have a common perpendicular P_1P_2, as shown in the figure 8.2. Suppose a rigid transformation maps $\mathcal{L}_1$ and $\mathcal{L}_2$ onto a second pair of lines $\mathcal{L}'_1$ and $\mathcal{L}'_2$ with a common perpendicular $P'_1P'_2$. Then the transformation also maps P_1 to P'_1 and P_2 to P'_2 respectively. To show that the rigid transformation is unique, we need only to show that the given lines will have at least 3 common perpendicular points under the given conditions. Let $R = R_2R_1^T$. According to the proposition 8.4.1, $R\mathbf{v}_i = \pm\mathbf{v}_i, i = 1, 2, 3$. (1) We first show that it is impossible for $R\mathbf{v}_i = -\mathbf{v}_i$ to be satisfied for one and only one unit vector in $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$. Without loss of generality, we assume that $R\mathbf{v}_1 = -\mathbf{v}_1$, $R\mathbf{v}_2 = \mathbf{v}_2$, $R\mathbf{v}_3 = \mathbf{v}_3$. If $\mathbf{v}_2 \neq \pm\mathbf{v}_3$, then both $\mathbf{v}_2$ and $\mathbf{v}_3$ will be the rotation axis of R, and this can only happen when R is the identity matrix I. This contradicts $R\mathbf{v}_1 = -\mathbf{v}_1$. Therefore, we must have $\mathbf{v}_2 = \pm\mathbf{v}_3$. In this case, the line $\mathcal{L}_2$ will be parallel to line $\mathcal{L}_3$. From proposition 8.4.1, $\mathbf{v}_1$ is perpendicular to rotation axis $\mathbf{v}_2 = \pm\mathbf{v}_3$. From condition 3, $\mathcal{L}_1$ is not parallel to $\mathcal{L}_2$ or $\mathcal{L}_3$. Let Ω be the plane determined by lines $\mathcal{L}_2, \mathcal{L}_3$. From condition 1, line $\mathcal{L}_1$ cannot be perpendicular

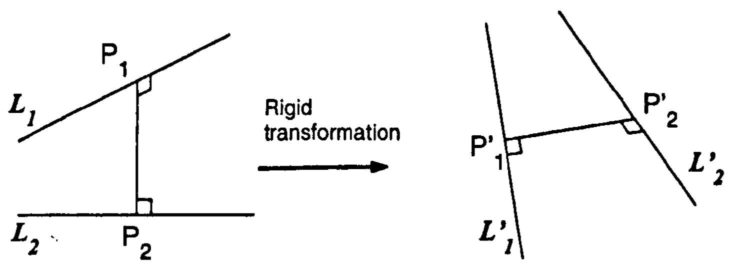

Figure 8.2: If a rigid transformation maps two unparallel lines $\mathcal{L}_1$ and $\mathcal{L}_2$ onto a second pair of lines $\mathcal{L}'_1$ and $\mathcal{L}'_2$, then the transformation also maps the common perpendicular points of the first pair of lines onto that of the second pair of lines.

to the plane since in this case, the three lines will have a common perpendicular line. From condition 2, $\mathcal{L}_1$ cannot intersect $\mathcal{L}_2, \mathcal{L}_3$ at the same time. Therefore, there must exist at least three non-collinear common perpendicular points for lines $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$. But this implies that $R_1 = R_2$, $T_1 = T_2$. This contradicts $R\mathbf{v}_1 = -\mathbf{v}_1$. This shows that we cannot have $R\mathbf{v}_i = -\mathbf{v}_i$ for $i = 1, 2, 3$.

(2) We then show that it is impossible for $R\mathbf{v}_i = -\mathbf{v}_i$ to be satisfied for any two unit vectors in $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$. Without loss of generality, we assume that $R\mathbf{v}_1 = \mathbf{v}_1$, $R\mathbf{v}_2 = -\mathbf{v}_2$, $R\mathbf{v}_3 = -\mathbf{v}_3$. Again, with proposition 8.4.1, $\mathbf{v}_1$ is perpendicular to both $\mathbf{v}_2, \mathbf{v}_3$. From condition 2, $\mathcal{L}_1$ cannot intersect both $\mathcal{L}_2$ and $\mathcal{L}_3$. Suppose that $\mathcal{L}_1$ does not intersect line $\mathcal{L}_2$. With condition 1, at least one end point of the common perpendicular to $\mathcal{L}_1, \mathcal{L}_3$ does not lie on the common perpendicular of $\mathcal{L}_1$ and $\mathcal{L}_2$. This means that $R_1 = R_2$, and $T_1 = T_2$, which contradicts the assumption that $R\mathbf{v}_1 = \mathbf{v}_1$, $R\mathbf{v}_2 = -\mathbf{v}_2$, $R\mathbf{v}_3 = -\mathbf{v}_3$. This shows that it is impossible for $R\mathbf{v}_i = -\mathbf{v}_i$ to be satisfied for any two unit vectors $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$.

(3) Finally, we show that $R\mathbf{v}_i = -\mathbf{v}_i$ for $i = 1, 2, 3$ cannot be true. If we do have such a case, then the rotation axis of R must be perpendicular to $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$. This implies that $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ are coplanar. From condition 3, we know that at least two lines are not parallel to each other. Suppose $\mathcal{L}_1, \mathcal{L}_2$ are not parallel to each other. From condition 1, the third line cannot intersect the common perpendicular of $\mathcal{L}_1$, $\mathcal{L}_2$. It can also be shown that in this case the common perpendicular end points of $\mathcal{L}_1$ and $\mathcal{L}_2$, $\mathcal{L}_1$ and $\mathcal{L}_3$, $\mathcal{L}_2$ and $\mathcal{L}_3$ cannot be collinear. Therefore we must have $R_1 = R_2$ and $T_1 = T_2$. This contradicts $R\mathbf{v}_i = -\mathbf{v}_i, i = 1, 2, 3$.

The only possibility now left is that $R\mathbf{v}_i = \mathbf{v}_i, i = 1, 2, 3$. Since at least two of the vectors $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ are linearly independent, we must have $R = I$, or equivalently,

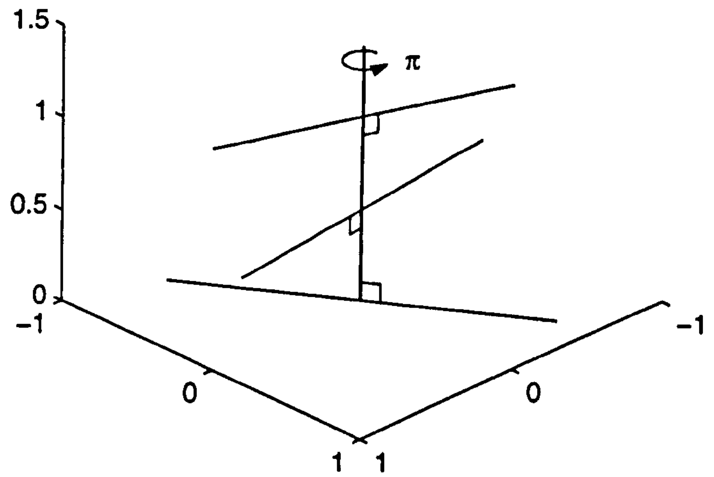

Figure 8.3: Three or more lines with a common perpendicular line cannot determine the orientation of an object

$R_1 = R_2$.

