MULTIPLE 3D SCAN DATA REGISTRATION

Ran Wang

A thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science
Concordia University
Montréal, Québec, Canada

July 2008

Your file Votre référence
ISBN: 978-0-494-42536-7
Our file Notre référence
ISBN: 978-0-494-42536-7

Abstract

MULTIPLE 3D SCAN DATA REGISTRATION

RAN WANG

A 3D geometric model representing the surface of a 3D object is most often used for synthesizing highly realistic 3D images. The advent of 3D scanners has made the acquisition of 3D surface geometry of existing objects relatively simple. In most 3D scanners, it is difficult, to acquire the complete surface geometry in one operation. For example, with a single fixed scan head, the object has to be rotated many times to present different views of the object's surface to the scan-head, so that the entire surface geometry is covered. Then, all the scan data sets from different views have to be combined from their own independent coordinate systems into a single consistent geometric coordinate space. This operation is called scan data registration. Most existing techniques mainly focus on registering two point data sets at a time. For scan data in multiple views, say n views, one has to resort to pair-wise registration, requiring n-1 registration operations. In this research, we study the problem of multiple scan data registration. In particular, we are interested in efficiency, robustness and accuracy of registration techniques. Our main contribution is a new registration technique that can simultaneously register any number of scan data sets as long as there is some overlap amongst them, thus requiring considerably less than n-1 registration operations. To demonstrate the performance of our approach, we have developed a registration system and we have experimented with a number of scan data sets and shown the improvements resulting from the use of our approach.

Acknowledgments

I would like express heartfelt thanks to my supervisor, Dr. Sudhir P. Mudur. During the period of my graduate study in Concordia University, he always guided me to the right direction not only in the academy, but also in research attitude. His enthusiasm of working set a great example for me. I also wish to thank Mr. Sushil Bhakar. He always listens with great patience to my questions and gives me good suggestions. I could not have finished my thesis without their help.

I would also thank all my colleagues in the graphics and visualization lab. They give me lots of help and let me learn knowledge of different fields.

I would like to thank my parents to give me the chance to come to Canada to further my graduate study. A special thanks to my wife; she gives me great support during the days of my life.

TABLE OF CONTENTS

List of Figures

List of Tables

Chapter 1: Introduction

Affordability with good performance has resulted in increased use of 3D scanners in domains like gaming, cinema, rapid prototyping and heritage documentation. Many of these scanners, such as the hand-held or desktop versions, can be used to acquire 3D sample points on the surfaces of complex objects with minimal or no significant set-up costs. The raw data resulting from the scanning operation consists primarily of range images from different views of an object [Pulli 1997]. Increasingly these days, with the availability of appropriate color cameras, these range images are augmented with corresponding color (texture) images. The number of scans needed in order to completely cover the surface of the object depends on the 3D object surface complexity, particularly the extent to which the object occludes parts of itself from a given view point. Typically, when the scanning is carried out in different passes (different facets of the object's surface are presented to the scanner camera head), each range image is in its own independent coordinate system, and the geometric relationship among these coordinate systems is only approximately known. The number of different scans needed to acquire the complete 3D surface data of an object varies from tens to hundreds of views (See Table 1). Hence, the volume of data is very large [Levoy 2000] and needs to merged to produce a 3D surface representation readily usable by standard 3D modeling and rendering programs. Sometimes this process is simplified by capturing data from precisely known camera locations recorded using appropriately calibrated scanning hardware set-up. But a priori

calibration is not always possible, for example when using a hand-held or a desktop scanner or when scanning a large building in multiple phases.

Model	# of Scans	Source
Bunny	10	Stanford Repository
Happy Buddha	60	Stanford Repository
Dragon	70	Stanford Repository
Armadillo	114	Stanford Repository
Lucy	47	Stanford Repository
Thai Statue	36	Stanford Repository
Frog	16	Local 3D Scanner
Rabbit	24	Local 3D Scanner

Table 1: Number of Views for 3D Scanned Objects

One traditional approach to handle this kind of data is to first convert all the datasets into polygonal meshes and then to register and merge these meshed surface patches in a final step [Turk 1994]. As the 3D point data acquired in each scan is in its own independent coordinate system, the process of registration determines the transformation to be associated with each scan, so as to bring all the scan data points into a single coordinate space. If the scan-head and object positions for the different scans are known exactly in some global coordinate system, then registration is simple. Otherwise, registration requires one to automatically determine the individual transformations to be associated with each scan, based on the content of the scans. Once the registration process is complete, merging two or more overlapping meshes mandates the complex geometric operation of determining the exact interference among polygonal meshes, and also the operation of

combining the disconnected meshes into a single connected mesh. This can require considerable manual effort. On the other hand, recent advances in point based graphics enable us to work only with point samples on the surface [Pfister 2004]. They do not require an explicit surface topology representation in the form of a polygonal mesh or an algebraic surface representation, say, NURBS. Point based graphics therefore requires one to only register scans using point samples. Combining point samples from multiple scans is an operation that is trivial in comparison to merging polygonal meshes.

If an adequate number of features can be exactly matched between two scans, then the transformation needed to register one scan with the other can be easily determined. However, in general, scanner resolution, noise, etc. make it difficult to find adequate number of exactly matching features. Hence most scan registration techniques are iterative in nature, based on accurately aligning the overlapping geometry among the scans. They normally use a numerical optimization technique to minimize the geometric alignment error. As with most of the efficient numerical techniques, the registration techniques used are local minimization techniques. They require a good initial estimate of the registration transformation to converge rapidly to the final accurate transformation. Hence the registration process is considered as a two step process – an initial registration step that provides an estimate for the starting solution and an accurate registration that minimizes the alignment error iteratively (See Figure 1).

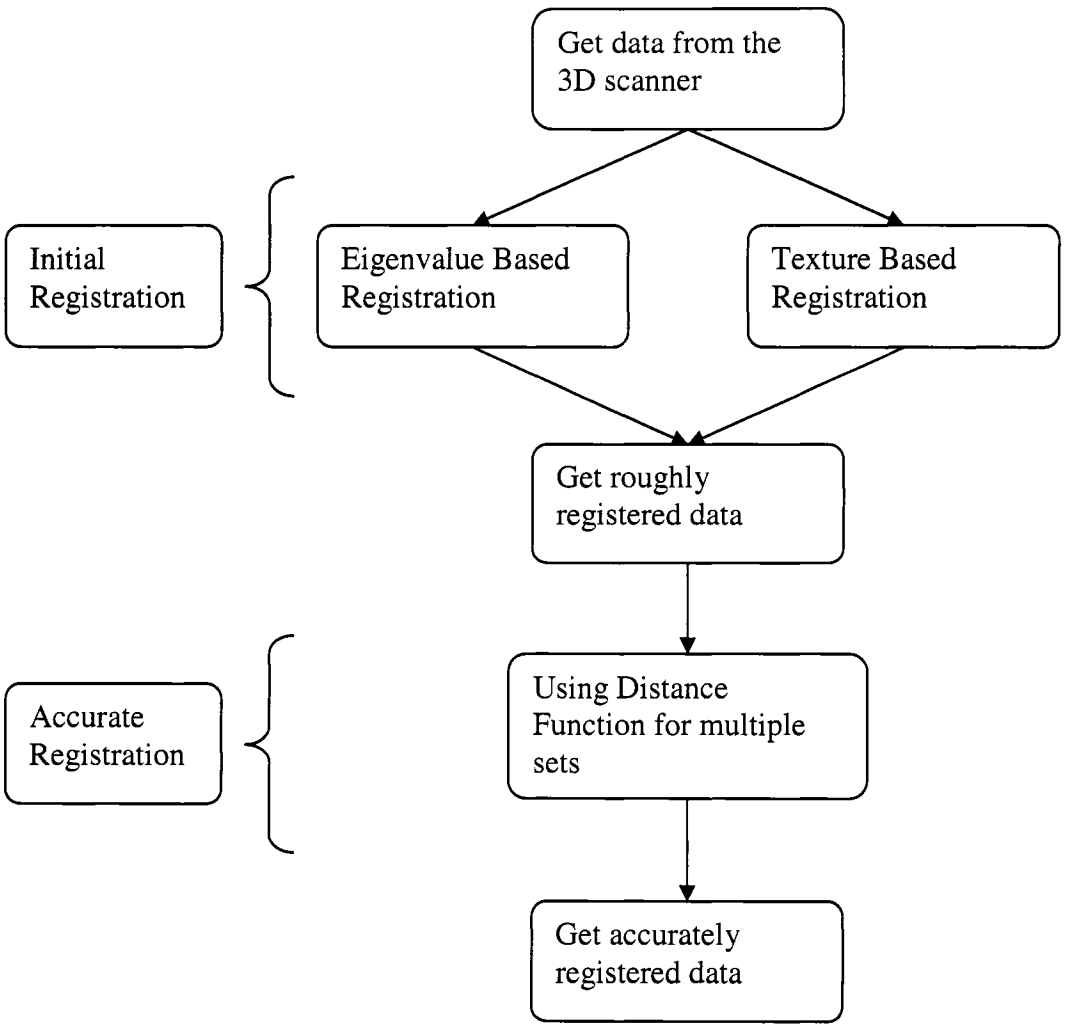


Figure 1: Graph of Registration Process

1.1 Initial Registration

Usually, the initial registration step brings the scanned data sets close to their final position/orientation in the global coordinate system. It can be thought of as a procedure for roughly registering the scanned data. Since we want to use iterative numerical optimization approach for doing the accurate registration, we need a good initial estimate for the alignment of the different scans. The usual approach for doing this initial

registration is to do it interactively. The scan geometries are shown on the screen and their positions and orientations are interactively manipulated to align them visually. While this is the standard supported in most commercially available 3D scanning software, a number of methods have been proposed to carry out this operation automatically. Usually this is done through some analysis of the geometry and/or other information available from the scan process. In our case, scan data not only contains geometric information, but it also comes with texture. Hence in our work, as discussed in all details in Chapter 3, we use both the texture information and geometric information for the initial registration.

1.2 Accurate Registration

It is clear that accurate registration of scan data is important for acquiring the correct geometry. This is particularly so for complex models, which usually require a larger number of scans for covering the complete surface of the object. Almost all the existing methods perform accurate registration in a pair-wise manner. That is, one scan data set is kept fixed and the 3D transformation needed to align the second scan data set to the first is determined as accurately as possible. Of course, there lies the underlying assumption that geometries of the two scans have considerable overlap between them. Multiple scan data sets are aligned as a chain of pair-wise registration and merge operations. In currently established approaches, pairs which have maximum overlap are chosen in sequence. The de facto standard algorithm for pair-wise registration is ICP (Iterative Common Point) [Chen 1991, Simon 1996]. ICP works on the principle of minimizing the

error formulation. The algorithm cycles between two steps, first finding correspondence between point sets and then applying the transformation to minimize the error. Computationally, establishing correspondence is a time consuming operation, as for every point of one set, it involves a search for the best corresponding point in the second set. Also, the number of iterations required for convergence of ICP and all its many derivative algorithms depends heavily on initial estimate and the distance metric used [Rusinkiewicz 2001].

In practice, depending on the field of view of the scanner camera head as well as the total number of scan views created to cover the complete surface of the object, it can be seen that there will be overlap not just between pairs of views, but also among larger subsets. However, in using a pair-wise registration technique, such as ICP, as the core alignment technique for registering multiple scans (more than 2), we can not easily make use of this additional overlap information present in the scan data. For this, ideally, we should be able to simultaneously align all the views that have common overlap among them in a single registration operation. The ICP error metric is essentially pair-wise, and extending it to accommodate multiple views is possible in principle, but not straightforward. To the best of our knowledge, there have been no attempts to develop an algorithm for simultaneously registering multiple (more than 2) overlapping scan data sets. Apart from the large number of pair-wise registration operations needed, the chained pair-wise registration process has another major problem – accumulation error. An iterative registration technique is terminated once the alignment error reaches below a pre-defined threshold or when a pre-set maximum number of iterations are completed. Hence there

will always be a residual alignment error between a pair of scans. As we register each new scan to the previously registered data set, this error can accumulate. It is our belief that new algorithms that can simultaneously register any number of overlapping scans need to be developed so as to reduce/eliminate the accumulated error for multiple scanned point sets registration. This has constituted our main research problem.

1.3 The Multiple 3D Scan data Registration Problem

In this work, we are concerned with directly registering and merging multiple 3D scan datasets without explicitly meshing them. Merging multiple aligned scan data sets represented by sampled surface points simply amounts to the union of the sample points in the different sets. Hence this will not be discussed further. As previously mentioned, one basic requirement for this process not to fail, is the presence of sufficient overlap amongst the scans that are considered for registration and merging. So our problem is that given multiple (overlapping) surface segments of an object's surface in their own coordinate space, we have to find the transformations needed to bring these segments into a common coordinate system. We assume that the required transformations are rigid body transformations. Most commonly encountered 3D registration problems fall in this category.

1.4 Significant Contributions

The main contributions of this research are the following:

1.4.1 Initial Registration

Eigenvector based Initial Registration using 3D Geometry Data

Eigenvalues and Eigenvectors describe the distribution of points. We build a low depth Octree for each scan data set, and calculate the Eigenvalues for each Octree node. Since Eigenvalues are independent of direction and position of the point set, we use it to find the corresponding parts (with similar point distribution) in the scanned point sets. We select the pair which has the best matched Eigenvalue as the corresponding part. When we find the corresponding parts, we also know the Eigenvectors for these parts. We can simply align these two sets of Eigenvectors to determine the transformation (rotation and translation) matrix. Since the method fails sometimes, we also developed a texture based initial registration method. Since this method was found to fail sometimes, we also developed a texture feature based registration method.

Texture Feature based Registration

In the 3D scanner set-up in our laboratory, texture accompanies with the scanned geometric data. We use digital image processing techniques to extract the scanned object's texture from the background and detect image-based feature point pairs for each scanned data pair. Selecting the best four matching feature pixel pairs, we look up the texture map for the matching 3D points. Using these four points, we can calculate the transformation matrix needed to align the two scans.

Since the initial registration is only a rough registration operation, we carry out the operation pair-wise to determine the starting transformation matrix for each of the multiple overlapping scans.

1.4.2 Accurate Registration

For the accurate registration, we have formulated and tested a new distance function based error metric for multi-scan registration. The distance function is defined within a 3D cube in which the object is embedded. We align the distance functions by choosing a subset of points close to the sample surface points (usually on a grid) in one of the data sets, and then transform these 3D space points for each of the other scans to determine the distance from the surface in each of the other scans. Thus we completely avoid the need to search and establish point correspondence among scan data sets, a time consuming operation needed in ICP-like algorithms. We carry out the error minimization using Levenberg-Marquardt algorithm and demonstrate improvements both in efficiency and reduced accumulated error when simultaneously registering a large number of scans.

1.5 Outline of Thesis

The rest of the thesis is organized as follows:

In Chapter 2, we provide a review of related work in 3D scan data registration. In Chapter 3, we present two initial registration approaches along with implementation results. In Chapter 4, we present our most significant contribution, our new approach based on distance function alignment for simultaneous registration of multiple 3D scan data sets. In Chapter 5, we conclude our work by summarizing the advantages and the weakness of our approach, and also indicate potential for future work.

Chapter 2: A Review of Multi-View 3D Scan Data Registration Techniques

The multi-view registration problem essentially requires one to associate a rigid body transformation with each 3D scan data set, such that the multiple scan data points obtained from different views of the object align themselves accurately according to the surface of the scanned 3D object. Figure 2 illustrates this diagrammatically; each T_i transforms the 3D points obtained from scan view i into a single global coordinate system, denoted here as the world coordinate system. Figure 3 shows a real case of scans from different views.

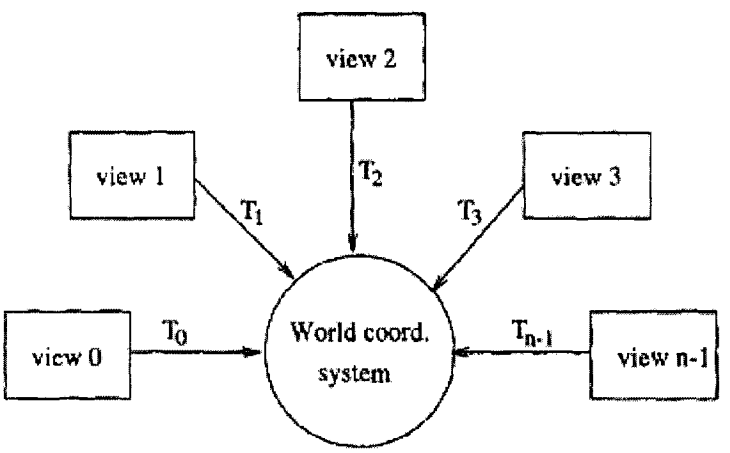


Figure 2: Registration of data from different views (Originally from [Neugebauer 1997])

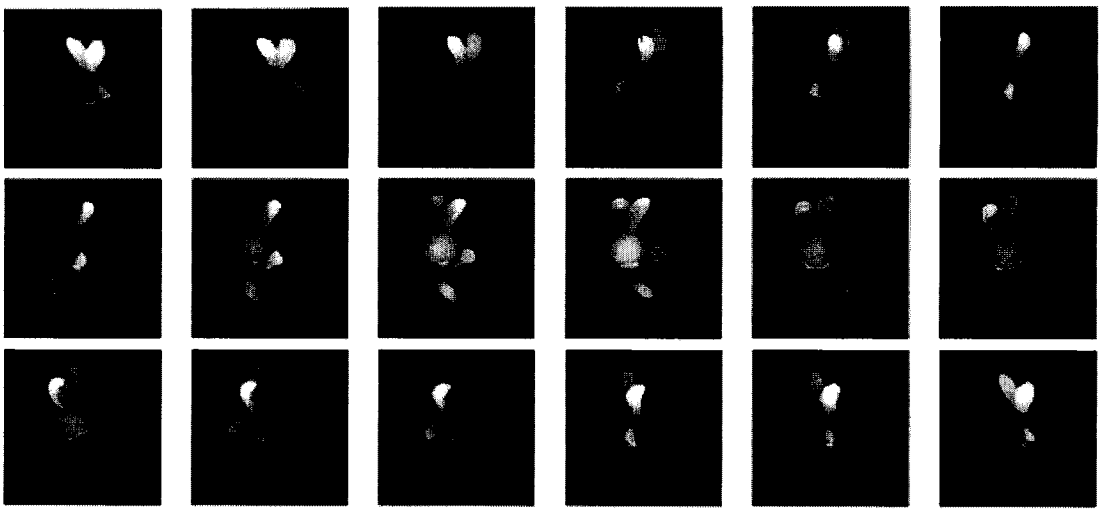


Figure 3: Multiple scans of a bunny statue (Originally from [Sharp 2004])

As mentioned in Chapter 1, the best known and most popular algorithm for the registration process is called ICP which stands for Iterative Closest Point algorithm. ICP was first introduced by [Besl 1992]. Most of the current registration approaches in use are derived from this method [Rusinkiewicz 2001]. ICP defines the registration problem as an iterative minimization process, by defining an error metric in the alignment of the overlapping geometric parts in the two scans. The basic idea behind this approach is to find the corresponding points (representing the overlapping subset of points) between two point data sets and then to minimize sum of the squared distance between those point pairs. This is explained in the flow chart shown in Figure 4.

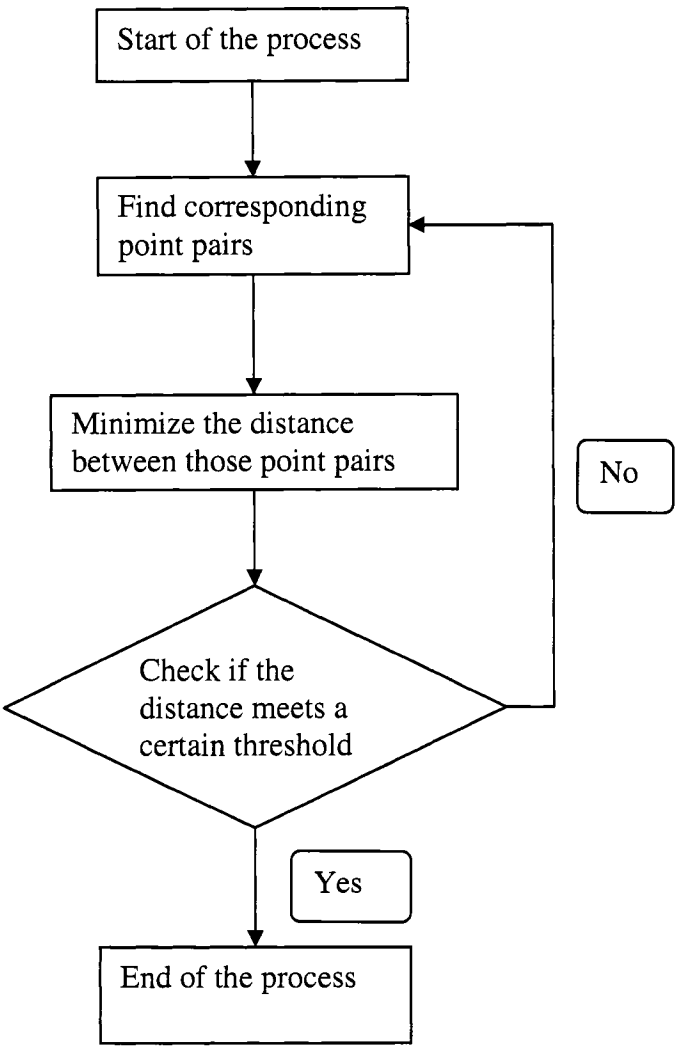


Figure 4: ICP Process

Since ICP uses at its core a numerical error minimization method to find the minimum of the distance-based error metric, it is unable to find the global minima. Hence, a good initial value for the set of alignment parameters is needed so that it can find the desired solution. Hence it is usual for the two data sets to be first roughly registered before starting the ICP technique. This requirement breaks the registration process into two

Registration which is sometimes refereed to as being a global registration process and accurate registration which searches locally for an optimum registration.