The proof of uniqueness of translation is direct when $R_1 = R_2 = R$. From condition 3, there exist at least two line directions, say $\mathbf{v}_1, \mathbf{v}_2$, that are linearly independent. If for points P_1, P_2, $RP_i + T_1$ and $RP_i + T_2$ are on line $\mathcal{L}_j (j = 1, 2)$, we must have $T_2 - T_1 = \lambda_1 \mathbf{v}_1 = \lambda_2 \mathbf{v}_2$. Thus we must have $T_2 = T_1$ as $\mathbf{v}_1$ and $\mathbf{v}_2$ are linearly independent. □

The above propositions demonstrate that there is only one possible position in space to arrange a given group of data points on their corresponding lines under the given conditions. Here we give some examples to show that when these conditions are violated, either the rotation or the translation cannot, in general, be uniquely determined. When three lines share a common perpendicular, a rotation of π about the common perpendicular will map the three lines onto themselves. When one line

$\mathcal{L}$ intersects and is the common perpendicular to the other two, the rotation about the line $\mathcal{L}$ by angle π will also map the three lines onto themselves. Obviously, when the three lines are all parallel to each other, the translation cannot be determined uniquely.

Proof of the uniqueness for a plane match is simpler by comparison. It follows directly that two planes cannot determine the position of an object, as the object can slide along the intersection line when the planes intersect, and can move between the two planes in any direction parallel to the two planes when the two planes are parallel. Even in the case of three mutually perpendicular planes, the orientation of an object still cannot be determined. Generally, we have the following result:

Proposition 8.4.3. *Let R_1, T_1, R_2, T_2 be two sets of rigid transformation such that they all transform three linearly independent points P_1, P_2, P_3 onto a plane Ω. Let* $\mathbf{v}$, $\mathbf{n}$ *be the normal of the plane and the axis of rotation $R_2R_1^T$ respectively. Then either* $\mathbf{n} = \mathbf{v}$ *or* $\mathbf{n} \perp \mathbf{v}$.

Proof. According to the assumption, $R_1P_j + T_1$, $R_2P_j + T_2 \in \Omega (j = 1, 2, 3)$. Then

$$[R_i(P_2 - P_1)] \cdot \mathbf{v} = 0, \quad [R_i(P_3 - P_1)] \cdot \mathbf{v} = 0, \quad (i = 1, 2).$$

Set $P = (P_2 - P_1) \times (P_3 - P_1)$. Then there exist $\lambda_1 \neq 0$ and $\lambda_2 \neq 0$ such that $R_1P = \lambda_1\mathbf{v}$, $R_2P = \lambda_2\mathbf{v}$. Hence,

$$\lambda_1 R_2 R_1^T \mathbf{v} = \lambda_2 \mathbf{v},$$

and this implies that

$$R_2 R_1^T \mathbf{v} = \pm\mathbf{v},$$

since $\lambda_1 \neq 0, \lambda_2 \neq 0$. Thus, for the rotation axis $\mathbf{n}$ of $R_2R_1^T$, either $\mathbf{n} = \mathbf{v}$ or $\mathbf{n} \perp \mathbf{v}$ when the rotation angle is π. $\square$

Proposition 8.4.4. *Let Ω_1, Ω_2, Ω_3 be three planes with normals $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ respectively. If $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ are linearly independent and no plane is perpendicular to the other two, then for two sets of rigid transformations R_1, T_1 and R_2, T_2, $R_1 = R_2$ and $T_1 = T_2$ when both rigid transformations transform three groups of nonlinear co-planar points onto the three planes respectively.*

Proof. Let the rotation axis of rotation $R = R_2 R_1^T$ be $\mathbf{n}$. With proposition 8.4.3, for $i = 1, 2, 3$, we have $R\mathbf{v}_i = \pm\mathbf{v}_i$. Firstly, it cannot be true that $R\mathbf{v}_i = -\mathbf{v}_i$, $i = 1, 2, 3$, since in this case, $\mathbf{n}$ will be perpendicular to all three linearly independent vectors, which is impossible. Secondly, since $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ are linearly independent of each other, it cannot be true that $R\mathbf{v}_i = \mathbf{v}_i$ for only one plane normal, which will imply that one of $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ is perpendicular to the other two, thus contradicting the given condition. What remains now is the case that $R\mathbf{v}_i = \mathbf{v}_i$ for at least two vectors in $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$, but this simply implies that $R = I$.

It is evident that $T_1 = T_2$ under the given condition when $R_1 = R_2$. □

When the conditions in proposition 8.4.4 are not satisfied, the rigid transformation cannot be determined uniquely. When the three normals are not linearly independent, the three planes will not intersect at a point and thus the translation cannot be uniquely determined. When one of $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$, say $\mathbf{v}_1$, is perpendicular to the other two, the intersection line of planes Ω_2 and Ω_3 will be parallel to $\mathbf{v}_1$. From figure 8.4, it can be seen that a rotation about the intersection line by an angle π maps the three planes onto themselves.

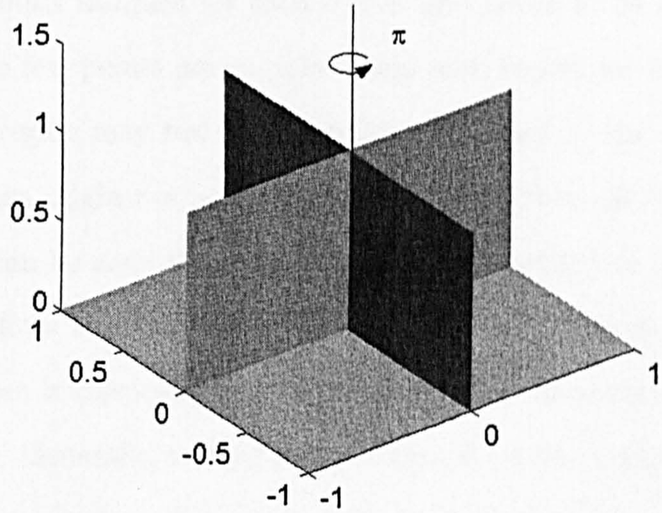

Figure 8.4: When one plane is perpendicular to the other two planes, it is impossible to determine the orientation of an object

8.4.2 Uniqueness for curved regions

In the previous subsection, we see that in some cases, the rigid transformation cannot be uniquely determined when regions are modeled with flat geometric objects. We have a similar problem with curved regions. For example, when the regions have a line of symmetry, a rotation about the axis of symmetry by an angle π results the same match.

Generally, when enough intraoperative data points are sampled for each region, two or three regions that do not have a line or plane of symmetry will be enough to determine the rigid transformation from geometric intuition.

The more general discussion of uniqueness for curved regions will be much more complicated and is an area for future research.

8.4.3 The number of points required for each region

The number of points sampled for each region also needs to be considered. As can be seen, when too few points are sampled from each region for intraoperative data, the shape of the region may not be properly represented by the data, and thus the rigid transformation might not be uniquely determined through region to region correspondence. It can be seen that when regions are modeled as lines, and only one point is sampled from some regions for intraoperative data, even if the lines satisfy the conditions given in proposition 8.4.2, the rigid transformation still cannot be determined properly. Generally, two points are required for line regions and three points are required for plane regions. For curved regions, more than four points are required, depending on the complexity of the shape of each region.

8.5 Refining the initial rigid transformation by fitting each region with a more precise implicit shape

The initial rigid transformation obtained from the above procedures can only guarantee that it is close to the true one, but it is far from accurate in general. This initial guess needs to be refined further by a more precise modelling with our constructive fitting techniques for the preoperative data.

Suppose there are n regions and the preoperative data for the i^{th} region has been represented as an implicit function $f_i(P) = 0$. The distance from each point P to the i^{th} region can be approximated by $|f_i(P)|$ or by $|f_i(P)|/||\nabla f_i(P)||$. The former is usually referred to as algebraic distance and the latter one is usually called the approximated Euclidean distance. Geometric distance from a point to an implicitly represented surface can also be considered, but it is computationally expensive [93]. Considering computational efficiency, we prefer to use algebraic distance or approximated Euclidean distance. Let $\{Q_{ij}\}_{j=1}^{m_i}$, $(i = 1, 2, \cdots, n)$ be the set of points sampled from the i^{th} region of the object under consideration in the operating theatre. Let R and T be the rotation and translation to be estimated. The total sum of squared distances can be represented by

$$S(R,T) = \sum_{i=1}^{n} \sum_{j=1}^{m_i} f_i^2(RQ_{ij} + T), \tag{8.5.1}$$

or by

$$S(R,T) = \sum_{i=1}^{n} \sum_{j=1}^{m_i} \left(\frac{f_i(RQ_{ij} + T)}{\|\nabla f_i(RQ_{ij} + T)\|} \right)^2. \tag{8.5.2}$$

The initial rotation and translation can then be refined by minimizing the sum of squared distances defined by (8.5.1) or by (8.5.2). Since each function $f_i(P)$ is of the form

$$f_i(P) = \sum_{j=1}^{n_i} g_{ij}(P) f_{ij}(P),$$

and those $g_{ij}(P)$ are just the gate functions, the value of $(\frac{f_i(P)}{\|\nabla f_i(P)\|})^2$ can therefore be approximated with

$$\sum_{j=1}^{n_i} g_{ij}(P) \left(\frac{f_{ij}(P)}{\|\nabla f_{ij}(P)\|} \right)^2.$$

This will greatly reduce the computation time because it is much simpler to compute the gradient $\nabla f_{ij}(P)$ than the gradient $\nabla(g_{ij}(P) f_{ij}(P))$.

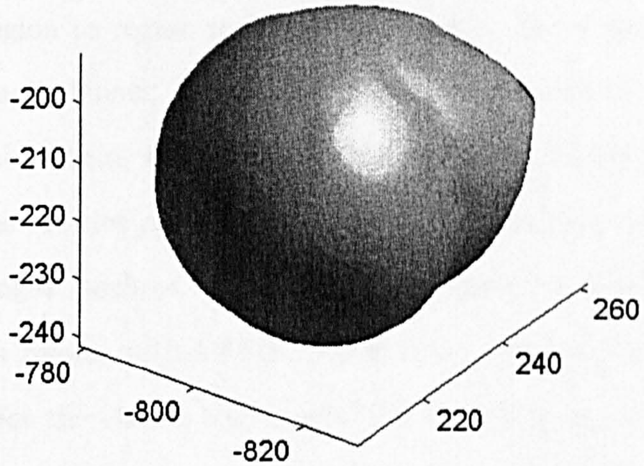

Figure 8.5: The implicit surface reconstructed from CT data of an actual plastic femur head

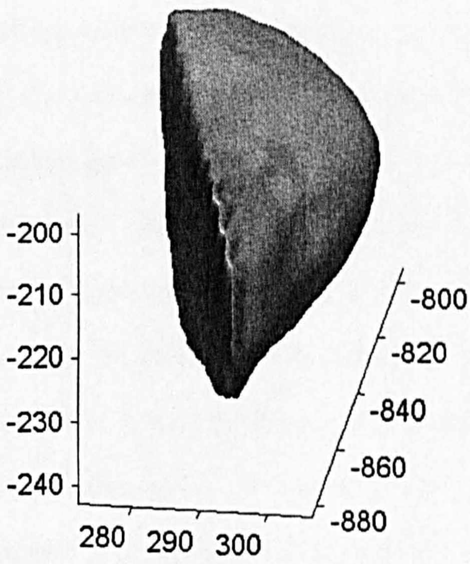

Figure 8.6: The surface reconstructed from CT data of an actual plastic femur in the area of lateral surface near the greater trochanter

8.6 Experimental results

In this section, some experimental results are given to demonstrate the matching results with the region to region registration strategy. The object considered in the experiment is a plastic femur. For the femur, four regions are considered: the femur head, the popliteal surface, the medial condyle and the lateral condyle (see figure 8.7). These regions are not only anatomically significant but can also be identified from computer images (such as CT images) more easily. A dense cloud of points is sampled from each region with OPMS. These data sets are then used for implicit surface modelling for the chosen region with the technique provided in the previous chapter. Having sampled the model data, the femur is placed in different positions with different orientations. For each position of the femur, a sparse set of points is sampled form each region to serve as the intraoperative data. To verify the registration accuracy, five points from the femur are used as reference marks to approximate accurately the actual rigid transformation that has been performed. To initialize the region to region matching, the curved centre for each region is computed for each set of intraoperative data. An initial guess is then obtained by reference marks registration using the quaternion algorithm. The estimated rotation is then parameterized with Euler angles. These initial rotation parameters plus the initial translation are then refined by minimizing the sum of squares errors given in equation (8.5.2) using the modified Levenbergue-Marquadt minimization routines [67]. As can be seen from table 8.1, the registration results are quite satisfactory. The error in rotation is measured by the Frobenius norm of the difference matrix between the rotation matrix obtained from reference marks registration and the one estimated from the region to region matching strategy. The error in translation is measured by the Euclidean