We have already previously mentioned that ICP has essentially been formulated for pair-wise registration, and use of it in a chain of pair-wise registration to handle multiple scan data sets could lead to accumulation error in the final merged geometry data set for a complex object. We implemented ICP and used a sequence of ICP registrations on a number of scan data for a number of objects. Accumulated error can be clearly seen in Figure 5. The scan data for this object consists of 16 scans (see row for Frog in Table 1 in Chapter 1.)

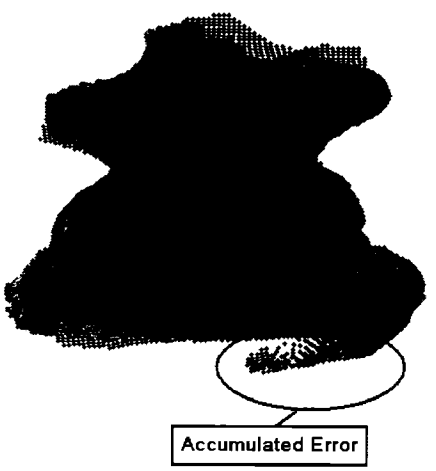


Figure 5: Accumulated Error

Simultaneous registration of multiple scans considering all overlaps during the process of minimization is one way of reducing/avoiding this accumulation error. For this the error metric has to be defined over all overlapping scan data sets. Since the ICP error metric is essentially pair-wise, extending it to accommodate multiple views is possible in principle, but not straightforward.

The ICP error metric is defined between 2 datasets as follows:

$$E = \sum_i (p_i - T_i^{-1} T_j p_{\alpha(i)})^2 \qquad (2.1)$$

where $p_{\alpha(i)}$ gives corresponding point in data set D_j for a point p_i in data set D_i, T_i^{-1} denotes the inverse of the current transformation for data set D_i and T_j denotes the current transformation for D_j. A good choice for corresponding point can be the nearest point in D_j.

To extend it so as to simultaneously register more than two scans, say, ***n*** overlapping views, we can formulate the error as the sum of ${}^{n}C_2$ pair-wise distance errors. However, this can quickly become inefficient. Also, our experiments have shown that much too often, it does not converge in a stable manner. What is needed is a different error metric formulation for simultaneous registration of multiple scans; one that is fast and converges to the minimum in a stable manner, which has formed the main goal of our research.

In the rest of this chapter we shall review related work on registering of 3D scan data sets, with particular emphasis of registering of multiple scans data sets.

2.1 Initial Registration

In the initial registration step, the primary assumption is that there is not even approximate knowledge of alignment between scan data sets. All the initial (global registration) techniques that are described in the literature are formulated for pair-wise registration and do not consider overlap among multiple (more than 2) scan data sets. This is not a major problem. Since the intent of the initial registration step is only to

provide a good starting estimate for the accurate registration step, the estimate provided by pair-wise registration is sufficient. There are three major approaches.

a) Interactive manual input: In some of the described approaches the goal is to make use of the scanner position information, assuming that it is available, and input that manually. For example, [Pulli 1999] simply track and record the movement of the scan head, and use that information to roughly align the data point set. However, in many low cost 3D scanning configurations, like desktop or hand-held scanners, there is very less likelihood of being able to track the scan head movement. Hence there is a need to develop other approaches for initial registration. It is clear that alignment of two scan data sets will depend on the detection of the overlapping components in the two scans. Since the overlapping information needed for estimating the initial alignment is not always easy to find automatically, interactive specification is a technique that has been widely used. Most systems require that the human operator visually identify the similarity of two data point sets, and then provide the initial registration transformation interactively. Many scanner packages provide good graphical interfaces for positioning and orienting scan data sets relative to each other. The problem with this approach is that it is not always obvious as to how closely should the human operator align the scans so that the subsequent accurate registration step works as desired.

b) Voting algorithms: This method of parameter estimation is also known as RANSAC-based method [Fischler 1981]. In [Chen 1998] a large number of disjoint subsets of point clouds are chosen from each scan data set and discrete transformations are found between corresponding subsets. The transformation which occurs the maximum number of times is chosen as the optimal transformation. The problem with

this type of method is that it needs to randomly try different combination of subsets. Hence it is not efficient and the processing time will be quite long.

c) Geometric Feature Matching algorithms: Algorithms like that of [Gelfand 2005, Chua 1997, Stein 1992] keep point signatures with each point. Chua et al compute features based on neighboring points [Chua 1997]. There is a compromise to be made on feature size. Very rich feature set takes too long to compute, while small feature size may not give an acceptable initial alignment. Spin images [Johnson 1997] are a typical example of feature rich matching where each point is augmented with an image representing the surface near it. One of the main draw backs of this approach is that it cannot handle the case that the scans do not have an adequate number of easily detectable matching geometric features. Especially in the case when noise is present, which is often the case in raw scanner data, it often does not always yield a usable initial registration.

For our work we have adopted the feature matching approach for initial registration. We have experimented with two kinds of features – geometric features in the form of Eigenvalues for subsets of scan data and color features derived from texture data.

2.2 Accurate Registration

In the Accurate Registration step, most of the approaches use different refinements of the basic ICP algorithm. All these variants of ICP differ mainly in their definition of the error measure and/or in the method used for correspondence establishment.

[Pulli 1999] uses pair-wise registration as in the classical ICP method. They extend the original ICP method in several areas. First, they add several constraints to the point pairs

selection strategy. For example: the difference of the normal of the corresponding points should be within 45 degrees; they also discard the points on the boundary to avoid wrong match; they also use dynamic threshold for measuring the closest distance between points. Second, they test two different alignment methods: Point to Point method and Point to Plane method. They find that in most cases for pair-wise registration, using the Point to Plane instead of using the Point to Point one can make the iteration faster, and also avoid the wrong corresponding pairs so that this method can improve the alignment result significantly.

[Turk 1994] uses a modified ICP which discards the point pairs if the points are too far from each other or the points are in the boundary. There are two main problems of their approach. First, they need the hardware to have the functionality to record the initial transformation information which is unavailable in most current 3D scanners. Second, they have to create the triangle mesh in order to accelerate the modified ICP algorithm.

In [Blais 1995], they use a similar approach as ICP which defines a cost function which is derived from reversing the rangefinder calibration process to describe the quality of registration. This function also uses the distance value to measure the quality. They also use a stochastic search to optimize the minimization of the cost function. This approach is stable and has an acceptable processing time, but as mentioned by the author, it needs a very long processing time in the Multi-view registration. Another problem with it is that it needs the positions of the scanner from different views to be recorded precisely in order to find the corresponding pairs. It is not practical in most real life scanning situations.

In [Chen 1998], the author proposes a method which is derived from the DARCES (data-aligned rigidity-constrained exhaustive search) method. Their method can register

two data point sets without any initial estimation. The special part in their approach is their method for selection of corresponding points. The point pairs are selected based on the geometric feature which is searched by using an improved exhaustive search at several different levels. This method is reliable but only works on the noiseless cases. Although it is much faster than the regular exhaustive search for very large data set, this algorithm will still take relatively long time.

In [Dorai 1997], they formulate a minimum variance estimator to compute the transformation factors which makes the registration calculation to be more accurate. They give a new error model by analyzing the dependencies between the orientation of a surface and the accuracy of surface normal estimation. As an improvement of the original ICP method, their method can handle the noise case very well, but the problem is that their algorithm is not suitable for registering multiple scanned data point sets simultaneously. [Dorai 1998] describes a prototype of multi scans registration. They implement the preprocessing for removing the noise according to certain standards. The main improvement in their work is that they design a point selection strategy which dynamically chooses corresponding point pairs through the iteration. The main disadvantage of their approach is that, as they mentioned in the paper, using the pair-wise ICP as the basic algorithm for registration will lead to an accumulated error for multiple scan registration.

[Holt 2001] extends the basic ICP by using color as the weight value to recognize the corresponding points. [Johnson 1996] goes further with the color ICP approach. They use the texture value in addition to using the geometrical information for selecting the corresponding pairs during the iteration. They also use a k-D tree data structure to make

the ICP more efficient. Since the texture of two different views may have different lighting conditions, the corresponding points may have different colors, which will make the final result inaccurate during registration. [Johnson 1997] generates a spin- image for each view by gathering the information from the whole surface. Then they use the image processing techniques to recognize the common features between views. This information is used for the Initial Registration. They also use the scene points which are generated from the first step as the corresponding points for implementing the ICP. Those scene points which describe the similar key features of two data point sets are good candidates of the corresponding points. Points around the scene points will be gradually selected during the iteration and will contribute to the registration.

[Weik 1997] develop an optical flow approach which makes use of the luminance information in order to find the corresponding point sets. [Chetverikov 2005] extends the original ICP by using Least Trimmed Squares approach. This approach increases the registration speed and robustness significantly. However, it still only works for registering two sets at a time. [Lomonosov 2004] extends their previous work Trimmed Iterative Closest Point algorithm (TriICP) by using a modified genetic algorithm to find the optimal transformation parameters. They also apply some data structure like k-D tree to speed up the process. But still, same as the previous one, it does not extend to Multi-view registration. In [Fitzgibbon 2001], the author uses the Levenberg-Marquardt method for optimizing the error metric which has a better performance than the original ICP method. [Simon 1996] demonstrates use of a constraint analysis method to select point sets wisely: only points around the feature are selected. It also shows a method to estimate the accuracy of registration by calculating the upper bound relation. [Mitra 2004]

demonstrates an optimization framework based on ICP which can handle the case that two sets are far away from each other. This overcomes the drawback of ICP which needs the two sets to be close enough. They use the *d2tree* (a variance of Octree, see Figure 6) to increase the processing speed. Furthermore, this optimization method converges much faster than the original ICP.

The above approaches are mainly focused on optimizing of registration for two data point sets with ICP technique at the core. But the special requirements of simultaneously registering multiple scans are rarely addressed well.

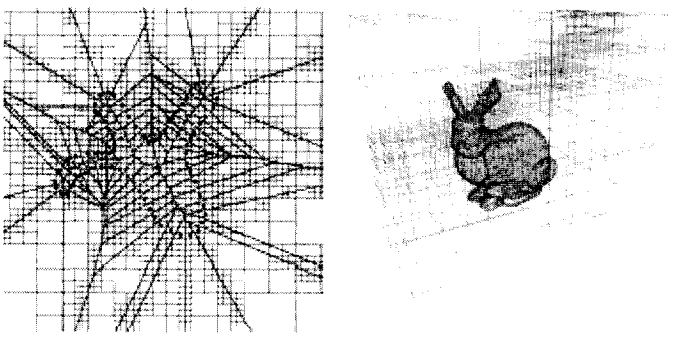


Figure 6: Example of using d2tree data structure to speed up ICP (Originally from [Mitra 2004])

2.3 Accurate Registration for multiple scanned data

There are several different approaches for multiple scanned data registration. Many of those methods use ICP in the inner loop, as the core alignment technique. That is, they perform multiple passes of ICP in a chained fashion by registering two 3D data sets at a time. Some of the methods use graph network to optimize the registration process. The problem with this type of approach is that the accumulated error will be introduced in the final registration. For example in [PENNEC 1996] the authors start with registering 2 scans and update a "mean shape". Then in each subsequent iteration, they register the

next scan against this "mean shape" and update the "mean shape" with this new dataset. [Neugebauer 1997] introduces a method for simultaneous registration. They define an error function and minimize the function by using numerical method with correction method. Due to the complexity of their error function, the processing time is relatively long. [Huber 2003]'s approach registers the data point sets pair-wise first. Then they use a graph structure to detect the globally incorrect, but locally consistent matches. (Figure 7) There are mainly two steps in their algorithm: In the first step, which they call it "Local Registration Phase", small amount of sets are registered, which creates a sub-graph. In the second step, which is called "Global Registration Phase", they connect all the subsets together by using a mix of discrete and continuous optimization algorithm. [Shih 2006] turns the registration problem into a quadratic programming problem of lie algebra parameters. They handle the accumulated error by distributing the error to the proper positions. They establish a graph representation to create a registration network in order to describe the registration problem. (Figure 8) Each node represents a data set and each edge represents a pair-wise registration. This graph structure is used for optimizing the registration process. In [Krishnan 2005]'s work, they present the global registration as an unconstrained optimization problem on a constrained manifold. They develop a method based on singular value decompositions which estimate the Initial Registration parameters. It works well in noise free cases. Then they use an iterative optimization method to minimize the local error on constrained manifold. [Sharp 2001] [Sharp 2004] first represent the global registration as a ring topology graph in frame space. Then, they decompose the graph to describe the neighboring view into several cycles and then calculate the optimal transformation parameters for each cycle. Then they use an iterative

method to integrate all the results got from the first set. In this way, they distribute the error to each cycle. The main contribution of this paper is that they develop a strategy to distribute the error thus ensuring that the accumulated error is not large for just the last scan of the registration process. In [Williams 2000], the author formulates the Multi-View registration into one constrained minimization where the translations are a function of optimal rotations. They encode the registration problem into a pre-computed matrix. Then they minimize the formula by using an iterative method.

All these methods are difficult to implement and do not make use of the information when more than 2 scans have a common overlap among them. The focus is more on how to optimally sequence the registration process rather than minimize the accumulated error simultaneously. Table 2 presents a comparison of the different features supported by these different methods.

Paper	Features	Remarks
[PENNEC 1996]	- Create Mean Shape	- Accumulated error is distributed
[Neugebauer 1997]	- Minimize the error function by using numerical method	- Long processing time
[Shih 2006]	- Uses graph to create a registration network	- Accumulated error is distributed
[Huber 2003]	- Pair-wise registration as the core algorithm - Create hierarchical graph to describe the registration problem	- Accumulated error is distributed
[Krishnan 2005]	- Manifold optimization approach	- Noise sensitive
[Sharp 2001] [Sharp 2004]	- Use ring topology graph	- Accumulated error is distributed
[Williams 2000]	- Encode the registration problem into a pre-computed matrix	- Need large amount of pre-computation

Table 2: Comparison of Main features of Different Multi-View Registration Techniques

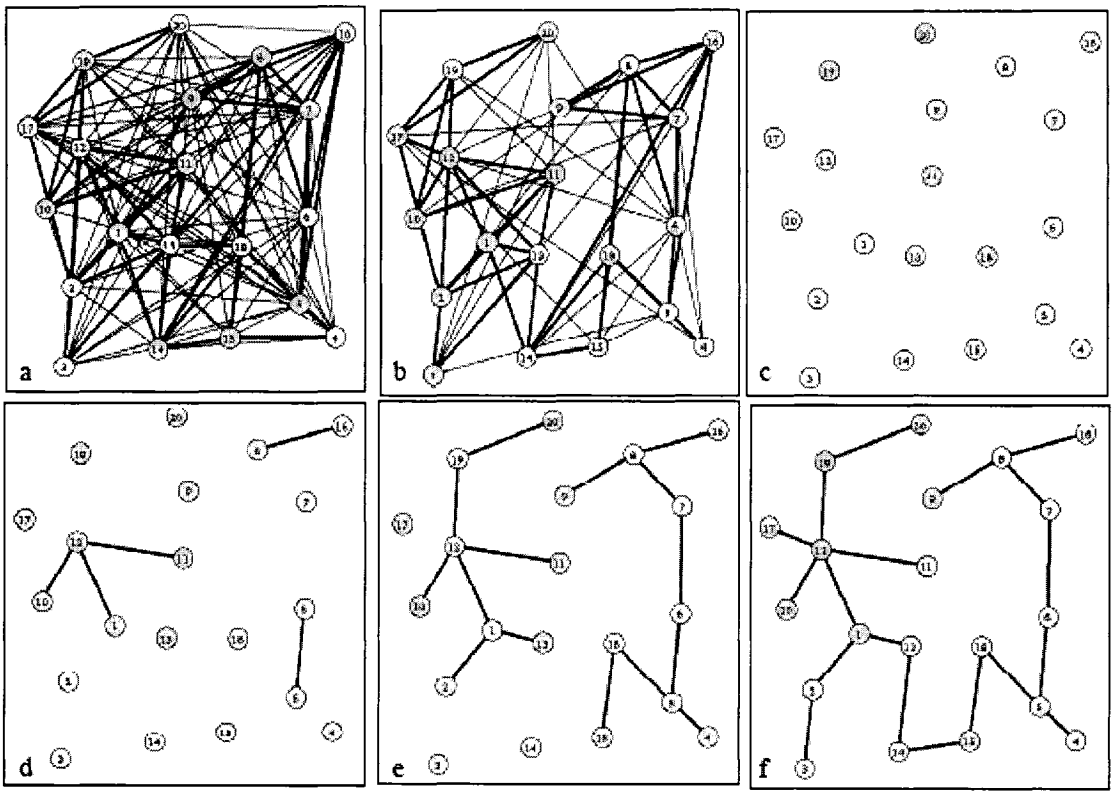


Figure 7: Registration Network Example (Originally from [Huber 2003])

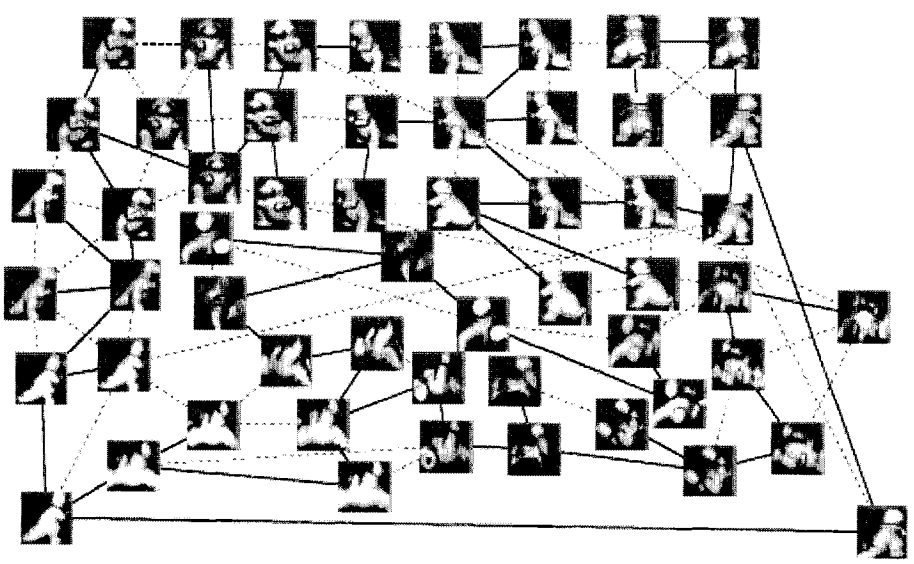


Figure 8: Registration Network Example (Originally from [Shih 2006])

2.4 Overlap in Multi-View Scans

Our research has been driven by the fact that all the above methods do not take full advantage of the fact that in the process of completely covering a complex object, very often various parts of the object will be captured by many more than 2 scans. Hence carrying out multi-view registration by considering more than 2 scans at a time is important and will help in increasing the accuracy. Detecting the subset of scans which have common overlap is something that can be done as part of the initial registration. For Accurate Registration, we have developed a new method which enables simultaneous registration of any number of overlapping scans by minimizing the variance in distance function values of a set of overlapping scans. This yields a fast and robust method of merging a large number of 3D scans into a single 3D model, firstly by eliminating the complicated point correspondence step, secondly by requiring fewer registration steps and lastly by using general-purpose nonlinear optimization (the Levenberg-Marquardt algorithm) for variance minimization, which make the registration converge in fewer iterations. Our approach is also independent of the sampling resolution and works well in the presence of noise. Implementation is also simple and straightforward.

Chapter 3: Initial registration

3.1 Our Development Environment

The environment we have set up for software development and experimentation includes a 3D scanning facility and the necessary program development framework.

The scanning facility consists of a 3D scanner (MegaCaptutor DK ©) from inSpeck Inc. with software driver for interactively cropping the 3D object data from the background. (See Figure 9) This scanner provides us 3D range data with associated color texture.

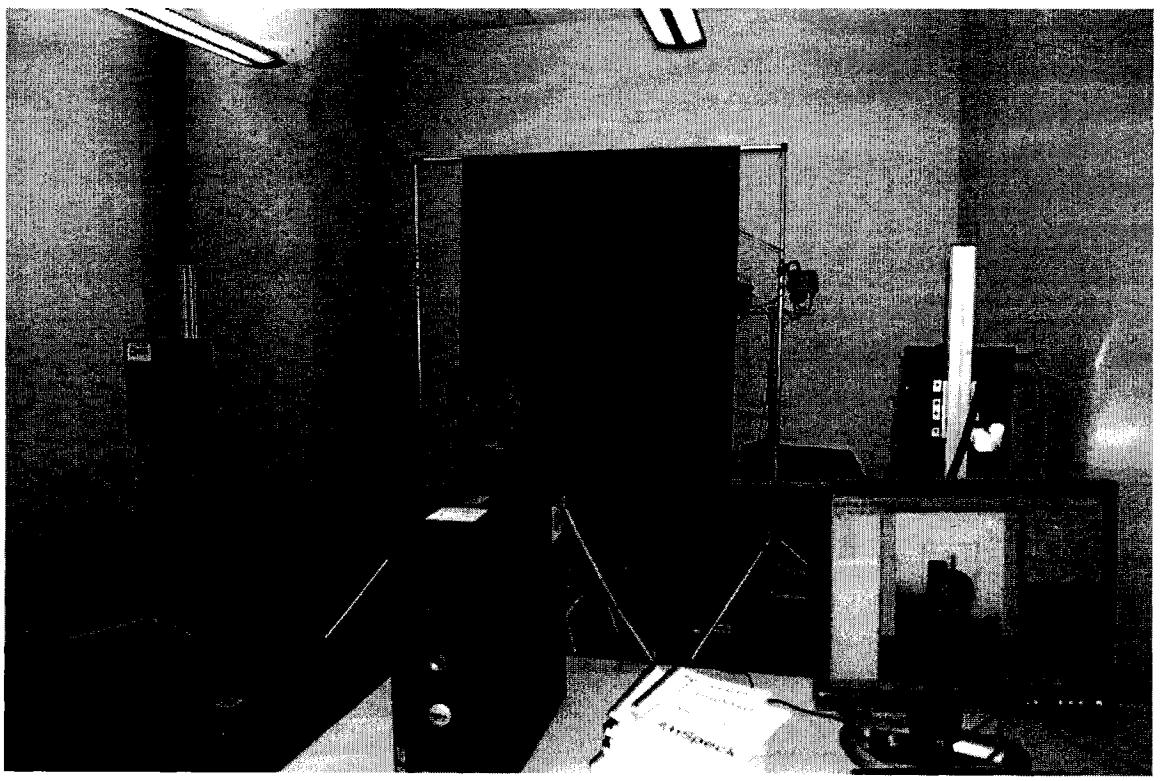


Figure 9: Our 3D Scanning Facility

For developing the software for the initial registration part, we have used C++ as the main programming language. Compared to Java and Matlab Script, C++ is more efficient. For extracting the model texture extract part, we have used Java. Since Java has some built-in classes for image processing and GUI interface design, it is convenient for the user to see the image processing result by using a graphic interface.

For the accurate registration part, we have used Matlab, because it has the required mathematical functions. While it is good for prototyping and testing, one disadvantage of using Matlab scripts is that even with code optimization, the speed of processing remains rather slow. Of course, the performance can be considerably improved by writing all the required mathematical functions in a procedural language such as C or C++.

3.2 Eigenvalue Based Registration

In order to get a good initial registration, we first need to find the overlapping parts among the scan datasets. The data in the overlapping parts would then be used to compute a 3D translation and a 3D rotation transformation. Since the data sets are captured through views of the object that could be freely rotated and translated with respect to each other, we use Eigenvalue analysis, which provides us with rotation independent features, to find the initial translation. Let us consider the case for two scanned 3D point sets. We shall call one of them as "Model Point Set", which remains unmoved (static). We call the other one as "Data Point Set". The basic idea is to use an octree like spatial data organization to divide the point sets into several groups in both "Model Point Set" and "Data Point Set". Each group is a node of the octree. For each node, if the number of points is larger than a prefixed threshold, we calculate the

Eigenvalue for this node. The procedure for deciding this threshold is discussed in the next subsection.

Eigenvalues and Eigenvectors are a kind of statistical measures on data. They describe the distribution of the points in the data set and it is independent of orientation and position. We match the Eigenvalues to find the corresponding nodes in two point sets. The best matching pair is then selected and their mean positions and Eigenvectors are used to compute the translation and rotation transformations respectively.

3.2.1 Octree Construction

3.2.1.1 Overview of Octree

The first step of Eigenvalue Based Registration is to build an Octree. We build the Octree using the following steps:

<1> compute an axis aligned bounding box for each data set.

<2> recursively divide the bounding box into 8 parts until a termination condition is met.

There are two ways of deciding on the termination condition.

1. By setting a threshold value for a minimum number of points within a node. This is set as a percentage of the total number in the data set.

Continue dividing the box until the number of points inside the node is less than the threshold, say 1% of the total number of points. (See Figure 10)

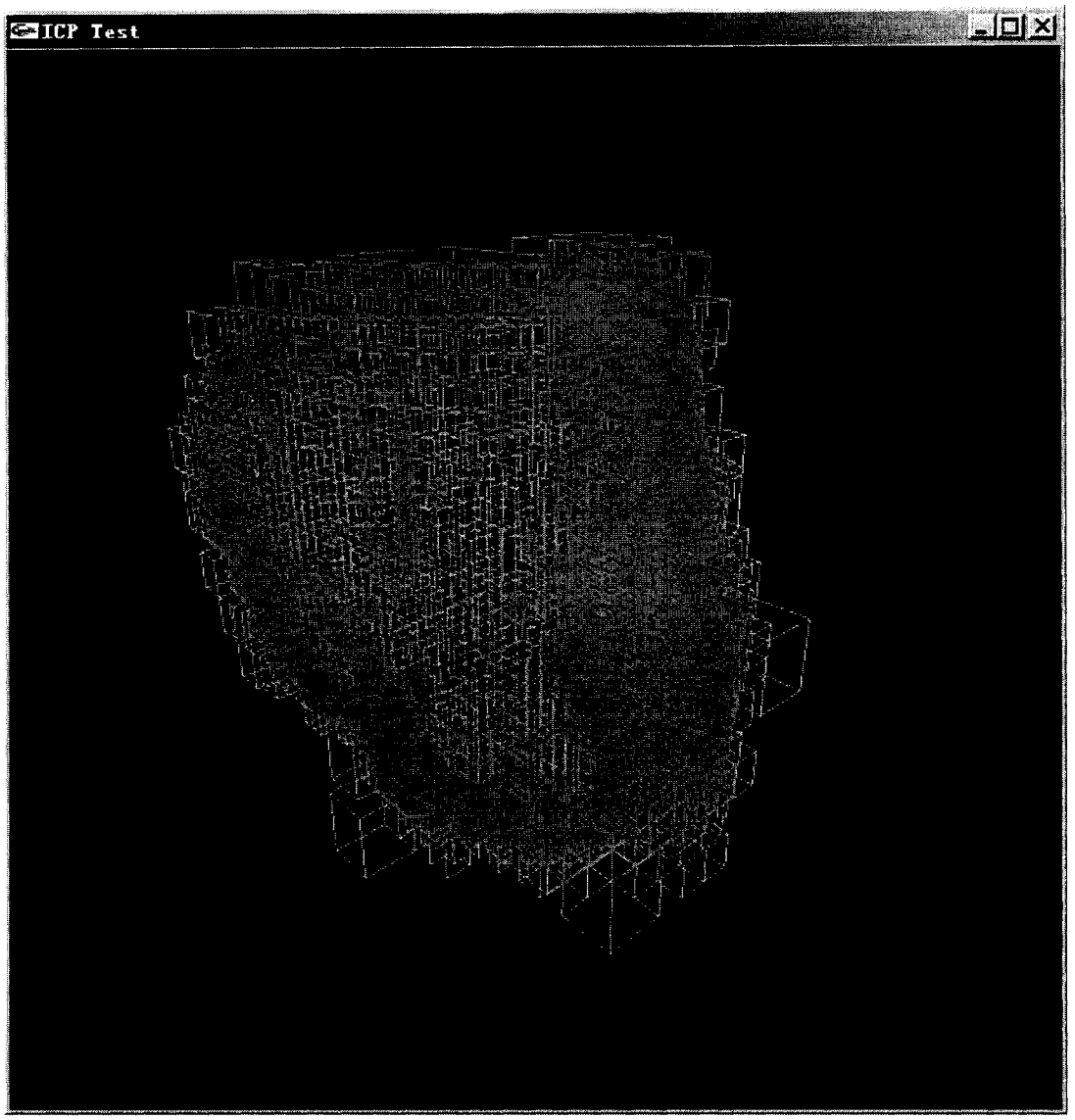


Figure 10: Octree with the threshold of 1%

2. By Level

Continue dividing until the leaf nodes of the octree are at given level, with the root considered as being at level 1. (See Figure 11)

One major advantage of the use of spatial data structure such as an octree is the fact that an octree structure provides an efficient way of determining spatial neighborhood

relationships among points. An often needed operation is to get immediate neighbors of a point or the group of points on a node. Hence being able to efficiently traverse through all neighbors of a cell is an advantage provided by an octree. It is important to note here that the neighbors of a cell may not all be at the same level as the node itself.

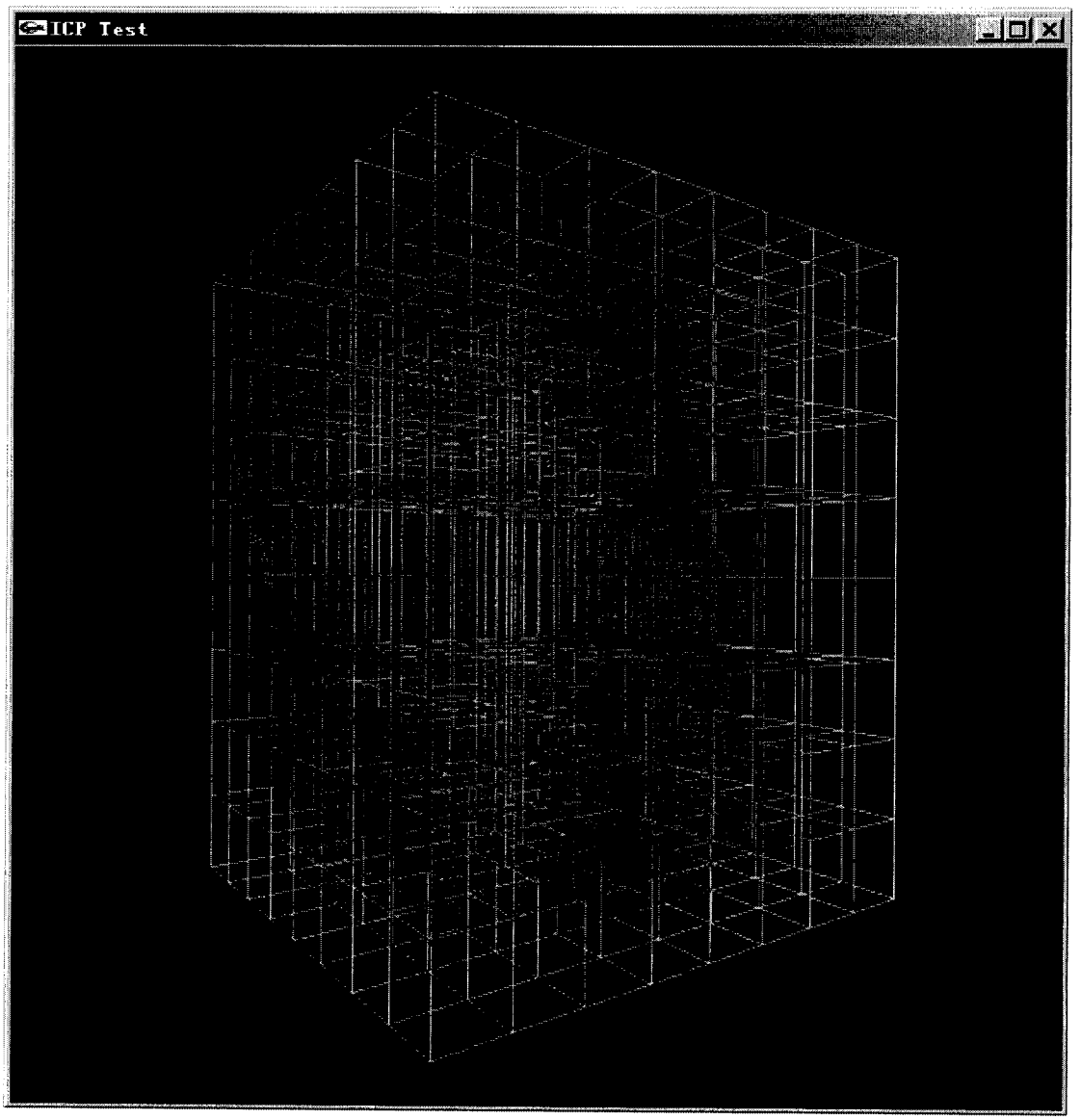


Figure 11: Octree with the threshold of Level 4

3.2.2 Registration by using Eigenvalue and Eigenvector

Since it is more difficult to describe the algorithm in 3D, we will discuss it in 2D first, and extend it to 3D later.

3.2.2.1. Finding Bounding Box

First, we need to build a proper bounding box for the two point sets (Data set and Model set). We choose the maximum length of the bounding box of each set as the length of our new bounding box. (See Figure 12)

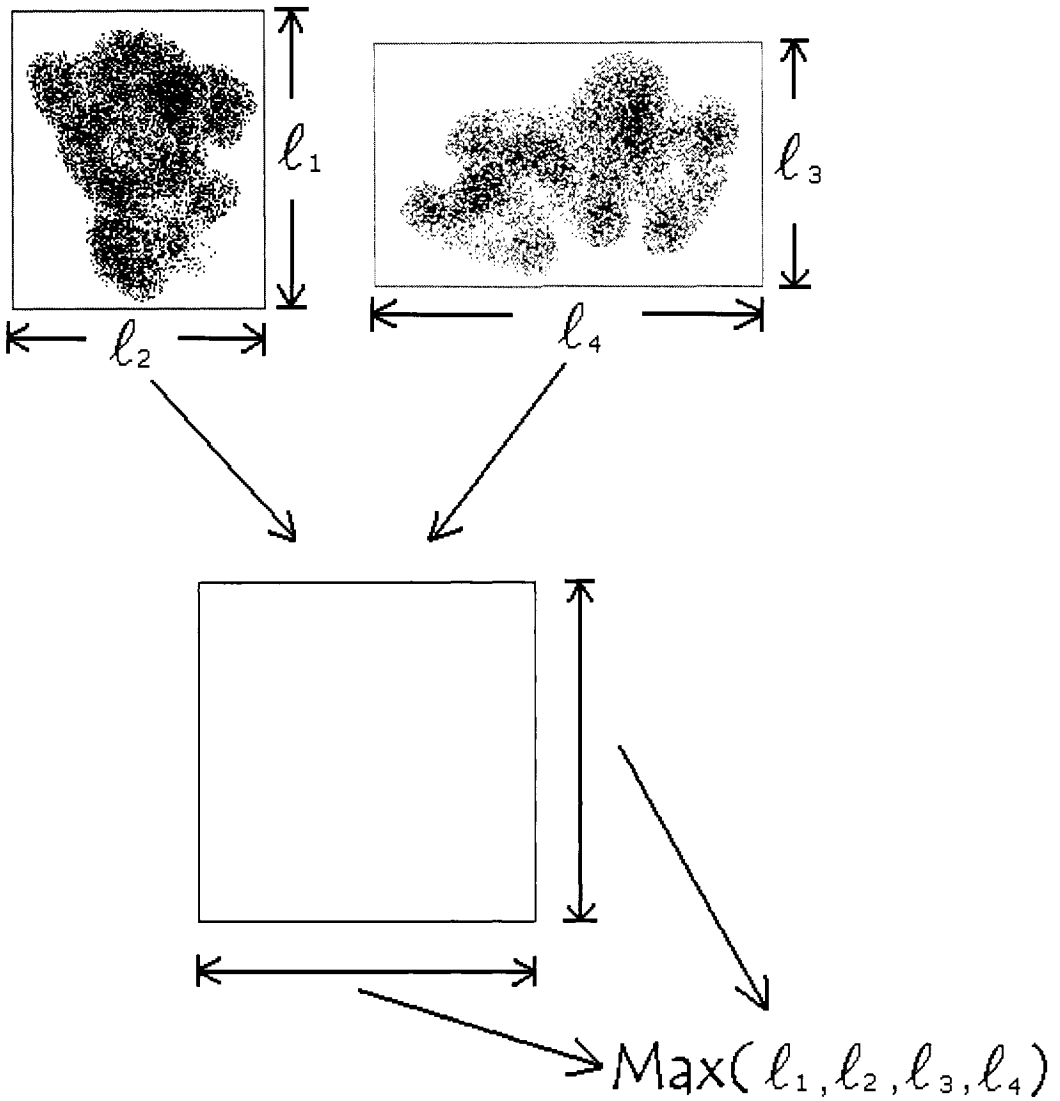


Figure 12: Creating bounding box for 2 scanned data point sets

3.2.2.2. Building Quadtree (Octree)

In the second step, we build a Quadtree (Octree). The level of the tree need not be very deep since we need adequate number of points in a node to determine spatial distribution information of a group of points using Eigenvalue Analysis.

3.2.2.3. Finding Matching Node Pairs

In order to find node pairs with matching Eigenvalues, we systematically move one quadtree (octree) with respect to the other with some nodes overlapping between the two. This is illustrated in Figure 13 - Figure 17.

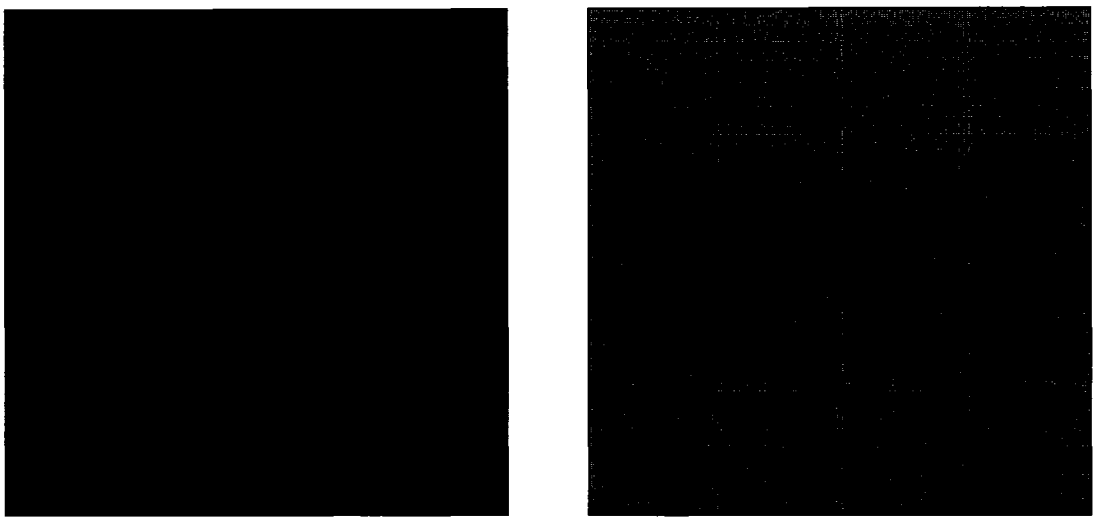


Figure 13: Red one is the Quadtree (Octree) of Data set, Green one is the Quadtree (Octree) of Model set. "*" is used to demonstrate the rotation. Yellow box represents the overlap part