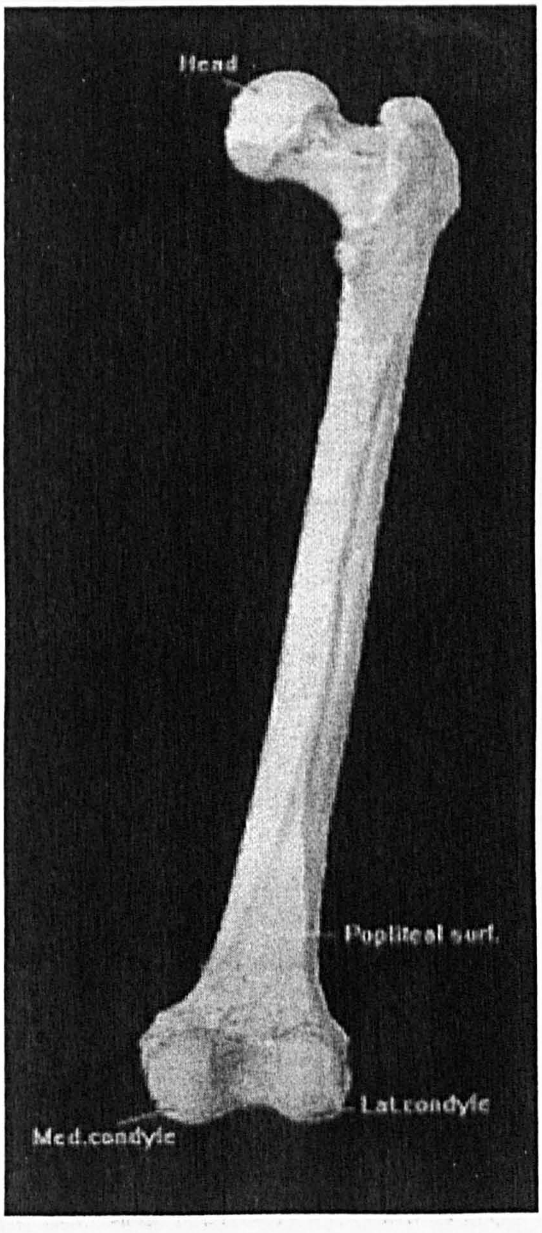

Figure 8.7: The relative positions of the four chosen regions

Femur pose	Rotation Error(mm)	Translation Error(mm)	Rms (mm)
pose 1	0.0042	2.715	0.1950
pose 2	0.0045	0.4072	0.2909
pose 3	0.0093	0.1599	0.4230
pose 4	0.0012	0.1676	0.2045
pose 5	0.0009	0.5291	0.0959
pose 6	0.0066	0.3163	0.4132
pose 7	0.0183	0.1607	0.6983
pose 8	0.0039	0.3103	0.3573
pose 9	0.0023	1.53	0.1682

Table 8.1: The rotation errors and translation errors between the rigid transformations estimated by region to region registration algorithm and the rigid transformations by reference mark registration, and the root mean square errors of corresponding reference marks for region to region registration.

distance between the two translations, in a similar way. The matching accuracy is measured by the root mean square error (RMS) between the five pairs of reference marks. The code is written in C and run under Microsoft Window NT4.0 using a PC with a Celeron 400mHz processor. The computation in all cases just takes less then a second.

Table 8.1 shows that the matching error is mainly affected by the error in rotation.

Generally speaking, the registration accuracy of our region to region matching strategy is influenced by several factors, for instance, the accuracy of model surface built, the size and the complexity of surface region chosen, the number of intraoperative data points sampled from each region, and how these sampled points are distributed on each region. In our experiment, to fit the sampled data accurately, the data from each area are subdivided into several subsets. The surface of medial condyle and the surface of the lateral condyle (Figure 8.8) of the femur are a combination of four quadric surfaces. The popliteal surface, unlike the other three areas, is a combination of four cubic surfaces which are the rigid transformation of the implicit

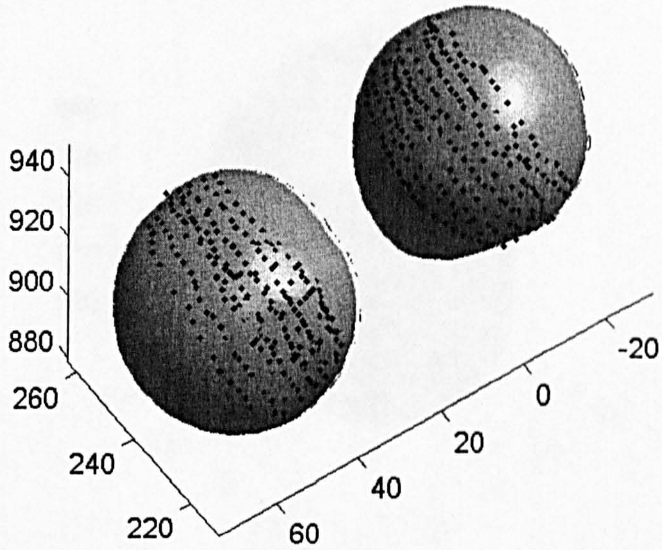

Figure 8.8: The OPMS data sets from the medial condyle and lateral condyle of the femur are represented with implicit surfaces

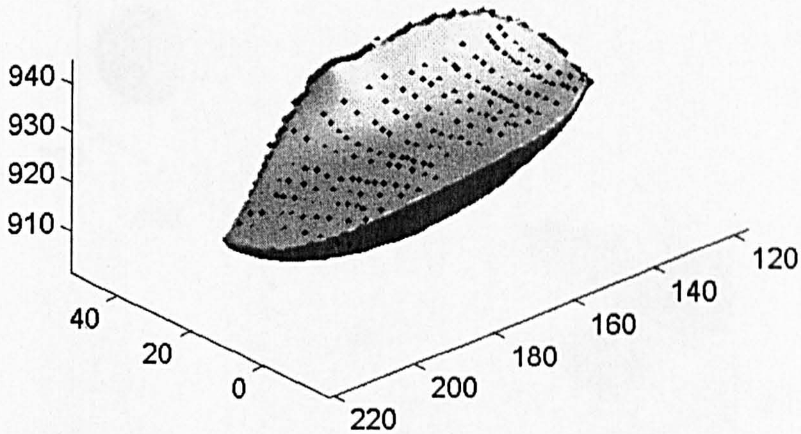

Figure 8.9: The OPMS data from the popliteal surface of the femur is fitted with an implicit surface

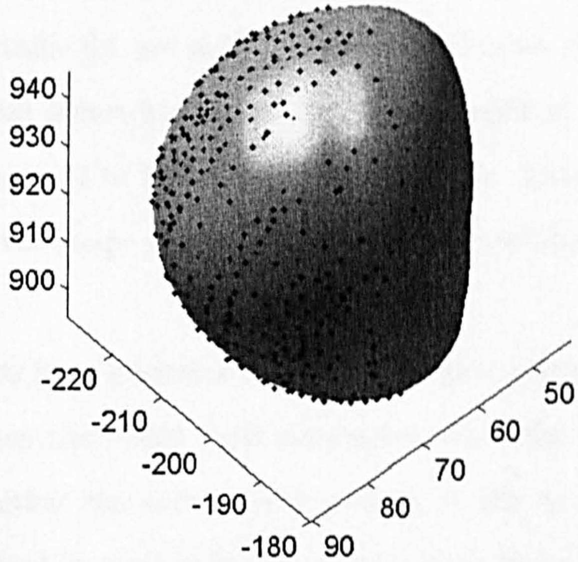

Figure 8.10: The OPMS data from the surface of the femur head is fitted with an implicit surface