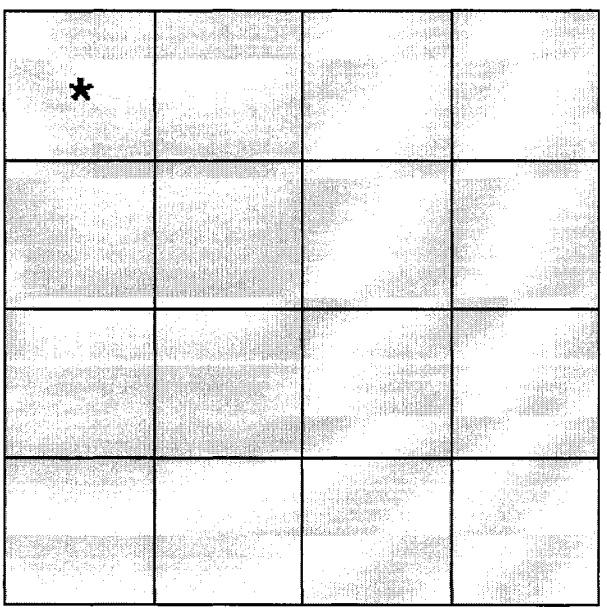


Figure 14: Two Octrees Fully Overlap

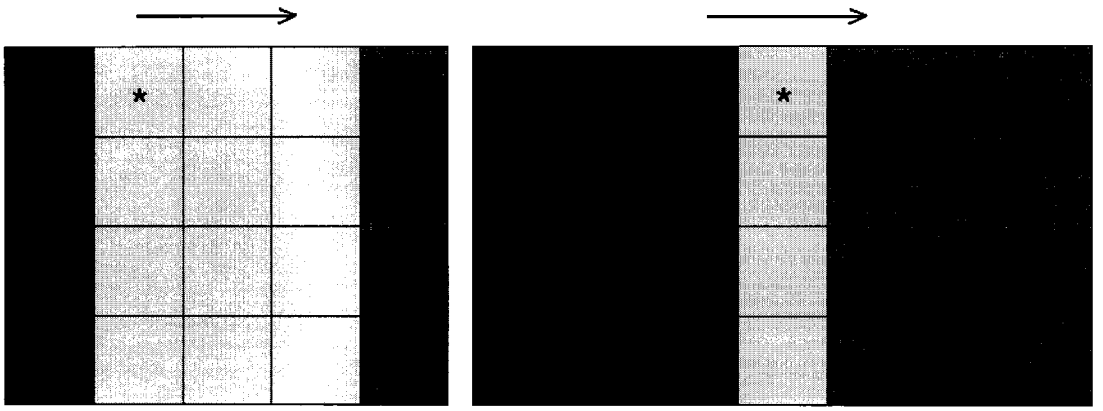


Figure 15: First Dimension Movement

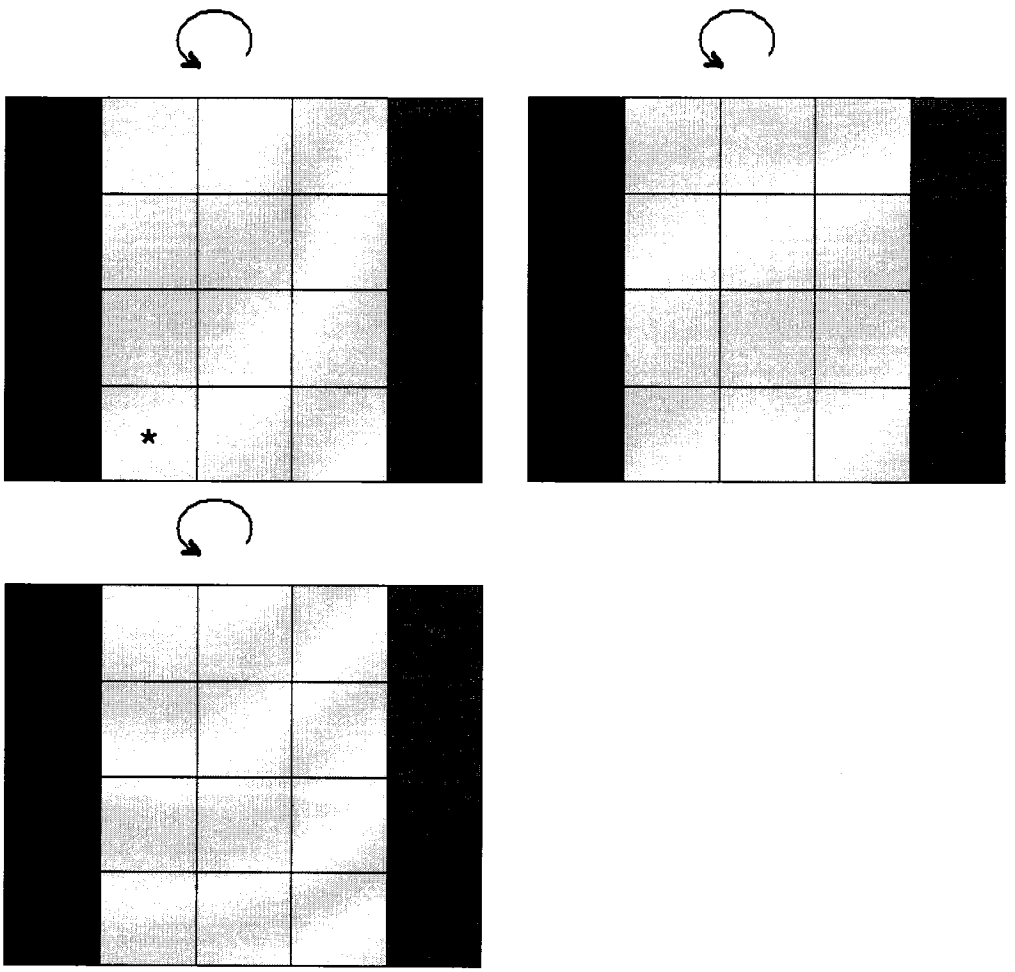


Figure 16: 2D-Rotation

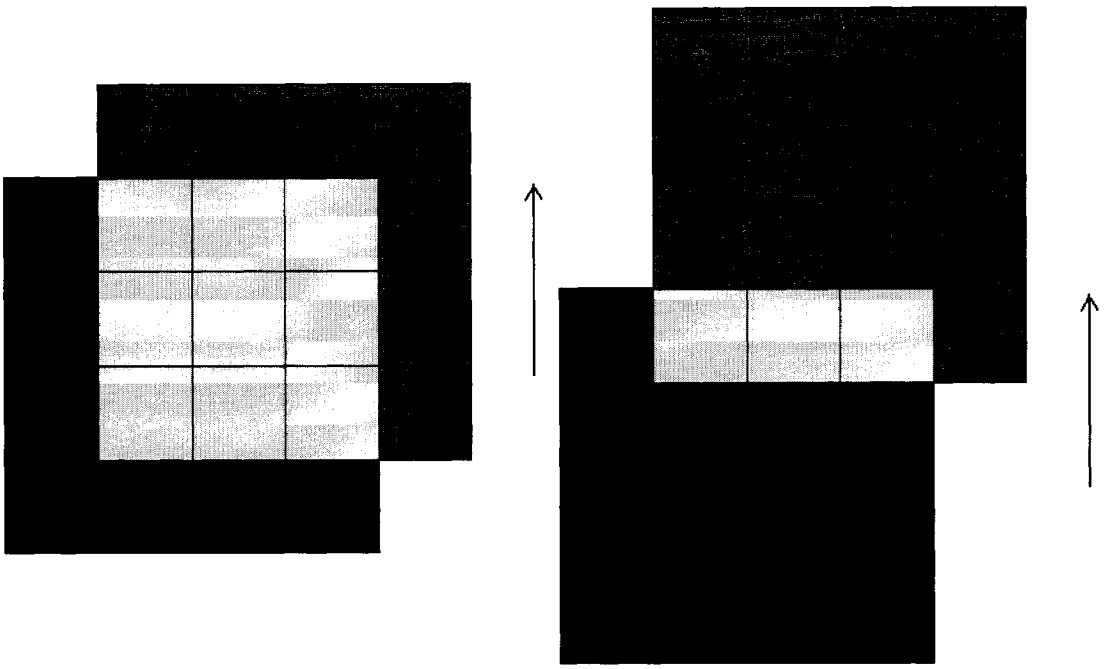


Figure 17: Second Dimension Movement

Through this process, we try all the possible combinations and find the best matched pair in terms of similar Eigenvalues.

3.2.2.4. Experimental Results

We have experimented with a large number of data sets. We have found that this method is not always successful in determining the Initial Registration with the required accuracy. We shall first demonstrate a case where the method is successful. Data of the two point sets are shown: (See Figure 18)

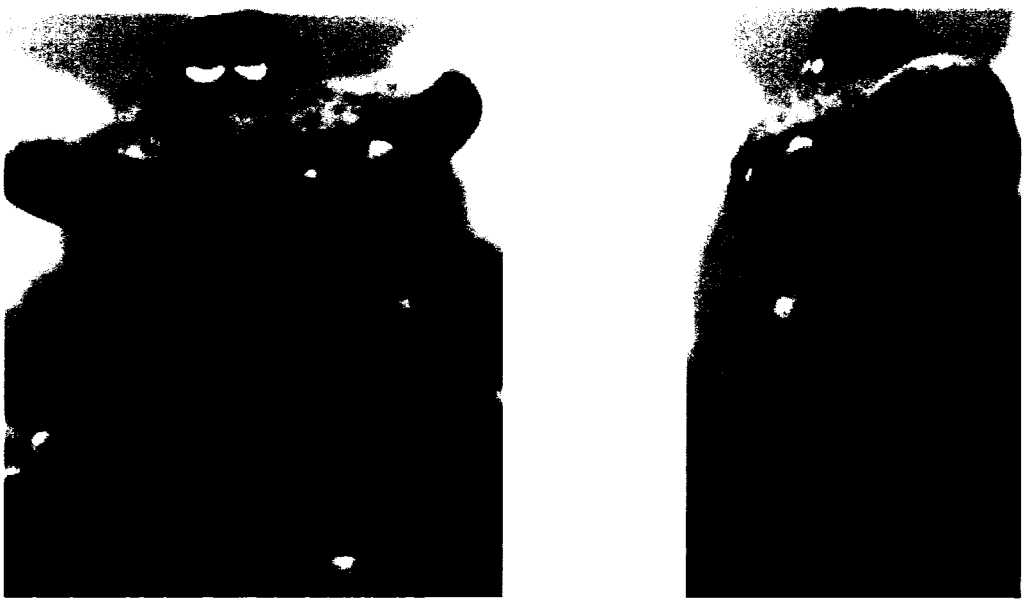


Figure 18: Two 3D Scan Views of an object

After Eigenvalue and Eigenvector calculation, the parts within the purple Octree nodes indicate the corresponding parts. (See Figure 19)

Figure 19: Node pairs which have similar Eigenvalue (purple part) with corresponding parts

Here is the corresponding part it finds. (Figure 20)

Figure 20: Corresponding part of the object

The second example shows a case where this method fails. (See Figure 21, Figure 22)

Figure 21: Node pairs which have similar Eigenvalue (purple part) but do not represent corresponding object parts

In the following we show other examples.

Figure 22: Two other 3D Scan Views of the object

Successful Case: (See Figure 23)

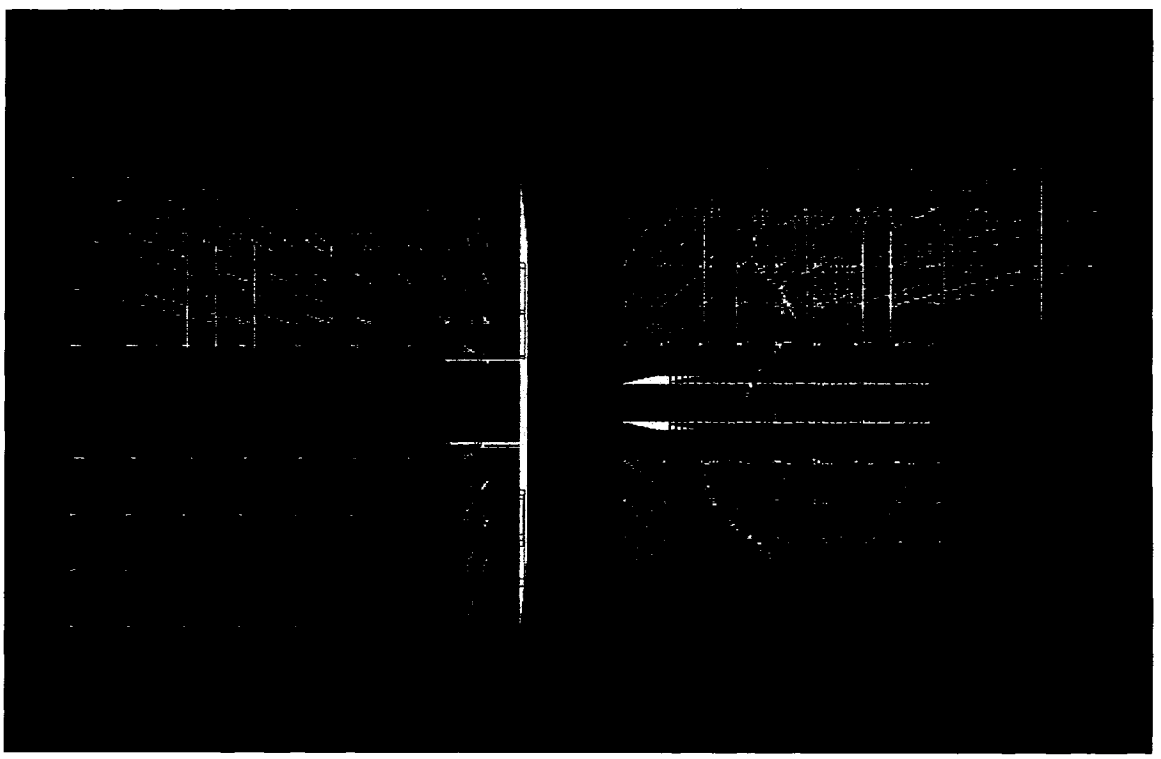


Figure 23: Node pairs which have similar Eigenvalue (purple part) with corresponding parts

Here is the corresponding part it finds. (See Figure 24)

Figure 24: Experiment2: Corresponding part of the object

Failed Case: (See Figure 25)

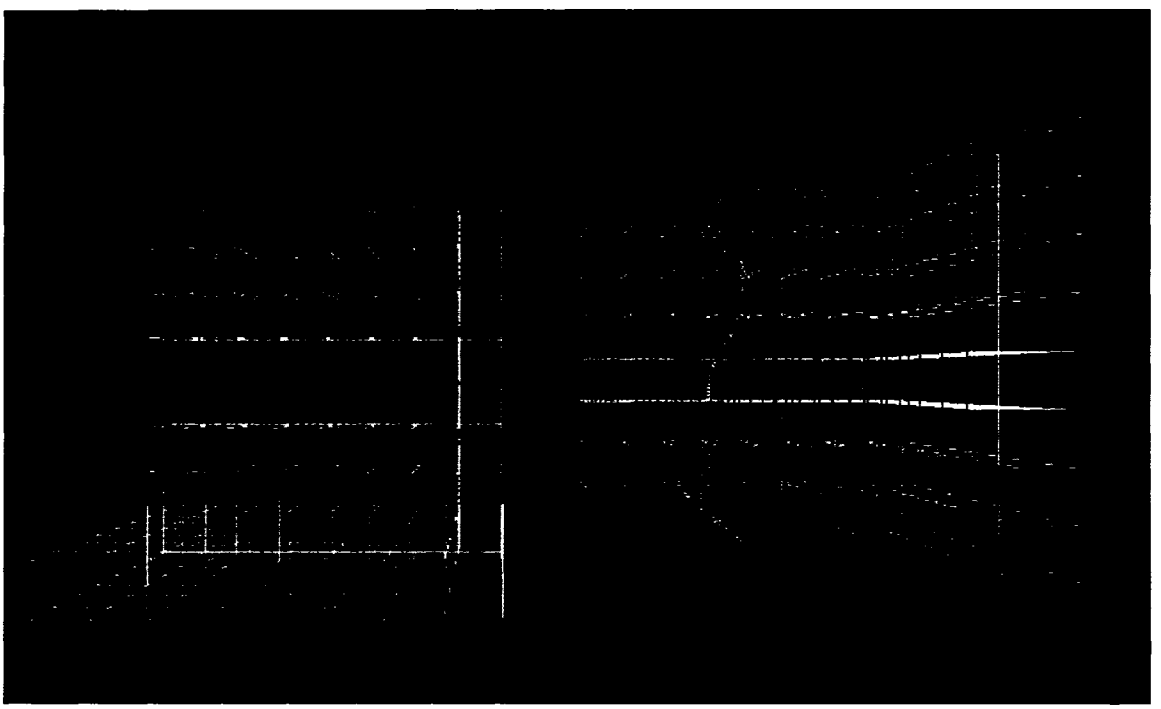


Figure 25: Node pair with similar Eigenvalue (purple part) but not corresponding parts

3.2.2.5 Conclusion

As shown in the above experiments, the partitioning by the Octree has to be proper, so that we can find the corresponding part by using Eigenvalue analysis. But if the point sets are not divided properly, the result will be wrong. Therefore, to improve the Initial Registration results, we have used Texture Feature based registration.

3.3 Texture Feature Based Registration

The first step for the Texture Feature Based Registration is to extract the model's texture from the background, a technique commonly referred to as foreground-background separation.

3.3.1 Extraction of Model Area from Background

There are two ways to extract the model area from the background. One is to let the user to select the area manually. This is what is supported by the manufacturer supplied software in our laboratory. Another way is to detect the area automatically using suitable image processing algorithms. The disadvantage for the second method is that it depends highly on the scanning set-up conditions. Lighting, background colors, model colors, etc. play very important roles in the success rate of these algorithms. In our work, we have used the following procedure.

First, we take a picture without model, we call it background image: (See Figure 26)

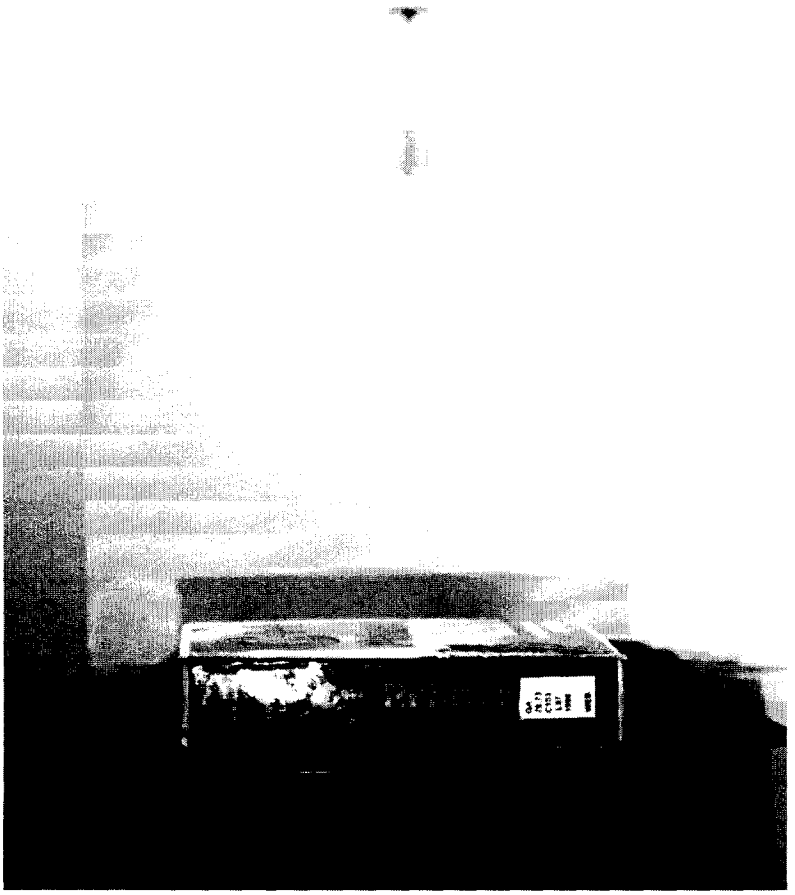


Figure 26: Background

And then, we take the picture with the model while we are doing the scan. We call it model image. (See Figure 27)

Figure 27: Background with scanned object

After that, we compare the two images. For each pixel in model image, if the difference in RGB value between the pixel in model image and the pixel in background image is within a certain threshold, it means that this pixel is in background. Otherwise, it is in the model.

If the lighting set-up of the 3D scan environment is not carefully set-up, shadows will always be a problem that has to be dealt with. Since the pixel within a shadowed region is

different from the corresponding pixel in the background image, it also has to be detected and should not be in model. (Figure 28)

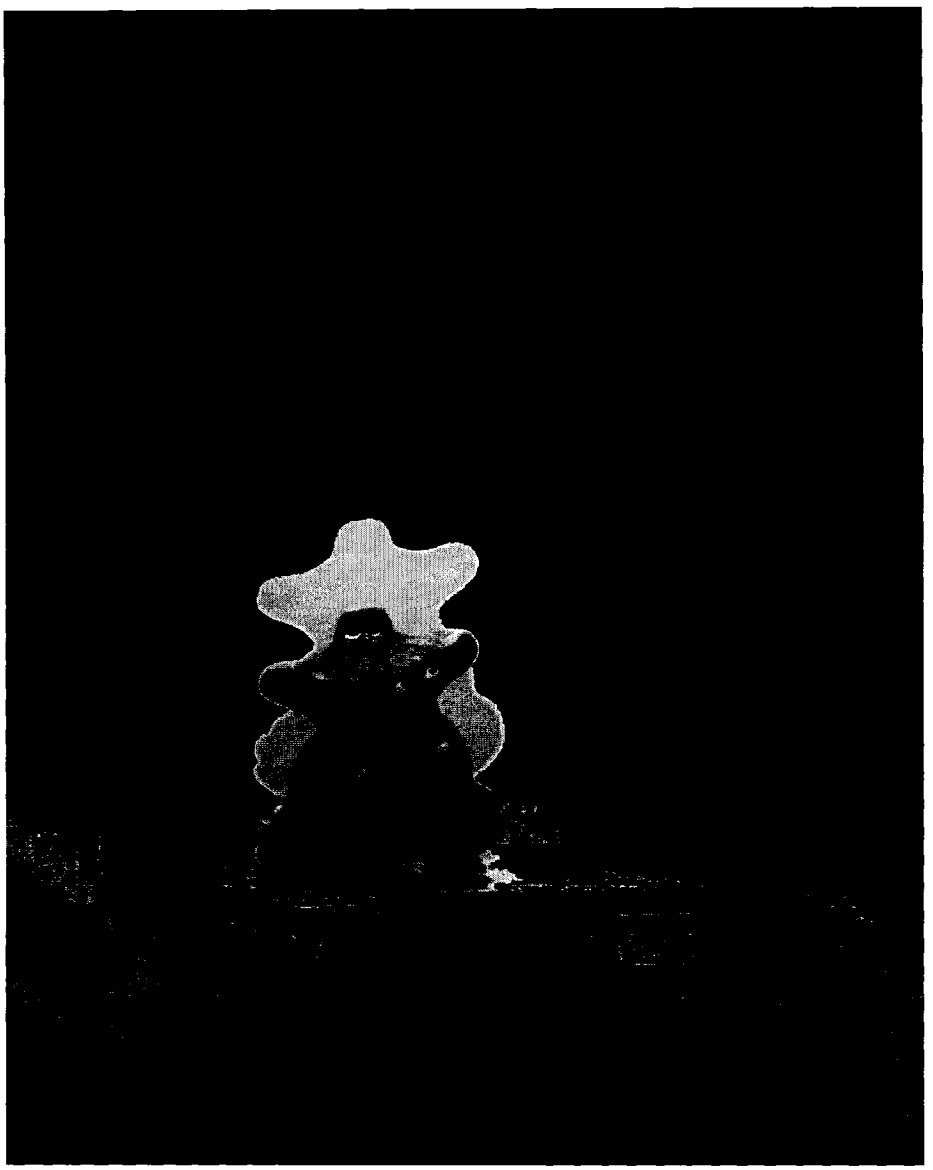


Figure 28: After removing background

We eliminate this problem by comparing V in the HSV (HSB) space for each pixel rather than comparing the RGB value.

The HSV (hue, saturation, value) system is also called the HSB (hue, saturation, and brightness) system. Whereas the RGB system is developed with hardware systems in mind, the HSB system is more visual perception oriented. It is a cylindrical coordinate system with hue, saturation, and brightness as the axes.

Hue represents the Color, the attribute that determines why we name a color in a particular way – say, red, yellow, or blue. Saturation tells how much white is added to monochromatic light. For instance, red is highly saturated, whereas pink is relatively unsaturated. Brightness describes how bright a color appears. For pixels inside shadow, their H, S value remains the same, but the brightness reduces.

Procedure for converting From RGB to HSV:

Let *max* equal the maximum of the (R, G, B) values, and *min* equal the minimum of those values. Following equation is used to convert RGB value into HSV space: (Equation 3.1)

$$h = \begin{cases} 0 & \text{if } \max = \min \\ 60° \times \frac{g-b}{\max - \min} + 0°, & \text{if } \max = r \text{ and } g \geq b \\ 60° \times \frac{g-b}{\max - \min} + 360°, & \text{if } \max = r \text{ and } g < b \\ 60° \times \frac{b-r}{\max - \min} + 120°, & \text{if } \max = g \\ 60° \times \frac{r-g}{\max - \min} + 240°, & \text{if } \max = b \end{cases} \tag{3.1}$$

$$s = \begin{cases} 0, & \text{if } \max = 0 \\ \frac{\max - \min}{\max} = 1 - \frac{\min}{\max}, & \text{otherwise} \end{cases}$$

$$v = \max$$

Then we compare all the pixels in the newly extracted part between the model image and the background image. If the differences of the corresponding pixels' H and S value in

two images are smaller than a certain threshold and the difference of their V value is larger than a certain value, it is in the shadow which should be the background.