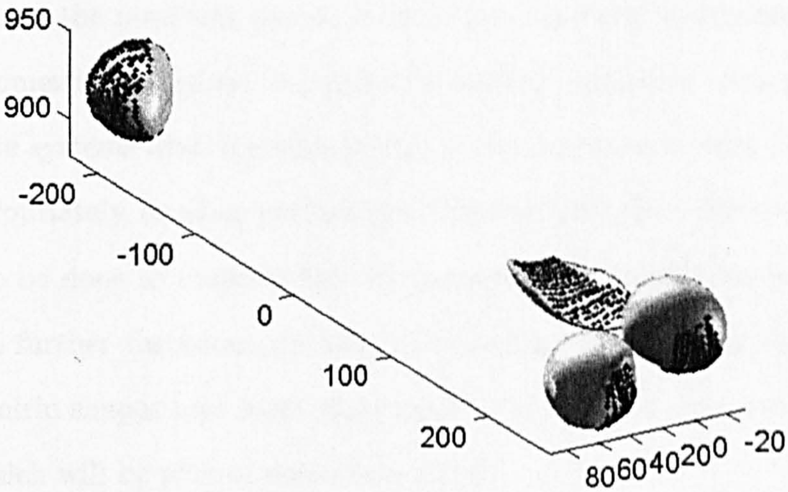

Figure 8.11: The relative positions of the four chosen regions

surface with form $f(x, y) - z = 0$ (Figure 8.9). The femur head surface is composed of 28 quadric surfaces combined with plane gates (Figure 8.10). The number of points sampled for intraoperative data depends on the size and the complexity of the region. The figures listed in table 8.1 are obtained with the number of points ranging from 20 to 43 sampled from femur head, 9 to 23 for the popliteal surface, 12 to 29 for the medial condyle, and 12 to 30 for the lateral condyle. These points are sampled quite evenly, so that the shape information can be captured by the data as much as possible.

In this chapter, we have presented a region to region registration method. This strategy first estimates the initial rigid parameters from the region to region correspondence, using either the curved data centers or the simple geometric fitting techniques. It is followed by a minimization procedure to refine the initial guess. The key technique for this matching procedure is the localized implicit fitting technique developed in the previous chapter. Experimental results given in this section show that our region to region matching strategy is quite satisfactory for both the matching accuracy and the matching speed. In addition, matching uniqueness is discussed for simple geometric primitives to emphasize that it might not be possible to align two coordinate systems from a region to region correspondence when the regions are chosen inappropriately, or when not enough points are sampled. However, much more work needs to be done to evaluate the effectiveness of the matching technique. This includes some further theoretical investigation on the matching uniqueness for more general geometric shapes and more experiments on different data sets from various situations, which will be part of our future work.

Chapter 9

Summary

The main theme investigated in this thesis is how to estimate the unknown rigid transformation that links the preoperative data and the intraoperative data in computer assisted surgery. As rotation is the main subject dealt with in the thesis, we have discussed systematically how a rotation can be parameterized. Though standard results can be found in the literature, we have presented them with our own way of inference and our own point of view. For example, in establishing the relationship between a quaternion and a rotation, we first discussed how a quaternion is linked with a four dimensional rotation. With this knowledge, exploring the link between a quaternion and a three dimensional rotation becomes investigating how a four dimensional rotation is linked with a three dimensional rotation. In this way, the quaternion representation of a rotation can be developed quite naturally.

Some properties of the rotation matrix have also been discussed in a very general way. We showed for example that, in general, the trace of an n dimensional rotation matrix cannot be larger than $n - 2 + 2c$ where c is any diagonal element of the rotation matrix. With this property, the proof of the modified SVD algorithm can be simplified.

As a starting point, we first surveyed the techniques for reference mark registration. It is in this thesis that we first mentioned the closed-form solution at the earliest given in 1966. Based on this investigation, the SVD algorithm, the quaternion algorithm and the polar decomposition algorithm were deduced in a much more direct and natural way. In addition, a new reference mark registration technique is developed from the idea of an estimate of the rotation axis.

As a direct generalization, we then developed the registration method when the reference objects are more general geometric primitives including points, lines and planes with the help of the concept of the center of these geometric objects. In addition, an interesting line fitting technique based on the geometric distance is given by solving an eigen system.

As far as non-landmark registration is concerned, two techniques are developed following two different ideas: Iterative closest geometric object matching and surface matching. With the first idea, iterative closest line segment registration and iterative closest triangle patch registration are developed. The proposed iterative procedures are always convergent when the data are properly scaled. Compared with the conventional ICP algorithm, the ICL and the ICT algorithms are much less sensitive to the initial guess.

The key point in the second idea is the implicit surface fitting. We begin the investigation on implicit fitting with ellipsoid fitting. The ellipsoid is the only quadric surface that is both centric and bounded. General quadric surface fitting techniques cannot guarantee to deliver an ellipsoid, while bounded implicit polynomial fitting techniques are not accurate enough when a good initial is not available. Our ellipsoid fitting technique is almost a closed form solution, in the sense that it is just a one step fitting in most cases. This fitting algorithm is used to find a good initial guess in our region to region matching strategy. For more complex shapes, a divide-conquer fitting procedure is adopted by using gate functions. With these functions, different implicit functions defined in different regions can be combined into a new implicit function with little distortion of the locally fitted functions.

Based on the constructive implicit surface fitting technique, region to region matching is developed. With this registration method, we first identify several regions on the surface of a bone object. Each region is then modelled with the preoperative data as an implicit function which serves as the distance function from a point to the surface. The matching procedure then becomes a search for the rotation and translation that minimizes the sum of 'distances' from intraoperative data to their corresponding regions. As a good initial estimate can be computed from region to region correspondences, the minimization procedure can be accomplished by a minimization technique. We have chosen the modified Levenberg-Marquardt algorithm for our minimization task. Initial experiment results shows that region to region matching is quite satisfactory though much more experiments needs to be done to show its effectiveness in general situation.

Chapter 10

Future work

Future research will focus mainly on two areas: the practical integration of the work described in this thesis into the real computer assisted orthopaedic system currently under development within the department, and the application of implicit fitting in computer vision and computer graphics.

10.1 The implementation of the developed matching method into actual computer-assisted surgery systems

In this thesis, the focus has been on the development of 3D data matching algorithms. Most of the work has been devoted to the theoretical aspect, to show the soundness and the robustness of the methods developed. Very little attention is given to the applications of the algorithms on particular image modalities like CT, MRI and X-ray. The immediate objective is to implement the methods developed into an actual computer assisted surgery system, and to apply the algorithms to different images formats.

10.2 More experiments and further investigation on matching uniqueness for curved regions

Though some initial experiments have been carried out on the region to region matching strategy, it is far from enough to show its effectiveness in general, more experiments will be done with both synthetic data and data sampled from actual bone surfaces by various scanners. In addition, the matching uniqueness for more general geometric shapes will be investigated. The idea of matching uniqueness discussed in chapter 8 is mainly for flat regions. For curved regions, the situation will be much more complicated, and we do not have a clear idea on when curved regions can determine the position and orientation of an object uniquely, for the given intraoperative data. Further research will be needed to discover how the given regions should be positioned and oriented, and how much intraoperative data is required, and how it should be distributed over the regions so that the matching can be determined uniquely.