Once we finish the comparing, we binarize the image by assigning the pixel within model 1, and the ones in background 0. (Figure 29)

Figure 29: After shadow removal and binarization

Then, we use a recursive function to find the largest connected region with 1s so that we can eliminate most of the noise. (See Figure 30)

Figure 30: After removing noise

Since the 3D point values at the boundary/silhouette of the object are less reliable (due to the surface being tangential to the scanner rays), the next step is to remove some of the boundary points in each scan from consideration during the registration process. For this, we use a 3 by 3 filter to first dilate (smoothen out the roughness in the boundary) and

erode the resulting image with one extra iteration. (See Figure 31) The resulting image represents the points interior to the scan view and of interest from this view.

Figure 31: After erosion and dilation

3.3.2 Registration by using Feature Points

Using an image processing based feature detection technique [David 2004] we find the feature points in each of the two scanned and processed texture images (See Figure 32 and Figure 33). Next, we select the best matching 3 feature points and use them for the initial registration.

The basic idea of this feature detection algorithm is to calculate the gradient for each pixel in the texture image, and to continuously scale down the resolution of the image, and calculate the gradient again. In this way, it sets a feature value for each pixel. Then it compares the feature values between two images. Finally, it will find the matching feature point. The disadvantage of this algorithm is, firstly, it is only an image based algorithm and works only when the image data is available for the 3D point set. Secondly, it cannot handle the case when the two images are very different. This means it can only be used to detect the matching feature points between two views that have reasonable overlap. Specifically for 3D surface data acquisition, this condition is not a problem.

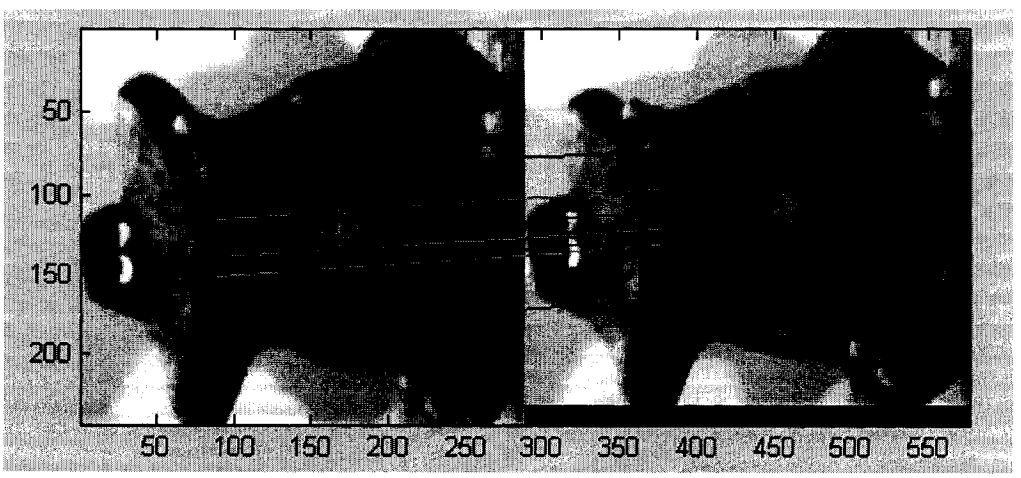


Figure 32: Feature Point Pairs 1

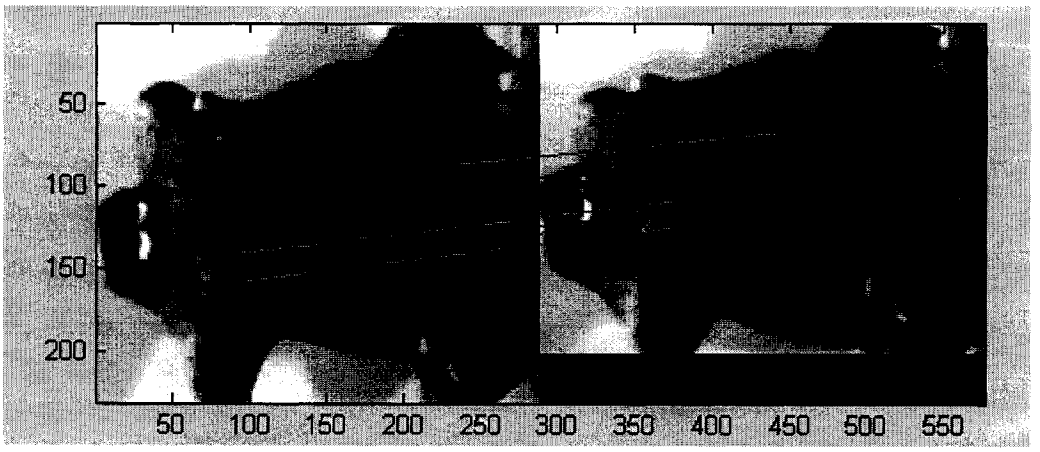


Figure 33: Feature Point Pairs 2

3.3.3 Calculation of Translation and Rotation

Once we have the three feature points, we can calculate the rotation and translation for the Initial registration.

For translation, we simply get the center of three feature points. Then we move the center of two scanned data point sets together. (See Figure 34)

We use quaternion to calculate rotation. (See Figure 35, Figure 36, Equations 3.2 - 3.14)

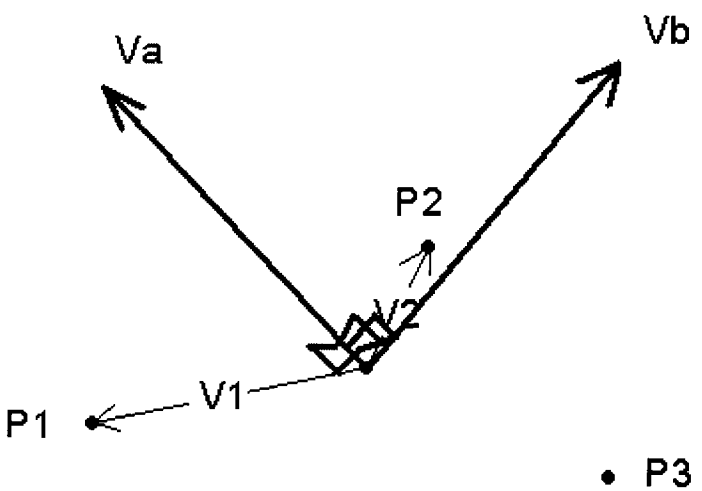


Figure 34: two vectors calculated from 3 points

$$< v, w >= \|v\| \cdot \|w\| \cdot \cos\theta \tag{3.2}$$

$$\|v \times w\| = \|v\| \cdot \|w\| \cdot |\sin \angle(v, w)| \tag{3.3}$$

$$Quaternion = [x, y, z, w] \tag{3.4}$$

All the vectors here are normalized.

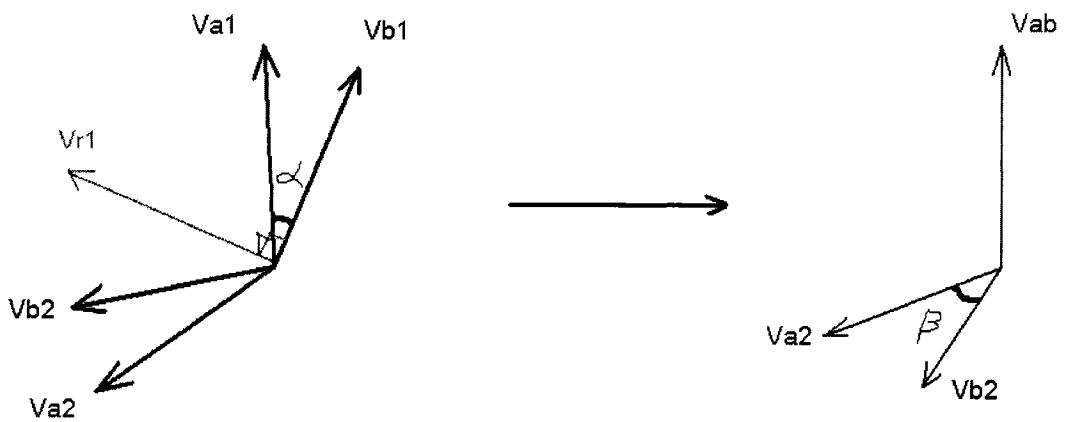


Figure 35: Use quaternion for rotation 1

$$\alpha = \arccos < Va1, Vb1 > \tag{3.5}$$

$$Vr1 = Va1 \times Vb1 \tag{3.6}$$

$$(\sin(\alpha/2))Vr1 = [x, y, z] \tag{3.7}$$

$$\cos(\alpha/2) = w \tag{3.8}$$

$$q = [\sin(\alpha/2) * x, \sin(\alpha/2) * y, \sin(\alpha/2) * z, \cos(\alpha/2)] \tag{3.9}$$

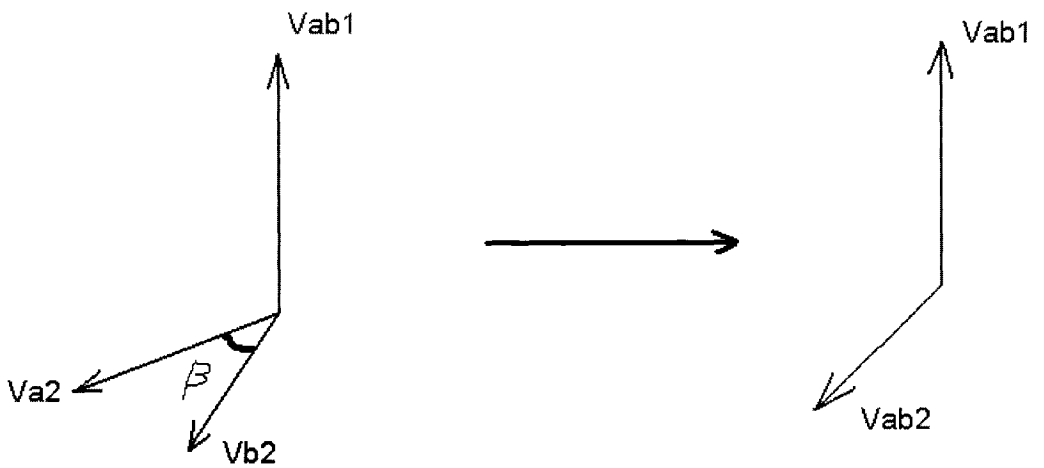


Figure 36: Use quaternion for rotation 2

$$\beta = \arccos < Va2, Vb2 > \tag{3.10}$$

$$Vab1 = Va2 \times Vb2 \tag{3.11}$$

$$(\sin(\beta/2))Vab1 = [x, y, z] \tag{3.12}$$

$$\cos(\beta/2) = w \tag{3.13}$$

$$q = [\sin(\beta/2) * x, \sin(\beta/2) * y, \sin(\beta/2) * z, \cos(\beta/2)] \tag{3.14}$$

(Only a unit quaternion encodes a rotation)

$$\begin{bmatrix} 1-2y^2-2z^2 & 2xy+2wz & 2xz-2wy \\ 2xy-2wz & 1-2x^2-2z^2 & 2yz+2wx \\ 2xz+2wy & 2yz-2wx & 1-2x^2-2y^2 \end{bmatrix}$$

3.3.4 Experimental Results

The following figures demonstrate the results of using the texture feature based initial registration approach as applied to scan data sets with 16 and 24 views: (See Figure 37, Figure 38)

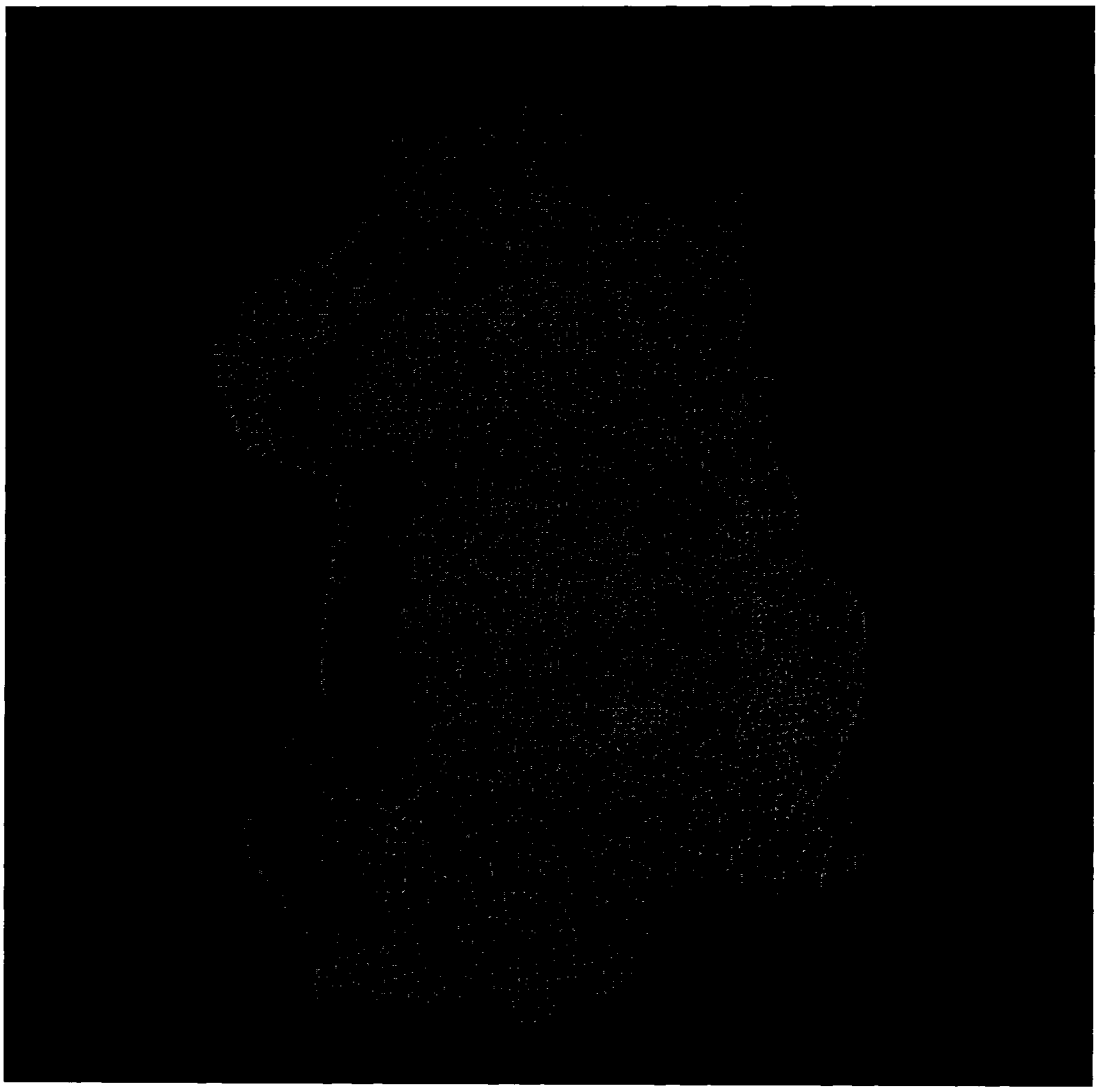


Figure 37: Initial Registration with 16 scanned data point sets (frog)

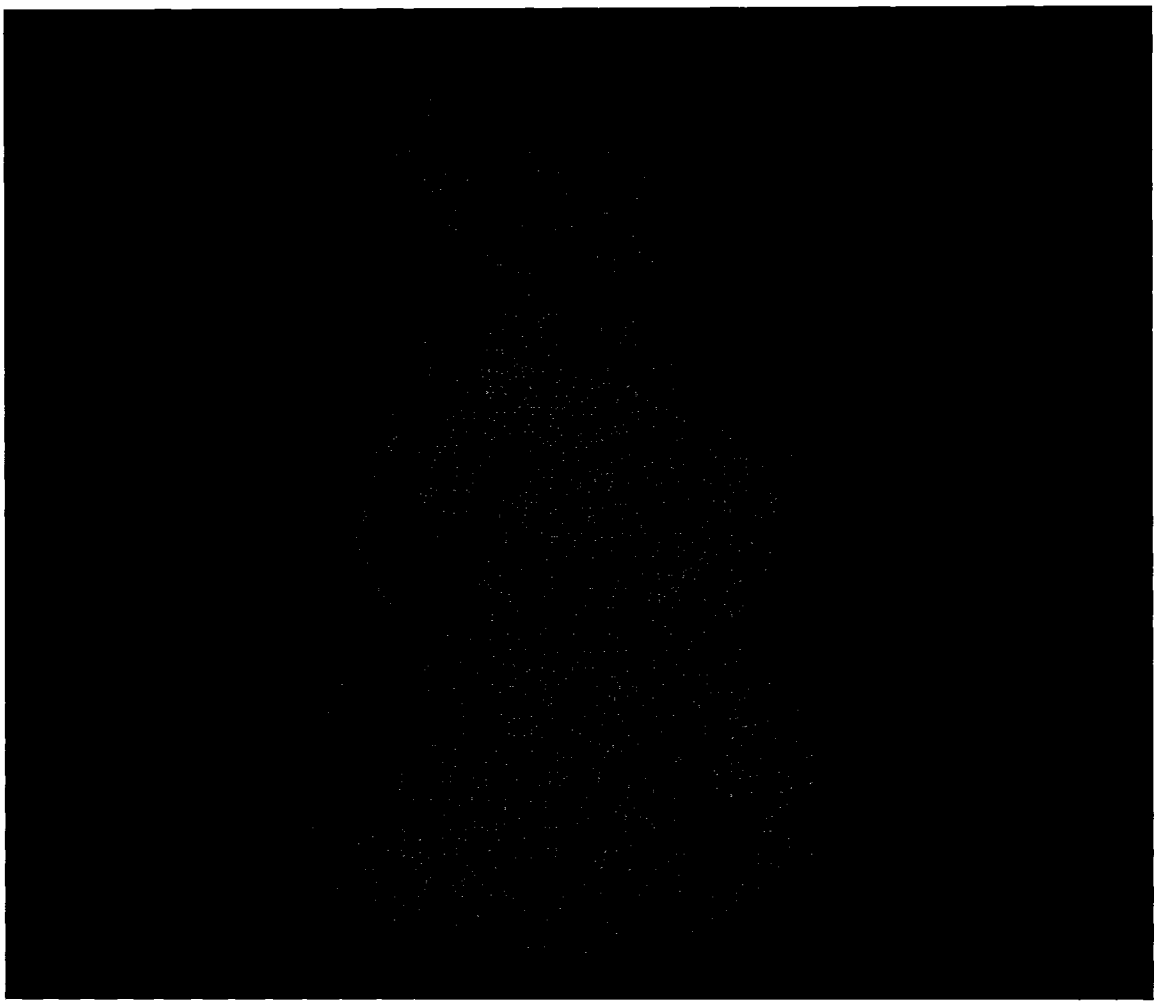


Figure 38: Initial Registration with 24 scanned data point sets (rabbit)

3.4 Conclusion

Eigen analysis based method is easy to implement, but it is not very stable in some cases. It depends strongly on the manner in which the initial bounding boxes are created. Simply using axis aligned bounding boxes does not yield a robust method. The texture based method is fast and more reliable, but it is sensitive to the lighting of the environment.

Chapter 4 Multi-view Registration

We introduce a new error metric based on distance field values which enables us to simultaneously register multiple overlapping 3D multi-view scan data sets. The first part of this chapter introduces the distance field based error for registering 2 data sets. The second part presents its extension to multi-view data sets.

4.1 Distance Function Error for Pair-wise Registration

The basic idea is to define a distance field around both Model point set and Data point set, and try to minimize the difference in field values between the spatially overlapping parts of the two fields. A discrete representation of the distance field is used in the form of distance field values defined on a 3D grid surrounding the point sets. Let us assume that there exist uniform 3D grids for both Model point set and Data point set within the bounding box enclosing each of the data sets. For every grid point, we find the nearest point in the point set, and record this distance as the field value.

Then, we apply the rotation and translation matrix to each Data grid point. To avoid distortion, we only use the upper left portion of the rotation plus translation vector. We have used this idea from an image processing algorithm [Richard 1994].

4.1.1 Distance Function

For accurate registration of point data sets, the essential and most important step is to correctly find the overlapping parts among the point sets. For this, we need to find and

match orientation independent features. First, we shall briefly try to understand how matching distance field values satisfy this requirement. Without loss of generality, let us consider only two scanned data point sets. Even if we store distance values, only at grid points, the distance field value at any other point inside the 3D bounding box can be suitably approximated using tri-linear linear interpolation. Hence every data set grid point also has an interpolated distance value in the model distance field. If the two scanned point sets are well registered, the distance values inside the "Data Grid Point Set" should be equal to the distance value inside the "Model Grid Point Set". Distance field values are clearly independent of orientation.

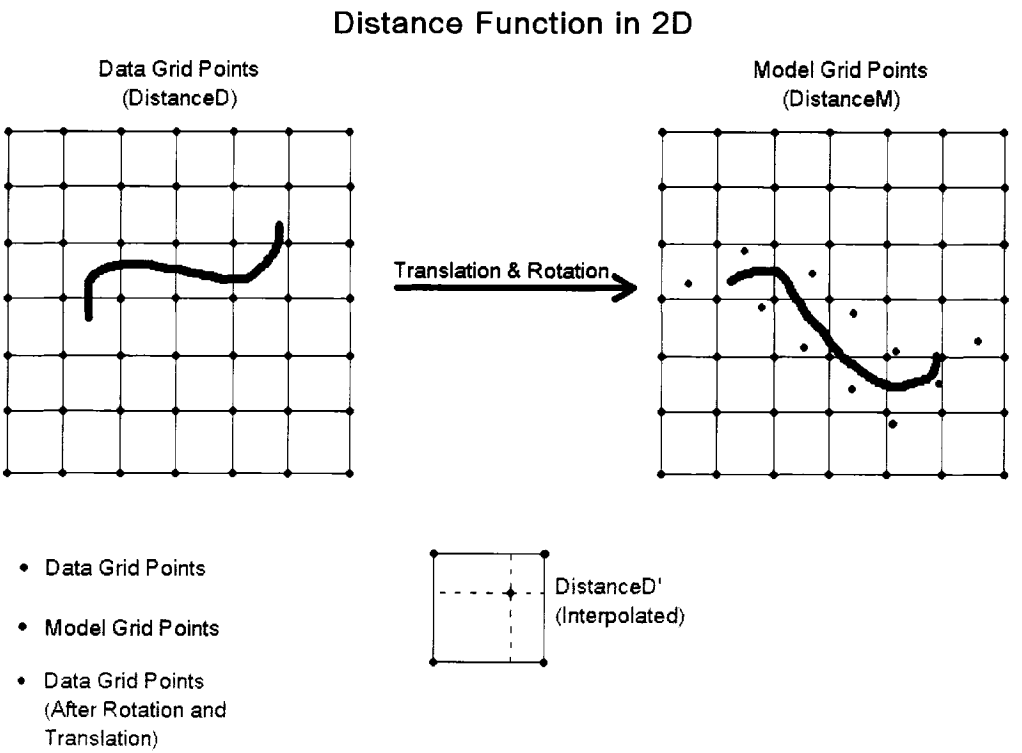


Figure 39: Moving from Data space to Model space

Data Grid point (after translation and rotation) in the model distance field. (See Figure 39)

Given $D(x', y', z')$, we try to find 8 valid bounding grid points surrounding this transformed point. (See Figure 40) Grid points are considered as valid, if they have been assigned a weight of 1. Typically, knowing that scanned points which are directly facing the scanner and more in the central region of a scan are more reliable, we assign weights of 0 to grid points lying closer to the boundary region of the scan. Similarly, we assign weight of 0 to grid points that are far from the surface; far is based on a user specified threshold distance.