10.3 Further experiments to compare the ICL and ICT algorithms with the ICP algorithm

The experiments carried out in Chapter 5 to compare the ICL (the ICT) algorithm with the ICP algorithm were done only for synthetic data. More experiments will be done to compare their robustness with noisy data or with data from surfaces of some actual geometric objects, which will provide a more complete comparison on their performances in different circumstances.

10.4 Non-rigid medical data registration

In this thesis, we have only considered how to align two coordinate systems by using data sets from a rigid object. The algorithms developed can only be used in computer assisted orthopaedic systems. In computer assisted surgery systems for soft tissue operations, alignment of two coordinate systems based on data sets from soft objects needs to be considered. Soft tissue medical data registration is a more challenging research area. In addition to rigid transformation, changes in object shape due to deformation like shearing and tearing need to be considered in the shape matching procedure. We will consider the problem by fitting the preoperative data with an implicit surface. The reason for this is that when the surface of the considered object is fitted with an implicit function $f(P)$, the signed distance from a point $P \in \mathbb{R}^3$ to the object can be approximated by the value of function f at the point P. Therefore, the surface of the deformed object can be approximated with a continuous topological transformation of the function $f(P)$. We will discuss what effects different deformation operations will have on an implicitly defined function and investigate the possible matching algorithms.

10.5 The application of implicit fitting to robotic boundary detection

In robotics and automation, we often need to constrain the hand of an robot to move within a certain area. In this case, the problem can be solved very easily when the boundary of the object is modelled with an implicit function $f(P)$. This is because the implicitly fitted function naturally divides space into three parts: the set of points on the boundary and the two sets of points that fall on either side of the boundary, corresponding to whether $f(P) = 0$, $f(P) < 0$, or $f(P) > 0$. For each movement, the robot need only check the value of the function to ensure that the sign does not change during movement. However, more research and experiments need to be done to show the efficiency of the way in controlling the movement of the robot hand.

10.6 Solid geometric object modeling using gate functions

Constructive Solid Geometry (CSG) is the process of building solid objects from other simple geometric primitives like the sphere, the ellipsoid, and the cube. In CSG, an object is regarded as a set of points, and the procedure for constructing an geometric object is just a series of set-theoretic operations on implicitly defined geometric objects. The three CSG operators are Union, Intersection, and Difference. Each operator acts upon two objects and produces a single object result. By combining multiple levels of CSG operators, complex objects can be produced from simple primitives. The drawback of this constructive procedure is that the functions constructed will be very complicated and not smooth enough. Unlike traditional solid geometric construction procedures, an object can be built by combining piecewise implicit functions using gate functions. Current CSG techniques are mainly concerned with how to combine two implicitly defined shapes, but little attention is paid to how accurately the original shapes are preserved. However, with our implicit surface combination technique, simple implicitly defined geometric shapes can be combined into very complicated surfaces given that the influence domain of each function is known. This is typically important in reconstructing the shape of an actual geometric shape precisely in computer graphics. The investigation of the application of gate functions to constructive solid geometry will be given, and software for building geometric graphics from real world data can be exploited.

Bibliography

[1] A. Albert. Conditions for positive and nonnegative definiteness in terms of pseudoinverse. *SIAM J. Appl. Math.*, 17:434–440, 1969.

[2] A. Albert. *Regression and the Moore-Penrose pseudoinverse.* Academic Press, New York, 1972.

[3] S. L. Altmann. *Rotation, Quaternion, and Double Groups.* Oxford University Press, New York, 1986.

[4] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:698–700, 1987.

[5] N. Ayache. *Artificial Vision for Mobile Robots: Stereo Vision and Multisensory Perception.* MIT, Cambridge, Massachusetts, 1991.

[6] C. Bajaj. Higher-order interpolation and least-squares approximation using implicit algebraic surfaces. *ACM Transactions on Graphics*, 12(4):327–347, October 1993.

[7] E. Bardinet, L. D. Cohen, and N. Ayache. A parametric deformable model to fit unstructured 3D data. *INRIA No. 2617*, 1995.

[8] P. A. Beardsley. Pose estimation of the human head by modelling with an ellipsoid. *Proc. of IEEE Conf on Automatic Face & Gesture Recognition*, Nara, 1998.

[9] R. Bellman. *Introduction to Matrix Analysis.* SIAM, Philadelphia, 1997.

[10] P. J. Besl and N. D. McKay. A method for registration of 3D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

[11] M. M. Blane, Z. Lei, and D. B. Cooper. The 3L algorithm for fitting implicit polynomial curves and surface to data. *Technical Report, Brown University LEMS Lab*, 1997.

[12] G. Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):849–864, November 1988.

[13] P. Borrel. Simple constrained deformations for geometric modelling and interactive design. *ACM Transactions on Graphics*, 13(2):137–155, April 1994.

[14] L. G. Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376, December 1992.

[15] G. Burel and H. Henocq. Determination of the orientation of 3D objects using spherical harmonics. *Graphical Models and Image Processing*, 57(5):400–408, September 1995.

[16] O. Carmichael, D. F. Huber, and M. Hebert. Large data sets and confusing scenes in 3D surface matching and recognition. *Proceedings of the Second International Conference on 3D Digital Imaging and Modelling (3DIM'99)*, pages 358–367, October 1999.

[17] A. V. Cideciyan. Registration of ocular fundus images. *IEEE Engineering in Medicine and Biology*, pages 52–58, January/February 1995.

[18] C. Davatzikos. Special transformation and registration of brain image using elastically deformable models. *Computer Vision and Image Understanding*, 66(2):207–222, May 1997.

[19] D. DeCarlo and D. Metaxas. Adaptive shape evolution using blending. *Proceedings ICCV'95*, pages 834–839, 1995.

[20] D. DeCarlo and D. Metaxas. Blended deformable models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):443–448, 1996.

[21] D. DeCarlo and D. Metaxas. Shape evolution with structural and topological changes using blending. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1186–1205, 1998.

[22] M. Dhome, M. Richetin, J. Laprest, and G. Rives. Determine of attitude of 3D objects from a simple perspective view. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 11(12):1265–1278, December 1989.

[23] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, January 1994.

[24] P. A. ven den Elsen, E. J. D. Pol, and M. A. Viergever. Medical image matching-a review with classification. *IEEE Engineering in Medicine and Biology*, pages 26–38, March 1993.

[25] M. T. Ensz, M. A. Ganter, and D. W. Storti. Implicit function alteration via radius mapping and direct function modification. *http://www .me .washington .edu /~cdam /People /Mark /papers /markpaper /ensz.asme.html*, 1999.

[26] A. C. Evans, D. L. Collings, P. Neelin, and T. S. Marrett. Correlative analysis of three-dimensional brain images. In R. H. Taylor, S. Lavallé, and G. C. Burdea, editors, *Computer Integrated Surgery-Technical and Clinical Application.* MIT press, 1996.

[27] H. Fan. On an earth ellipsoid best-fitted to the earth surface. *Journal of Geodesy*, 72(9):511–515, 1998.

[28] T. J. Fan, G. Medioni, and R. Nevatia. Recognition 3D objects using surface descriptions. *IEEE Transaction on Pattern Analysis and Matching Intelligence*, 11(11):1140–1156, November 1989.