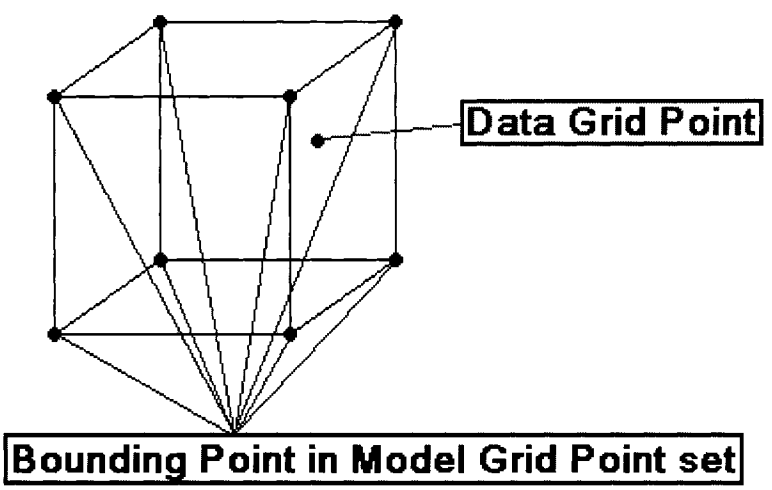


Figure 40: Grid point with its 8 bounding points in Model Space

If any of the 8 bounding grid points in the Model Grid Point Set are not valid (weight = 0), we simply discard the Data Grid Point. If they have valid Distance Value (weight = 1), we can get the Distance Value in model distance field by tri-linear interpolation using the distance field values at the 8 bounding grid points.

$$
\begin{aligned}
x_d &= x' - \lfloor x' \rfloor \\
y_d &= y' - \lfloor y' \rfloor \\
z_d &= z' - \lfloor z' \rfloor
\end{aligned}
\tag{4.1}
$$

$$
\begin{aligned}
i_1 &= D[\lfloor x' \rfloor, \lfloor y' \rfloor, \lfloor z' \rfloor] \times (1 - z_d) + D[\lfloor x' \rfloor, \lfloor y' \rfloor, \lceil z' \rceil] \times z_d \\
i_2 &= D[\lfloor x' \rfloor, \lceil y' \rceil, \lfloor z' \rfloor] \times (1 - z_d) + D[\lfloor x' \rfloor, \lceil y' \rceil, \lceil z' \rceil] \times z_d \\
j_1 &= D[\lceil x' \rceil, \lfloor y' \rfloor, \lfloor z' \rfloor] \times (1 - z_d) + D[\lceil x' \rceil, \lfloor y' \rfloor, \lceil z' \rceil] \times z_d \\
j_2 &= D[\lceil x' \rceil, \lceil y' \rceil, \lfloor z' \rfloor] \times (1 - z_d) + D[\lceil x' \rceil, \lceil y' \rceil, \lceil z' \rceil] \times z_d
\end{aligned}
$$

$$
\begin{aligned}
w_1 &= i_1(1 - y_d) + i_2 y_d \\
w_2 &= j_1(1 - y_d) + j_2 y_d
\end{aligned}
$$

$$
D'[x', y', z'] = w_1(1 - x_d) + w_2 x_d
$$

4.1.2 Iterative registration

We use Levenberg-Marquardt algorithm to minimize the sum of the squared differences in distance field values of a sampled set of data grid points. The difference is between the value in the data distance field and its value in the model distance field. The Levenberg-Marquardt algorithm is an iterative algorithm. It constantly updates the variable with a small value until the error reaches a value within a pre-specified threshold.

4.1.2.1 Rotation and Translation

Our alignment is based on three rotations and three translations.

Rotation and Translation:

$$Rx = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(A) & -\sin(A) \\ 0 & \sin(A) & \cos(A) \end{bmatrix}$$

$$Ry = \begin{bmatrix} \cos(B) & 0 & \sin(B) \\ 0 & 1 & 0 \\ -\sin(B) & 0 & \cos(B) \end{bmatrix}$$

$$Rz = \begin{bmatrix} \cos(C) & -\sin(C) & 0 \\ \sin(C) & \cos(C) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

$$T = \begin{bmatrix} Tx \\ Ty \\ Tz \end{bmatrix}$$

$$\begin{bmatrix} x_i' \\ y_i' \\ z_i' \end{bmatrix} = Rx * Ry * Rz * V + T$$

V is the vector notation for x, y, z.

A, B, C are the three rotation angles around X axis, Y axis and Z axis respectively.

Tx, Ty, Tz are the three translations along X axis, Y axis and Z axis respectively.

$$x_i' = \cos(B) * \cos(C) * x_i - \cos(B) * \sin(C) * y_i + \sin(B) * z_i + Tx$$

$$y_i' = (\sin(A) * \sin(B) * \cos(C) + \cos(A) * \sin(C)) * x_i$$
$$+ (-\sin(A) * \sin(B) * \sin(C) + \cos(A) * \cos(C)) * y_i - \sin(A) * \cos(B) * z_i + Ty$$

$$z_i' = (-\cos(A) * \sin(B) * \cos(C) + \sin(A) * \sin(C)) * x_i$$
$$+ (\cos(A) * \sin(B) * \sin(C) + \sin(A) * \cos(C)) * y_i + \cos(A) * \cos(B) * z_i + Tz$$

4.1.2.2 Error Function

We minimize the following error function:

$$E = \sum_i [D(x', y', z') - D(x, y, z)]^2 = \sum_i e_i^{\,2} \tag{4.3}$$

The next step for the algorithm is to calculate the weighted gradient vector and Hessian Matrix of e_i.

The weighted gradient vector is given by the equation:

$$b_i = -2 * e_i * Jacobian(e_i) = -2 * e_i * \begin{bmatrix} \frac{\partial e_i}{\partial A} \\ \frac{\partial e_i}{\partial B} \\ \frac{\partial e_i}{\partial C} \\ \frac{\partial e_i}{\partial Tx} \\ \frac{\partial e_i}{\partial Ty} \\ \frac{\partial e_i}{\partial Tz} \end{bmatrix} \tag{4.4}$$

The Hessian Matrix is defined by the equation:

$$Hessian(e_i) = \begin{bmatrix} \frac{\partial^2 e_i}{\partial A^2} & \frac{\partial^2 e_i}{\partial A \partial B} & \frac{\partial^2 e_i}{\partial A \partial C} & \frac{\partial^2 e_i}{\partial A \partial Tx} & \frac{\partial^2 e_i}{\partial A \partial Ty} & \frac{\partial^2 e_i}{\partial A \partial Tz} \\ \frac{\partial^2 e_i}{\partial B \partial A} & \frac{\partial^2 e_i}{\partial B^2} & \frac{\partial^2 e_i}{\partial B \partial C} & \frac{\partial^2 e_i}{\partial B \partial Tx} & \frac{\partial^2 e_i}{\partial B \partial Ty} & \frac{\partial^2 e_i}{\partial B \partial Tz} \\ \frac{\partial^2 e_i}{\partial C \partial A} & \frac{\partial^2 e_i}{\partial C \partial B} & \frac{\partial^2 e_i}{\partial C^2} & \frac{\partial^2 e_i}{\partial C \partial Tx} & \frac{\partial^2 e_i}{\partial C \partial Ty} & \frac{\partial^2 e_i}{\partial C \partial Tz} \\ \frac{\partial^2 e_i}{\partial Tx \partial A} & \frac{\partial^2 e_i}{\partial Tx \partial B} & \frac{\partial^2 e_i}{\partial Tx \partial C} & \frac{\partial^2 e_i}{\partial Tx^2} & \frac{\partial^2 e_i}{\partial Tx \partial Ty} & \frac{\partial^2 e_i}{\partial Tx \partial Tz} \\ \frac{\partial^2 e_i}{\partial Ty \partial A} & \frac{\partial^2 e_i}{\partial Ty \partial B} & \frac{\partial^2 e_i}{\partial Ty \partial C} & \frac{\partial^2 e_i}{\partial Ty \partial Tx} & \frac{\partial^2 e_i}{\partial Ty^2} & \frac{\partial^2 e_i}{\partial Ty Tz} \\ \frac{\partial^2 e_i}{\partial Tz \partial A} & \frac{\partial^2 e_i}{\partial Tz \partial B} & \frac{\partial^2 e_i}{\partial Tz \partial C} & \frac{\partial^2 e_i}{\partial Tz \partial Tx} & \frac{\partial^2 e_i}{\partial Tz \partial Ty} & \frac{\partial^2 e_i}{\partial Tz^2} \end{bmatrix} \quad (4.5)$$

For simplifying the computation, we only compute an approximate Hessian Matrix A:

$$a_i = \begin{bmatrix} \frac{\partial e_i}{\partial A}\cdot\frac{\partial e_i}{\partial A} & \frac{\partial e_i}{\partial A}\cdot\frac{\partial e_i}{\partial B} & \frac{\partial e_i}{\partial A}\cdot\frac{\partial e_i}{\partial C} & \frac{\partial e_i}{\partial A}\cdot\frac{\partial e_i}{\partial Tx} & \frac{\partial e_i}{\partial A}\cdot\frac{\partial e_i}{\partial Ty} & \frac{\partial e_i}{\partial A}\cdot\frac{\partial e_i}{\partial Ty} \\ \frac{\partial e_i}{\partial B}\cdot\frac{\partial e_i}{\partial A} & \frac{\partial e_i}{\partial B}\cdot\frac{\partial e_i}{\partial B} & \frac{\partial e_i}{\partial B}\cdot\frac{\partial e_i}{\partial C} & \frac{\partial e_i}{\partial B}\cdot\frac{\partial e_i}{\partial Tx} & \frac{\partial e_i}{\partial B}\cdot\frac{\partial e_i}{\partial Ty} & \frac{\partial e_i}{\partial B}\cdot\frac{\partial e_i}{\partial Ty} \\ \frac{\partial e_i}{\partial C}\cdot\frac{\partial e_i}{\partial A} & \frac{\partial e_i}{\partial C}\cdot\frac{\partial e_i}{\partial B} & \frac{\partial e_i}{\partial C}\cdot\frac{\partial e_i}{\partial C} & \frac{\partial e_i}{\partial C}\cdot\frac{\partial e_i}{\partial Tx} & \frac{\partial e_i}{\partial C}\cdot\frac{\partial e_i}{\partial Ty} & \frac{\partial e_i}{\partial C}\cdot\frac{\partial e_i}{\partial Ty} \\ \frac{\partial e_i}{\partial Tx}\cdot\frac{\partial e_i}{\partial A} & \frac{\partial e_i}{\partial Tx}\cdot\frac{\partial e_i}{\partial B} & \frac{\partial e_i}{\partial Tx}\cdot\frac{\partial e_i}{\partial C} & \frac{\partial e_i}{\partial Tx}\cdot\frac{\partial e_i}{\partial Tx} & \frac{\partial e_i}{\partial Tx}\cdot\frac{\partial e_i}{\partial Ty} & \frac{\partial e_i}{\partial Tx}\cdot\frac{\partial e_i}{\partial Ty} \\ \frac{\partial e_i}{\partial Ty}\cdot\frac{\partial e_i}{\partial A} & \frac{\partial e_i}{\partial Ty}\cdot\frac{\partial e_i}{\partial B} & \frac{\partial e_i}{\partial Ty}\cdot\frac{\partial e_i}{\partial C} & \frac{\partial e_i}{\partial Ty}\cdot\frac{\partial e_i}{\partial Tx} & \frac{\partial e_i}{\partial Ty}\cdot\frac{\partial e_i}{\partial Ty} & \frac{\partial e_i}{\partial Ty}\cdot\frac{\partial e_i}{\partial Ty} \\ \frac{\partial e_i}{\partial Tz}\cdot\frac{\partial e_i}{\partial A} & \frac{\partial e_i}{\partial Tz}\cdot\frac{\partial e_i}{\partial B} & \frac{\partial e_i}{\partial Tz}\cdot\frac{\partial e_i}{\partial C} & \frac{\partial e_i}{\partial Tz}\cdot\frac{\partial e_i}{\partial Tx} & \frac{\partial e_i}{\partial Tz}\cdot\frac{\partial e_i}{\partial Ty} & \frac{\partial e_i}{\partial Tz}\cdot\frac{\partial e_i}{\partial Ty} \end{bmatrix} \quad (4.6)$$

Since we cannot calculate the derivative directly, we use following function to calculate the derivative:

$$e_i = D(x_i', y_i', z_i') - Di$$

$$\frac{\partial D(x_i', y_i', z_i')}{\partial x_i'} = \frac{D(x_i' + \Delta x, y_i', z_i') - D(x_i' - \Delta x, y_i', z_i')}{2 \times \Delta x}$$
$$\frac{\partial D(x_i', y_i', z_i')}{\partial y_i'} = \frac{D(x_i', y' + \Delta y_i, z_i') - D(x_i', y_i' - \Delta y, z_i')}{2 \times \Delta y} \quad (4.7)$$
$$\frac{\partial D(x_i', y_i', z_i')}{\partial z_i'} = \frac{D(x_i', y_i', z_i' + \Delta z) - D(x_i', y_i', z_i' - \Delta z)}{2 \times \Delta z}$$

$$\frac{\partial e_i}{\partial A} = \frac{\partial x_i'}{\partial A} \cdot \frac{\partial D(x_i', y_i', z_i')}{\partial x_i'} + \frac{\partial y_i'}{\partial A} \cdot \frac{\partial D(x_i', y_i', z_i')}{\partial y_i'} + \frac{\partial z_i'}{\partial A} \cdot \frac{\partial D(x_i', y_i', z_i')}{\partial z_i'}$$
$$\frac{\partial e_i}{\partial B} = \frac{\partial x_i'}{\partial B} \cdot \frac{\partial D(x_i', y_i', z_i')}{\partial x_i'} + \frac{\partial y_i'}{\partial B} \cdot \frac{\partial D(x_i', y_i', z_i')}{\partial y_i'} + \frac{\partial z_i'}{\partial B} \cdot \frac{\partial D(x_i', y_i', z_i')}{\partial z_i'}$$
$$\frac{\partial e_i}{\partial C} = \frac{\partial x_i'}{\partial C} \cdot \frac{\partial D(x_i', y_i', z_i')}{\partial x_i'} + \frac{\partial y_i'}{\partial C} \cdot \frac{\partial D(x_i', y_i', z_i')}{\partial y_i'} + \frac{\partial z_i'}{\partial C} \cdot \frac{\partial D(x_i', y_i', z_i')}{\partial z_i'}$$
$$\frac{\partial e_i}{\partial Tx} = \frac{\partial x_i'}{\partial Tx} \cdot \frac{\partial D(x_i', y_i', z_i')}{\partial x_i'} + \frac{\partial y_i'}{\partial Tx} \cdot \frac{\partial D(x_i', y_i', z_i')}{\partial y_i'} + \frac{\partial z_i'}{\partial Tx} \cdot \frac{\partial D(x_i', y_i', z_i')}{\partial z_i'}$$
$$\frac{\partial e_i}{\partial Ty} = \frac{\partial x_i'}{\partial Ty} \cdot \frac{\partial D(x_i', y_i', z_i')}{\partial x_i'} + \frac{\partial y_i'}{\partial Ty} \cdot \frac{\partial D(x_i', y_i', z_i')}{\partial y_i'} + \frac{\partial z_i'}{\partial Ty} \cdot \frac{\partial D(x_i', y_i', z_i')}{\partial z_i'}$$
$$\frac{\partial e_i}{\partial Tz} = \frac{\partial x_i'}{\partial Tz} \cdot \frac{\partial D(x_i', y_i', z_i')}{\partial x_i'} + \frac{\partial y_i'}{\partial Tz} \cdot \frac{\partial D(x_i', y_i', z_i')}{\partial y_i'} + \frac{\partial z_i'}{\partial Tz} \cdot \frac{\partial D(x_i', y_i', z_i')}{\partial z_i'}$$

$$\frac{\partial x_i'}{\partial A} = 0$$
$$\frac{\partial y_i'}{\partial A} = (\cos(A) * \sin(B) * \cos(C) - \sin(A) * \sin(C)) * x_i$$
$$+ (-\cos(A) * \sin(B) * \sin(C) - \sin(A) * \cos(C)) * y_i - \cos(A) * \cos(B) * z_i$$
$$\frac{\partial z_i'}{\partial A} = (\sin(A) * \sin(B) * \cos(C) + \cos(A) * \sin(C)) * x_i$$
$$+ (-\sin(A) * \sin(B) * \sin(C) + \cos(A) * \cos(C)) * y_i - \sin(A) * \cos(B) * z_i$$

$$\frac{\partial x_i'}{\partial B} = -\sin(B) * \cos(C) * x_i + \sin(B) * \sin(C) * y_i + \cos(B) * z_i$$
$$\frac{\partial y_i'}{\partial B} = \sin(A) * \cos(B) * \cos(C) * x_i - \sin(A) * \cos(B) * \sin(C) * y_i + \sin(A) * \sin(B) * z_i$$
$$\frac{\partial z_i'}{\partial B} = -\cos(A) * \cos(B) * \cos(C) * x_i + \cos(A) * \cos(B) * \sin(C) * y_i - \cos(A) * \sin(B) * z_i$$

$$\frac{\partial x_i'}{\partial C} = \text{-cos(B)*sin(C)*}x_i \text{ - cos(B)*cos(C)*}y_i$$

$$\frac{\partial y_i'}{\partial C} = (\text{-sin(A)*sin(B)*sin(C)} + \text{cos(A)*cos(C)})*x_i + (\text{-sin(A)*sin(B)*cos(C)} - \text{cos(A)*sin(C)})*y_i$$

$$\frac{\partial z_i'}{\partial C} = (\text{cos(A)*sin(B)*sin(C)} + \text{sin(A)*cos(C)})*x_i + (\text{cos(A)*sin(B)*cos(C)} - \text{sin(A)*sin(C)})*y_i$$

$$\frac{\partial x_i'}{\partial Tx} = 1$$

$$\frac{\partial y_i'}{\partial Tx} = 0$$

$$\frac{\partial z_i'}{\partial Tx} = 0$$

$$\frac{\partial x_i'}{\partial Ty} = 0$$

$$\frac{\partial y_i'}{\partial Ty} = 1$$

$$\frac{\partial z_i'}{\partial Ty} = 0$$

$$\frac{\partial x_i'}{\partial Tz} = 0$$

$$\frac{\partial y_i'}{\partial Tz} = 0$$

$$\frac{\partial z_i'}{\partial Tz} = 1$$

For each Data Grid Point, we calculate a_i and b_i, and sum all of them together.

$$A = \sum a_i$$

$$b = \sum b_i$$

In each iteration, we calculate Δm by solving this equation:

$$\Delta m = (A + \lambda I)^{-1} b \qquad (4.8)$$

λ is a time-varying stabilization parameter [Press et al., 1992].

4.1.2.3 Implementation

In summary, the complete registration algorithm consists of following steps:

1. Start of iteration.

2. For each Data Grid Point at location, compute its corresponding position in the Model Grid space after translation and rotation.

3. Compute the interpolated Distance $d_i(x_i, y_i, z_i)$

4. Give an initial variable vector m and compute the error $E(m)$

5. Compute the Jacobian Vector b and Hessian Matrix A.

6. Compute the update value Δm

(a) Let "m" be the variable vector: $m = [\alpha, \beta, \gamma, t_x, t_y, t_z]$

(b) Pick a modest value for λ*, say* $\lambda = 0.001$

(c) Solve the system of equation $\Delta m = (A + \lambda I)^{-1} b$ *and update the motion estimate*

$m_{trial} = m + \Delta m$

(d) Apply $m^{(t+1)}$ *to equation* $E(m + \Delta m)$

If $E(m + \Delta m) \geq E(m)$*, increase* λ *by a factor of 10 (or any other substantial factor).*

If $E(m + \Delta m) < E(m)$*, decrease* λ *by a factor of 10, update the trial solution*

$m^{(t+1)} = m^{(t)} + \Delta m$

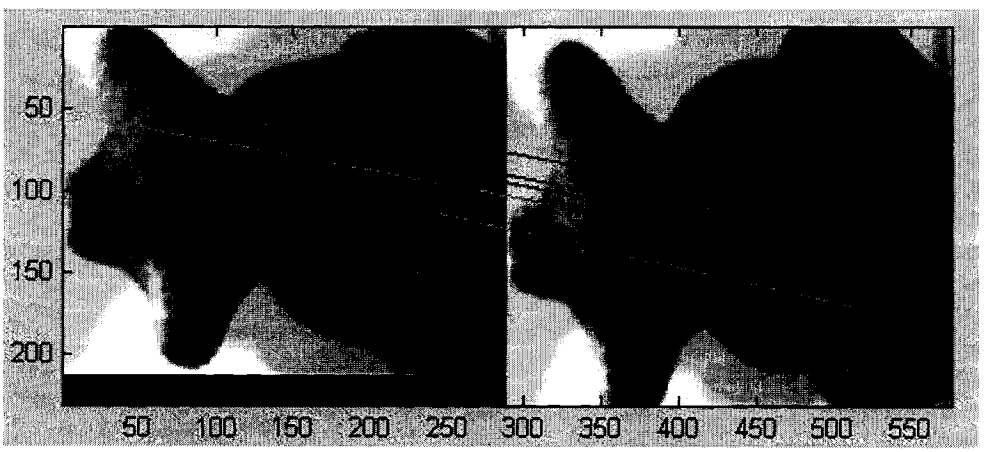
7. If the number of iterations t is larger than a threshold or $E(m^{(t)})$ *is smaller than a certain threshold, stop the loop. Else go back to step 2.*

4.1.3 Selection of Grid Points

To improve the performance of the algorithm, instead of using the complete set of valid Data Grid Points, we can select a smaller subset of those points and carry out the minimization procedure. We experimented with 2 ways of selecting a smaller subset of points:

4.1.3.1 Select the points around feature points

Let us recall that we already have some matching feature points by image processing the texture obtained during the process of texture image based Initial registration, and those feature point pairs are roughly matched. We can select 3D scan points around the feature points so those point pairs should be roughly correct as illustrated in Figure 41 and Figure 42.



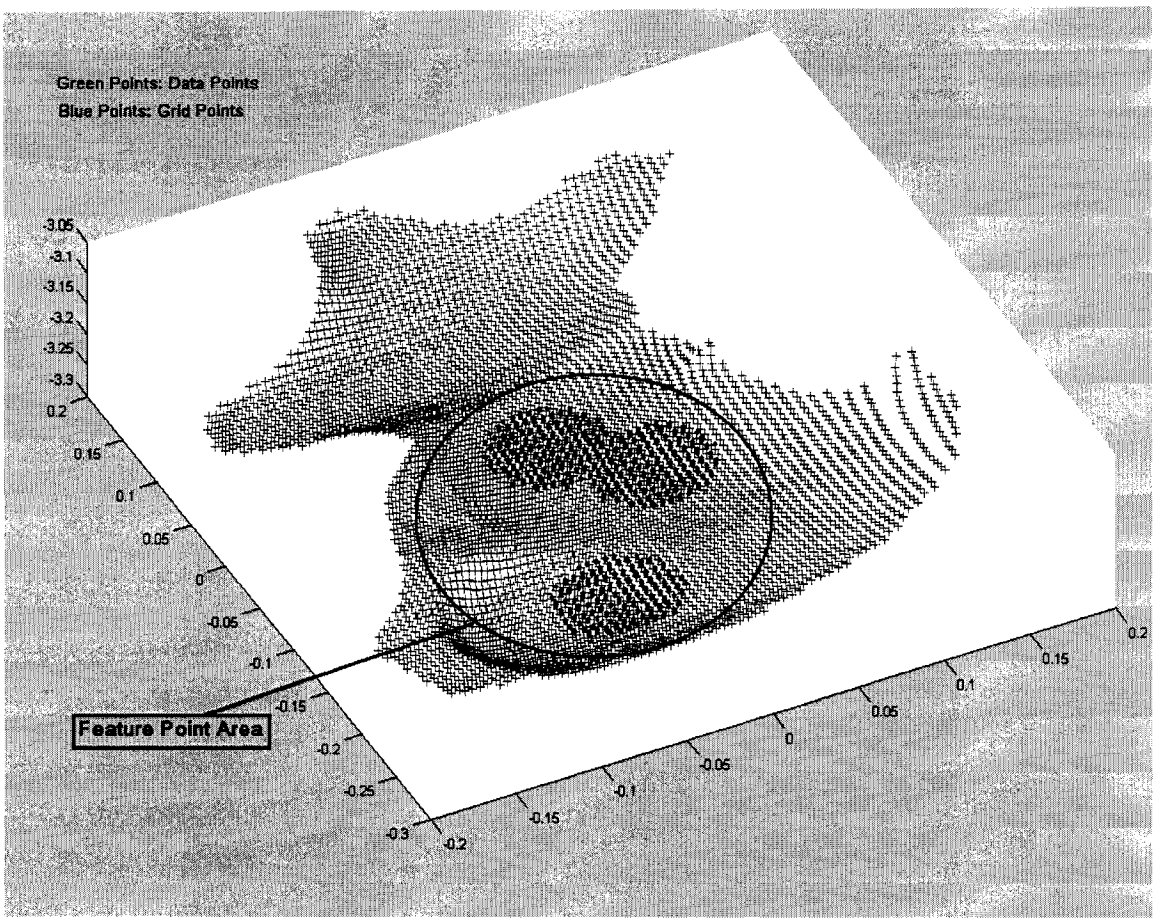


Figure 41: Selecting points surround feature points

Based on a number of experiments, we have concluded the following. The main advantage of this method is that the error converges very fast.

A major disadvantage of this method is that sometimes the feature points are too close to each other. As a result, the subset of points that we select for distance field matching covers only a small portion of the overlapping parts. This in turn implies that we only minimize the error of this small region, and in such cases, the resulting registration transformation is not accurate enough.

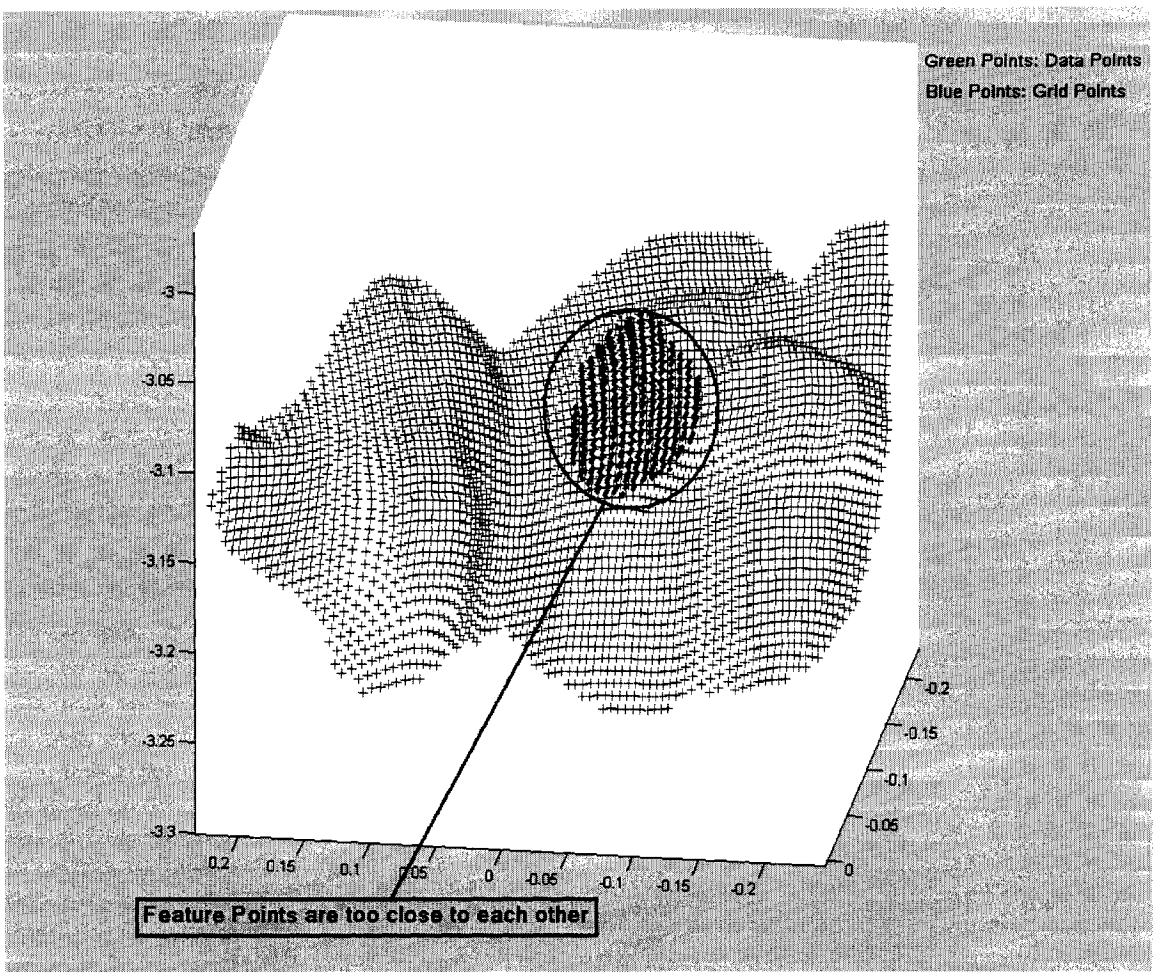


Figure 42: Bad selection of surround feature points

4.1.3.2 Randomly select points

To avoid the above drawback, we use a different strategy. We randomly select a subset of the Data Grid Point Set in each iteration. The advantage of this method is that we can eliminate any kind of bias and we also make the process independent of noise since we use different set of Grid Points in each iteration. And the error still converges very fast. The result of the registration process is very good.

4.1.4 Experimental Results

We choose different point sets to test our algorithm. Those data are from our 3D scanner and Stanford graphics lab. We scan the model from several different views to get the data sets. Each set scanned by using our 3D scanner contains around 4000 points. We first register the data by the Texture Based initial registration method pair-wisely to get the roughly registered data. Then, we apply our accurate registration method to those data. In each iteration, we randomly select 70 points to calculate the error metric. The number of the points might be adjusted. Choosing around 2% of the total amount of points in the scan seems to work in all the cases we have tried. We set the threshold for the iteration to be 100, but generally the number of iterations is below 10. We set the error threshold to be 1/10 of the distance between grid points. These values have been chosen through experimentation, there could be better values to further improve the performance.

4.1.4.1 Registration of 2 scanned data point sets

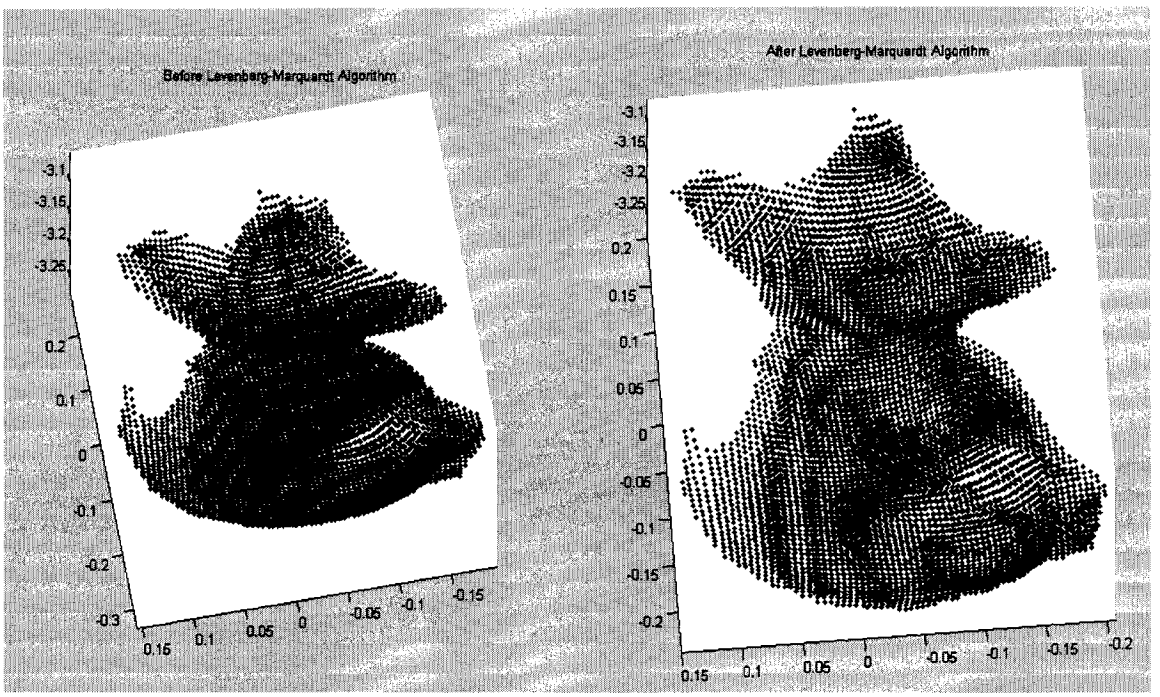


Figure 43: Registration of 2 scanned point sets (frog)

To test for noise independence, we introduced noise in the data sets by randomly selecting points from both Model and Data sets and adding random offset to the selected points. Figure 44 and Figure 45 show the results in the case of 10% noise.

4.1.4.2 Register 2 scanned data point sets with 10% noise

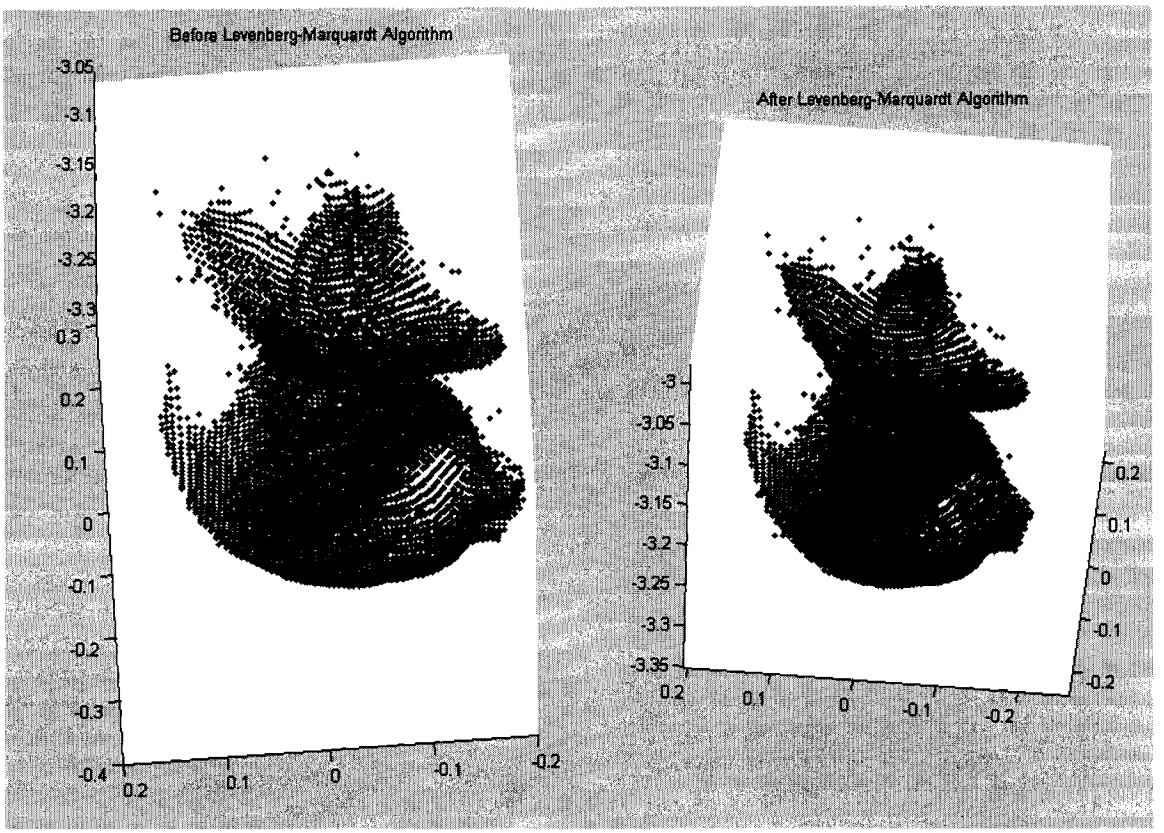


Figure 44: registration for 2 scanned point sets with 10% noise (frog)

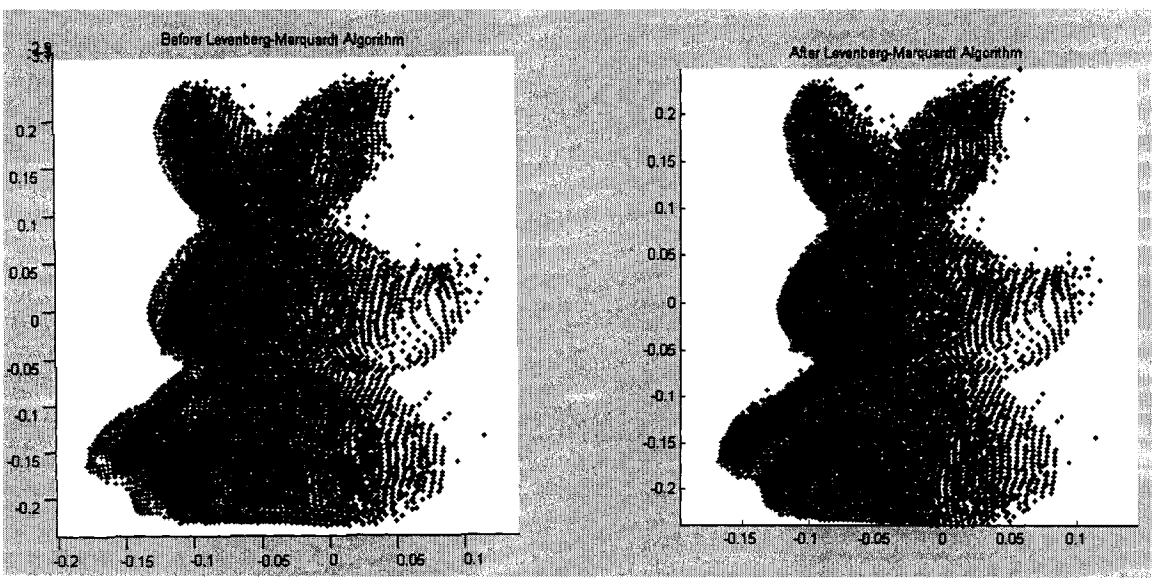


Figure 45: registration for 4 scanned point sets with 10% noise (rabbit)

4.2 Distance Function for multiple point sets registration

4.2.1 Overview

By using the error function above, we can solve the case with two scanned data point sets. In its present form, which is similar to the ICP it is rather difficult to extend the formula to multiple point sets registration. Therefore, we introduce a new form of the error function for multiple point sets registration. Like the previous algorithm, we still make one grid point set to be static. Again, we call it "Model Grid Point Set". Other grid point sets have their own varying transformation matrices. But this time, we take all the valid grid points inside "Model Grid Point Set" and calculate the interpolated distance value in other grid point set space. (Valid means it is within all the gird point set spaces and close to the scanned data point sets.) To get the interpolated distance value, for each iteration of Levenberg- Marquardt algorithm, we first update the transformation matrix for each data grid point set. Then we need to apply the reverse transformation of each moving point set to all the grid points inside the "Model Grid Point Set" so that we can get the interpolate distance value in different data fields. (See Figure 46)

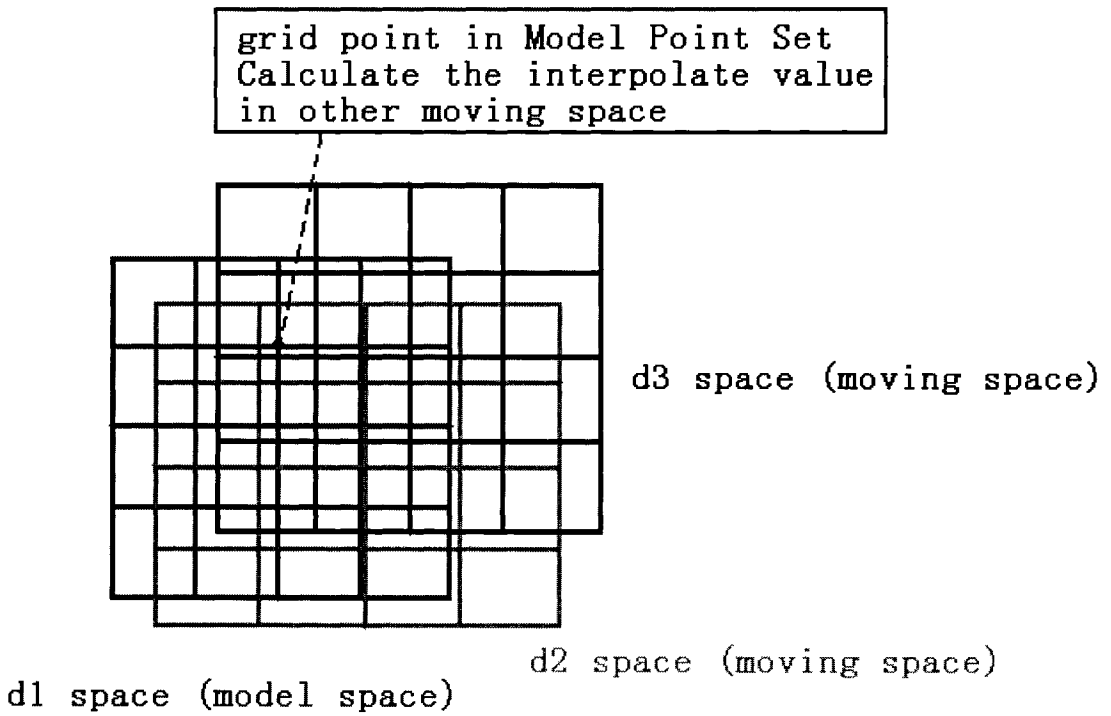


Figure 46: Model grid points in other spaces

4.2.2 Reverse Rotation and Translation

Rotation and Translation:

$$Rx = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-A) & -\sin(-A) \\ 0 & \sin(-A) & \cos(-A) \end{bmatrix}$$

$$Ry = \begin{bmatrix} \cos(-B) & 0 & \sin(-B) \\ 0 & 1 & 0 \\ -\sin(-B) & 0 & \cos(-B) \end{bmatrix}$$

$$Rz = \begin{bmatrix} \cos(-C) & -\sin(-C) & 0 \\ \sin(-C) & \cos(-C) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (4.9)$$

$$T = \begin{bmatrix} -Tx \\ -Ty \\ -Tz \end{bmatrix}$$

$$\begin{bmatrix} x_i' \\ y_i' \\ z_i' \end{bmatrix} = Rz * Ry * Rx(V + T)$$

V is the vector of x, y, z.