[29] J. L. Farrell, J. C. Stuelpnagel, R. H. Wessner, and J. R. Velman. Solutions to problem 65-1. *SIAM Review*, 8(3):384–386, Jul. 1966.

[30] O. D. Faugeras and M. Hebert. The representation, recognition, and locating of 3D objects. *The International Robotics Research*, 5(3):27–52, 1986.

[31] J. Feldmar. Extension of the ICP algorithm to non-rigid intensity based registration of 3D volumes. *Computer Vision and Image Understanding*, 66(2):193–206, May 1997.

[32] J. Feldmar, N. Ayache, and F. Betting. 3D-2D projective registration of free-form curves and surfaces. *Computer Vision and Image Understanding*, 65(3):403–424, March 1997.

[33] A. Fitzgibbon, M. Pilu, and R. B. Fisher. Direct least square fitting of ellipses. *IEEE Transaction on Pattern Analysis and Machine intelligence*, 21(5):476–480, May 1999.

[34] M. A. Ganter and D. W. Storti. Implicit solid modelling: A renewed method for geometric design. *http://swhite .me .washington .edu /~cdam /ISMASME /ismasme.html*, 1999.

[35] D. Goryn and S. Hein. On the estimation of rigid body rotation from noise data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1219-1219, 1995.

[36] A. R. Gourlay and G. A. Watson. *Computational Methods for Matrix Eigenproblems.* John wiley & sons, London, 1973.

[37] A. A. Grattarola. Volumetric recognition from object silhouettes: A regularization procedure. *Signal Processing*, 27:27–35, 1992.

[38] J. W. Harris and H. Stocker. *Handbook of Mathematics and computational Science.* Springer-Verlag, New York, 1998.

[39] B. K. P. Horn. Closed -form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, 4(4):629–642, April 1987.

[40] B. K. P. Horn. Closed-form solution of absolute orientation using orthonormal matrices. *J. Opt. Soc. Am. A*, 5(7):1127–1135, 1988.

[41] J. B. A. Maintz and M. A. Viergever. A survey of medical image registration. *Medical Image Analysis*, 2(1):1–36, March 1998.

[42] F. John. Extremum problems with inequities as subsidiary conditions(1948). In J. Moser, editor, *Fritz John Collected Papers*, volume 2, pages 543-560. Birkhauser, Boston, 1985.

[43] B. Kamgar-Parsi and B. Kamgar-Parsi. Matching sets of 3D line segments with application to polygonal arc matching. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 19(10):1090–1099, December 1997.

[44] K. Kanatani. Analysis of 3D rotation fitting. *IEEE Transaction on Pattern Analysis and Matching Intelligence*, 16(5):543–549, May 1994.

[45] W. C. Karl, G. C. Verghese, and A. S. Willsky. Reconstructing ellipsoids from projections. *CVGIP-Graphical models and image processing*, 56(2):124–139, 1994.

[46] D. Keren and D. Cooper. Describing complicated objects by implicit polynomials. *IEEE Transaction on Pattern Analysis and Machine intelligence*, 16(1):38–53, January 1994.

[47] D. Keren and C. Gotsman. Fitting curves and surfaces with constrained implicit polynomials. *IEEE Transaction on Pattern Analysis and Machine intelligence*, 21(1):31–41, January 1999.

[48] D. Kozinska, O. J. Tretiak, J. Nissanov, and C. Ozturk. Multidimensional alignment using the Euclidean distance transformation. *Graphical Models and Image Processing*, 59(6):373–387, 1997.

[49] Y. S. Kwoh, J. Hou, E. A. Jonckeere, and S. Hayati. A robot with improved absolute positioning accuracy for CT guided stereotactic brain surgery. *IEEE Transaction on biomedical Engineering*, 35:153–160, 1988.

[50] P. Lancaster. *Curve and Surface Fitting.* Academic Press, London, 1986.

[51] A. Langenbucher and B. Seitz et al. Ellipsoidal fitting of corneal topography data after arcuate keratotomies with compression sutures. *Ophthalmic Surgery and Lasers*, 29(9):Clinical Science(abstract), 1998.

[52] S. Lavallé. Registration for computer integrated surgery: Methodology, stâte of art. In R. H. Taylor, S. Lavallé, and G. C. Burdea, editors, *Computer Integrated Surgery-Technical and Clinical Application.* MIT press, 1996.

[53] S. Lavallé and J. Troccaz et al. Computer-assisted spinal surgery using anatomy-based registration. In R. H. Taylor, S. Lavallé, and G. C. Burdea, editors,

Computer Integrated Surgery-Technical and Clinical Application. MIT press, 1996.

[54] S. Lavallé, R. Szeliski, and L. Brunie. Anatomy-based registration of 3D medical images, range images, x-ray projections, and three- dimensional models using octree-splines. In R. H. Taylor, S. Lavallé, and G. C. Burdea, editors, *Computer Integrated Surgery-Technical and Clinical Application.* MIT press, 1996.

[55] Z. Lei, M. M. Blane, and D. B. Cooper. 3L fitting of higher degree implicit polynomials. *Proceedings of Third IEEE Workshop on Applications of Computer vision,* pages 148–153, December 1998.

[56] Q. Li and J. G. Griffiths. Some algorithms for coordinate systems alignment. *Technique report, Department of Computer Science, University of Hull,* 1998.

[57] Q. Li and J. G. Griffiths. Iterative closest geometric object registration. *Computers and Mathematics with applications,* 40:1171–1188, 2000.

[58] C. Liao and G. Medioni. Surface approximation of a cloud of 3D points. *Graphical Models and Image Processing,* 57(1):67–74, January 1995.

[59] J. A. Little, D. L. G. Hill, and D. J. Hawkes. Deformations incorporating rigid structures. *Computer Vision and Image Understanding,* 66(2):223–232, May 1997.

[60] M. Desbrun M. P. Cani-Gascuel. Animation of deformable models using implicit surfaces. *IEEE Transactions on Visualization and Computer Graphics,* 3(1), 1997.

[61] P. G. Maillot. *Using quaternion for coding 3D Transformations, Graphics GEMS,* pages 498–515. Academic Press, 1990.

[62] W. H. Marlow. *Mathematics for Operation Research.* John wiley & sons, New York, 1978.

[63] L. Mirsky. Diagonal elements of orthogonal matrices. *American Mathematical Monthly*, 66(1):19–22, Jan. 1959.

[64] D. Moore and J. Warren. Approximation of dense scattered data using algebra surfaces. *Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences, Kauai, HI*, pages 681–690, 8-11,January 1991.

[65] M. Moshfeghi and H. Rusinek. Three dimensional registration for multimodality medical images using the principle axes techniques. *Philips J. Research*, 47:81–97, 1992.

[66] S. Muraki. Volumetric shape description of range data using 'blobby model'. *Computer Graphics*, 25:227–235, January 1991.

[67] netlib. http://www.netlib.org/minpack/index.html.

[68] A. Pasco, V. Adzhiev, A. Sourin, and V. Savchenko. Function representation in geometric modelling: Concepts, implementation and applications. *the Visual Computer*, 11(8):429–446, 1995.

[69] C. A. Pelizzari, G. T. Y. Chen, D. R. Spelbring, R. R. Weichselbaum, and C. T. Chen. Accurate three-dimensional registration of CT, PET, and/or MR images of the brain. *J Comput Assist Tomogr*, 13(1):20–26, January 1989.

[70] A. P. Pentland. Perceptual organization and the representation of natural form. *Readings in computer vision, M. Kaufmann publisher, Los Altos, CA*, pages 680–699, Nara, 1987.

[71] F. H. Post, W. C. de Leeuw, I. A. Sadarjoen, F. Reinders, and T. van Walsum. Global, geometric, and feature-based techniques for vector field visualization.

In F. H. Post and D. Silver, editors, *Future Generation Computer Systems*, volume 15, pages 87–98. Elsevier, 1999.

[72] V. Pratt. Direct least-squares fitting of algebraic surfaces. *Computer Graphics*, 21(4):145–151, July 1987.

[73] A. Racci. A constructive geometry for computer graphics. *The Computer Journal*, 16(2):157–160, 1973.

[74] I. Ragnemalm. The Euclidean distance transform in arbitrary dimensions. *Pattern Recognition Letters*, 14:883–888, 1993.

[75] F. Reinders, M. E.D. Jacobson, and F. H. Post. Skeleton graph generation for feature shape description. In W. de Leeuw and R. van Liere, editors, *Data Visualization 2000*, pages 73–82. Springer Verlag, 2000.

[76] F. Reinders, F. H. Post, and H. J. W. Spoelder. Attribute based feature tracking. In E. Groller, H. Loffelmann, and W. Ribarsky, editors, *Data Visualization '99*, pages 63–72. Springer Verlag, 1999.

[77] F. Reinders, H. J.W. Spoelder, and F. H. Post. Experiments on the accuracy of feature extraction. In D. Bartz, editor, *Visualization in Scientific Computing '98*, pages 49–58. Springer Verlag, 1998.

[78] E. Rimon and S. P. Boyd. Efficient distance computation using best ellipsoid fit. *Technical Report*, Stanford University, Department of Electrical Engineering, 1992.

[79] A. P. Rockwood. The displacement method for implicit blending surfaces in solid models. *ACM Transactions on Graphics*, 8(4):279–297, October 1989.

[80] R. F. Sarraga. Computer modelling of surfaces with arbitrary shapes. *IEEE Computer Graphics and Applications*, 10:67–75, March 1990.

[81] V. Savchenko and A. Pasko. Implicit curved polygons. *Technical Report 96-1-004, University of Aizu, Japan*, 1996.

[82] V. V. Savchenko, A. A. Pasko, O. G. Okunev, and T. L. Kunii. Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum*, 14(4):181–188, 1995.

[83] H. R. Schwarz. *Numerical analysis of symmetric Matrices.* Prentice-Hall, Englewood Cliffs, New Jersey, 1973.

[84] L. Serra, T. Poston, and N. Hern et al. Interaction techniques for a virtual workspace. *http://www .informatik .umu .se/~jwworth /vrst95-1.html*, 1996.

[85] A. Shapiro. Real functions for representation of rigid solids. *J. Computer Aided Geometric Design*, 11:153–175, 1994.

[86] A. Sherstyuk. Shape design using convolution surface. *http://www .ugcs .caltech .edu /~andrei /papers*, Technical Report, 1999.

[87] K. Shoemake. *Euler Angle Conversion , Graphics GEMIV*, pages 222–229. AP Professional, 1994.

[88] K. Shoemake. *Polar Matrix Decomposition , Graphics GEMIV*, pages 207–221. AP Professional, 1994.

[89] R. J. Smith, W. K. Chan, and R. K. Maloney. The prediction of surgically induced refractive change from corneal topography. *American Journal of Ophthalmology*, 125(1):44–53, 1998.

[90] B. Spain. *Analytical Quadrics.* Pergamon Press, Oxford, 1960.

[91] G. W. Steward. *Introduction to Matrix Computation.* Academic Press, Orlando, Florida, 1973.

[92] J. Stuelpnagel. On the parametrization of the three-dimensional rotation group. *SIAM Review*, 6:422–430, 1964.

[93] S. Sullivan, L. Sandford, and J. Ponce. Using geometric distance fits for 3D object modelling and recognition. *IEEE Transaction on Pattern Analysis and Machine intelligence*, 16(12):1183–1196, December 1994.

[94] G. Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Transaction on Pattern Analysis and Machine intelligence*, 13(11):1115–1137, November 1991.

[95] G. Taubin, F. Cukierman, S. Sullivan, J. Ponce, and D. J. Kriegman. Parameterized families of polynomials for bounded algebraic curve and surface fitting. *IEEE Transaction on Pattern Analysis and Machine intelligence*, 16(3):287–303, March 1994.

[96] U. Tiede, K. H. Hoehne, M. Bomans, A. Pommer, M. Riemer, and G. Wiebecke. Investigation of medical 3D-rendering algorithms. *IEEE Computer Graphics and Applications*, 10(2):41–53, March 1990.

[97] K. D. Toennies, J. K. Udupa, and G. T. Herman. Registration of 3D objects and surfaces. *IEEE Computer Graphics and Applications*, pages 52–62, May 1990.

[98] N. Tsingos, E. Bittar, and M. Gascuel. Implicit surfaces for semi-automatic medical organ reconstruction. *Computer Graphics: Developments in Virtual Environments(Proc. Computer Graphics International 95, R A Ernshaw and J A Vince eds)*, May 1995.

[99] S. Uneyama. Least-square estimation of transformation parameters between two point patterns. *IEEE Transaction on Pattern Analysis and Machine intelligence*, 13(4):376–380, April 1991.

[100] M. Unser and P. thevenaz et al. Registration and statical analysis of PET images using the wavelet transformation. *IEEE Engineering in Medicine and Biology*, 14(5):603–611, September/October 1995.

[101] G. Wahba. Problem 65-1: A least square estimate of satellite attitude. *SIAM Review*, 7(3):409, Jul. 1965.

[102] S. X. Wang, D. Silver, and S. Bhat. Extraction and quantification in 3D dataset:user manual. *http://www .caip .rutgers .edu /~xswang /object-segment /objhtml /objhtml.html*, 1995.

[103] J. Warren. Blending algebraic surfaces. *ACM Transactions on Graphics*, 8(4):263–278, October 1989.

[104] A. Watt and M. Watt. *Advanced Animation and Rendering Techniques, theory and practice.* Addison-Weley, ACM Press, New York, 1992.

[105] E. Weisstein. quaternion , Eric Weisstein's World of Mathematics. *http://mathworld .wolfram .com /quaternion.html.*

[106] E. Weisstein. Heaviside step function, Eric Weisstein's World of Mathematics. *http://mathworld .wolfram .com /HeaviSideStepFunction.html.*

[107] H. Weyl. *The Classical Groups.* Princeton University Press, Princeton, New Jersey, 1946.

[108] M. V. Wickerhauser. *Adapted Wavelet Analysis from Theory to Software.* A K Peters, Wellesley Massachusetts, 1994.