A, B, C are the three rotation angles around X axis, Y axis and Z axis respectively.

Tx, Ty, Tz are the three translation along X axis, Y axis and Z axis respectively.

4.2.3 Error Function

Now, we are using a new form of the error function for the Levenberg- Marquardt algorithms as defined below:

$$E = \sum_i \sum_k (d_{ik} - \bar{d}_i)^2 = \sum_i e_{ik}^2$$

$$e_{ik}^2 = (d_{ik} - \bar{d}_i)^2 \qquad (4.10)$$

$$\bar{d}_i = \frac{\sum_k d_{ik}}{k}$$

Accordingly, we need to calculate k Hessian Matrices A and k weighted gradient vectors b for each grid point.

$$a_{mnk} = \sum_i \sum_k \frac{\partial e_{ik}}{\partial x_m} \cdot \frac{\partial e_{ik}}{\partial x_n} \qquad (4.11)$$

$$b_{mk} = 2 \sum_i \sum_k \frac{\partial e_{ik}}{\partial x_m}$$

Here x is the vector of variables. In our case, it includes 3k rotations and 3k translations, where k is the number of data point sets in addition to the model point set.

4.2.4 Implementation

The complete registration algorithm consists of following steps:

1. Start of iteration.

2. Randomly select several grid points in Model Grid Set at location. Compute their corresponding positions in each Data Grid space after reverse translation and rotation.

3. Compute the interpolated Distance in each space $d_i(x_i, y_i, z_i)$

4. Give an initial variable vector m and compute the error

$$E_i(m) = \sum_i \sum_k (d_{ik} - \bar{d}_i)^2 = \sum_i e_{ik}^2$$

5. Compute the Jacobian Vector b and Hessian Matrix A.

6. Compute the update value Δm

(a) Let "m" be the variable vector: $m = [\alpha, \beta, \gamma, t_x, t_y, t_z]$

(b) Pick a modest value for λ*, say* $\lambda = 0.001$

(c) Solve the system of equation $\Delta m = (A + \lambda I)^{-1} b$ *and update the motion estimate*

$m_{trial} = m + \Delta m$

(d) Apply $m^{(t+1)}$ *to equation* $E(m + \Delta m)$

If $E(m + \Delta m) \geq E(m)$*, increase* λ *by a factor of 10 (or any other substantial factor).*

If $E(m+\Delta m) < E(m)$, *decrease* λ *by a factor of 10, update the trial solution*

$m^{(t+1)} = m^{(t)} + \Delta m$

7. If the number of iteration t is larger than a threshold or $E(m^{(t)})$ *is smaller than a certain threshold, stop the loop. Else go back to step 2.*

4.2.5 Extended Error Function

One of the drawbacks of the above error function is that we need to select the points only from the portion which has overlapping part with all other data point sets. We call these points "valid points". (See Figure 47)

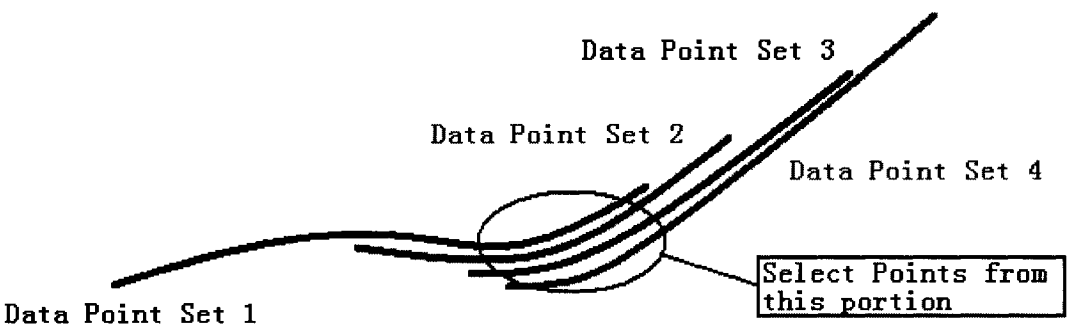


Figure 47: Example of registering 4 scanned data point sets

Our previously defined selection technique randomly selects points from the point set, and if the point is out of the overlapping portion, we simply discard it. If this overlapping portion is very small, it will take a large amount of time to find the valid set of points.

Furthermore, if the selected region of the points is very small, the registration will not be accurate.

To improve our algorithm, we extend the original error function for multiple scanned data point sets. We want to make use of other overlapping information for the registration. (See Figure 48)

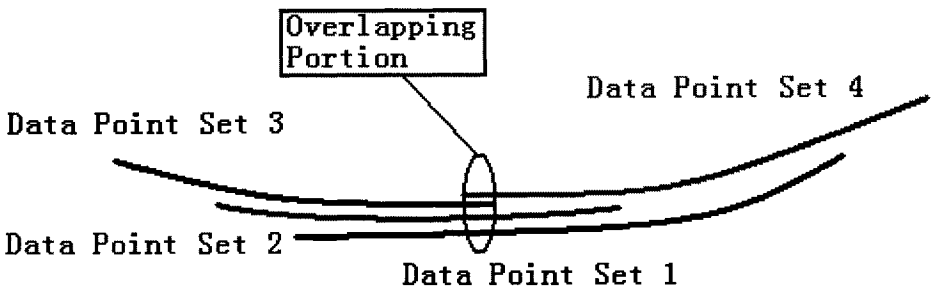


Figure 48: Example of registering 4 scanned data point sets with small overlapping portion

We rewrite our error function as follows:

$$E = \sum_{m}\sum_{i}\sum_{k}(d_{mki} - \bar{d}_{mi})^2 = \sum_{i} e_{mik}^2 \qquad (4.12)$$

"m" indicates the index of the overlapping portion.

For example, in the upper case, instead of using the points from the overlapping portion with 4 point sets, we use two sets of points from two overlapping portion: (See Figure 49)

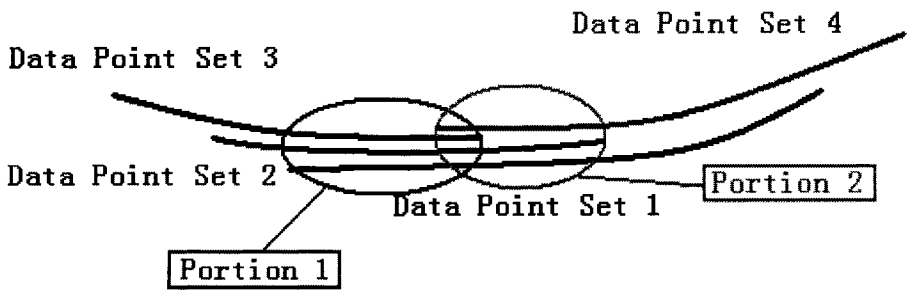


Figure 49: Select points in two portions

Now, the error function is given as below:

$$E = \sum_{i}\sum_{m}(d_{mi} - \bar{d}_i)^2 + \sum_{j}\sum_{n}(d_{nj} - \bar{d}_j)^2 \qquad (4.13)$$

"i" indicates the points in "Portion 1", "m" indicates the index of sets (1, 2, 3).

"j" indicates the points in "Portion 2", "n" indicates the index of sets (1, 2, 4).

4.2.6 Grouping Registration Approach

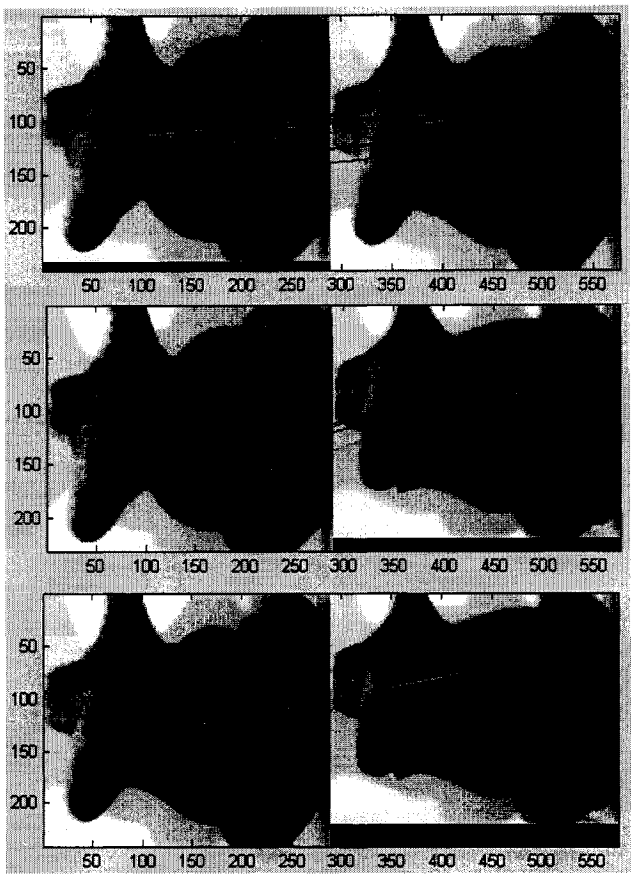
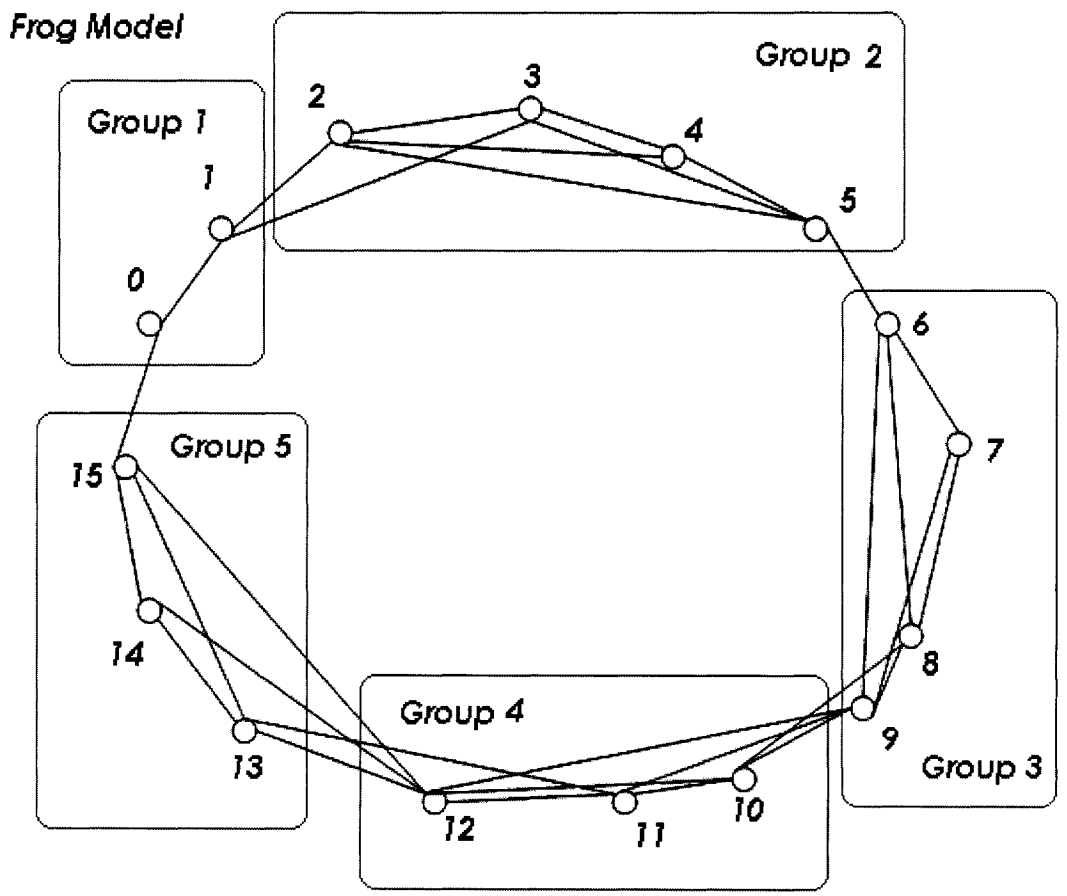
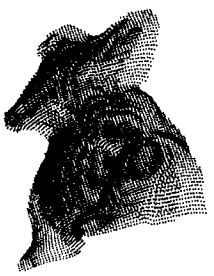


Figure 50: Feature matches among scans based on texture images for group 4 of Figure 3a. Top : scan#10 – scan#11; Middle: scan#10 – scan#12; Bottom: scan#11 – scan#12.

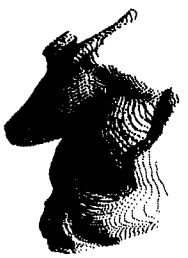
An important point to note is that in covering the complete surface of an object, only subsets of views will have common overlap. Hence, it becomes necessary to first determine the groups of view subsets which have common overlap. Since all the point data sets in one group will have one overlapping portion they can be registered together. We use the results from our image processing technique for grouping. (See Figure 50) Within each group, we apply the previously defined simultaneous registration procedure. This, we follow with pair-wise registration to create a single complete 3D model from the multiple scan data sets. (Figure 51)


Frog Model
Group 1
Group 2
Group 3
Group 4
Group 5
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

(a)



Step1: Register 0, 1 to give partial model m1



Step2: Register 2, 3, 4, 5 to give partial model m2

Step3: Register 6, 7, 8, 9 to give partial model m3

Step4: Register 10, 11, 12 to give partial model m4

Step5: Register 13, 14, 15 to give partial model m5

Step6: Register m1, m2 to give partial model m12

Step7: Register m3, m4 to give partial model m34

Step8: Register m12, m34 to give partial model m1234

(b)

(c)

Figure 51: (a) Graph showing overlap cliques; (b) subgroup registration examples; (c) final registered model consisting of all scan data.

4.2.7 Building the distance field

In order to implement the distance function, we need to build a distance field so that each grid point has a distance value in its own space. We have two approaches:

4.2.7.1 Building in pre-processing time

We first build the grid for each point set with a higher resolution than the point set. To get the distance value for each grid point, for each grid point, we calculate the distance value

value from the grid point to every point in the point set. If the smallest distance value is larger than a certain threshold, we discard this grid point. Otherwise, we save the smallest distance as the distance value for this grid point. When we need the distance value of the Model Grid Point in other spaces, we need to interpolate the distance value of the grid points in Data Grid Point Set.

The processing will take relatively long time, but after it is done, this distance information is reusable. The speed of processing distance function for determining error etc. will be very fast.

4.2.7.2 Building in runtime

Since we do not use every grid point in processing the algorithm, the above approach will waste large amount of memory and computation time. To avoid this, instead of pre-processing and obtaining distance field values at all the data grid points, we compute the distance value for the model grid points in other spaces, as needed. By using this method, there is no need to interpolate the distance value in other space, because we calculate the distance value in each space directly.

4.2.8 Experimental Results

4.2.8.1 Speed Comparison

Compared to the ICP algorithm, our approach is faster and more efficient. The following shows the test data results: (ICP implementation is downloaded from http://www.csse.uwa.edu.au/~ajmal/code/icp2.m)

One benefit of using our algorithm is that in general, we have observed that our method converges faster than ICP in terms of the number of iterations needed to arrive at optimum solution. Even in case of pair-wise registration, since we use Levenberg-Marquardt method which is known to have better convergence properties than other algorithms, our algorithm requires less number of iterations. We randomly selected 20 pairs of overlapping scans from different models and collected the statistics for number of iterations. The following graph shows comparison of the number of iterations required by ICP and the number required by our method. (See Figure 52)

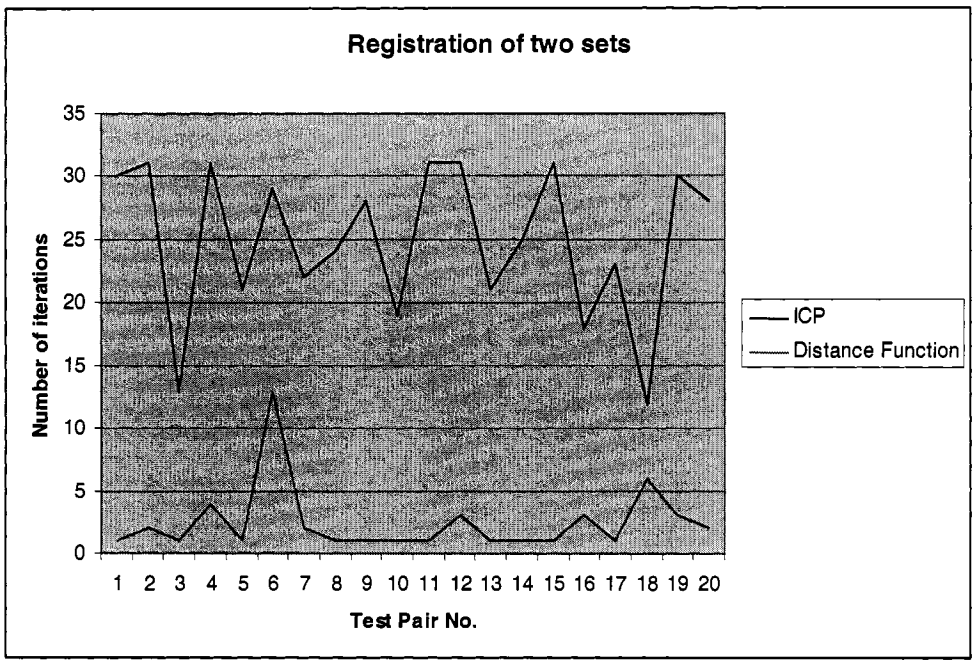


Figure 52: Graph showing total number of iterations required for pair wise registrations.

We require even fewer iterations when we carry out simultaneous multi-scan registrations using our method. Again, we randomly selected 10 sets, each with 4 overlapping data sets. Then we used ICP in a pair-wise chain fashion, 3 registrations per set and counted the total number of iterations. We also registered each set in a single multi-scan registration

operation using our method. The following graph shows the results of these tests (See Figure 53).

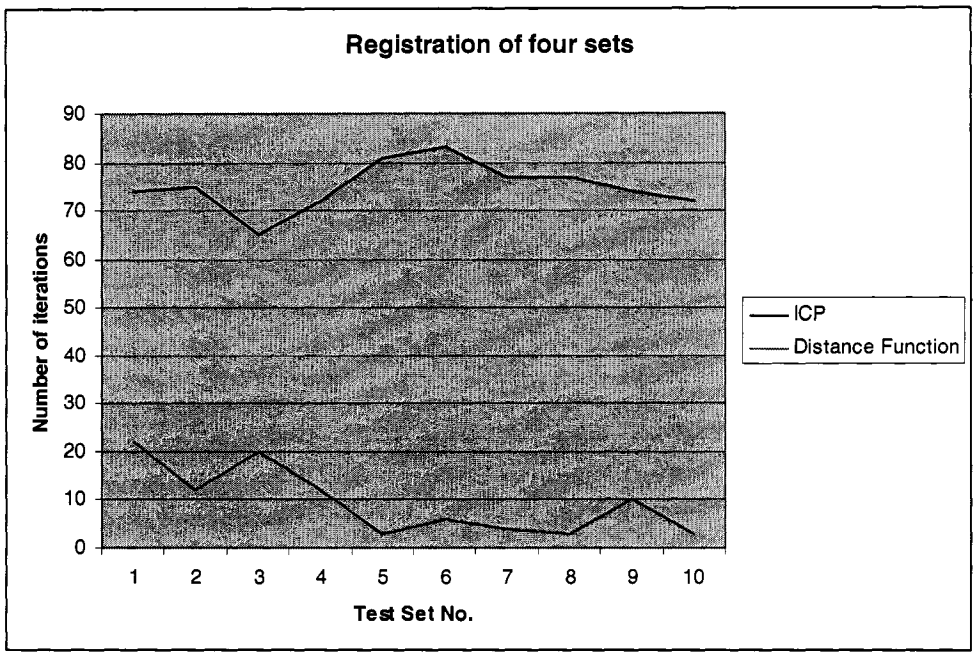


Figure 53: Graph showing total number of iterations required in case of multiple registrations.

We also compared the final registration accuracy of our results with pair wise ICP algorithm. For comparison, we used the residual ICP error metric where it is defined as sum of distance between corresponding points. In all cases, our MSE was less than ICP (See Table 3). We also compared the time taken to perform full registration. Most of the times, our method was around 10-20% faster than ICP based pair-wise chain registration, even for models with dozens of scans. In cases, of hundreds of scans the benefit will be significantly more, as the total number of registration operations reduces significantly in comparison to pair-wise registration.

Testset1 (8 sets: 31031 points)	Number of Registration	Mean Square Error	Time (Second)
ICP	7	0.0028	60.521
Our Algorithm	3	0.0016	49.011
Testset2 (16 sets: 66515 points)			
ICP	15	0.0028	117.077
Our Algorithm	7	0.0018	100.025
Testset3 (7 sets: 25063 points)			
ICP	6	0.0022	41.385
Our Algorithm	2	0.0013	25.569
Testset4 (8 sets: 30412 points)			
ICP	7	0.0021	52.852
Our Algorithm	3	0.0013	29.126
Testset5 (3 sets: 116376 points)			
ICP	2	0.000817	292.297
Our Algorithm	1	0.000743	245.062
Testset6(4sets: 13457 points)			
ICP	3	0.0024	21.141
Our Algorithm	1	0.0015	14.586
Testset7(4sets: 15400 points)			
ICP	3	0.0031	24.599
Our Algorithm	1	0.0015	24.479
Testset8(7sets: 24266 points)			
ICP	6	0.0021	38.701
Our Algorithm	2	0.0014	25.594

Table 3: Time comparison

4.2.8.2 Visualized Result

For registering 2 scanned data point sets: (See Figure 54, Figure 55)

Figure 54: Registration for 2 scanned point sets (frog)

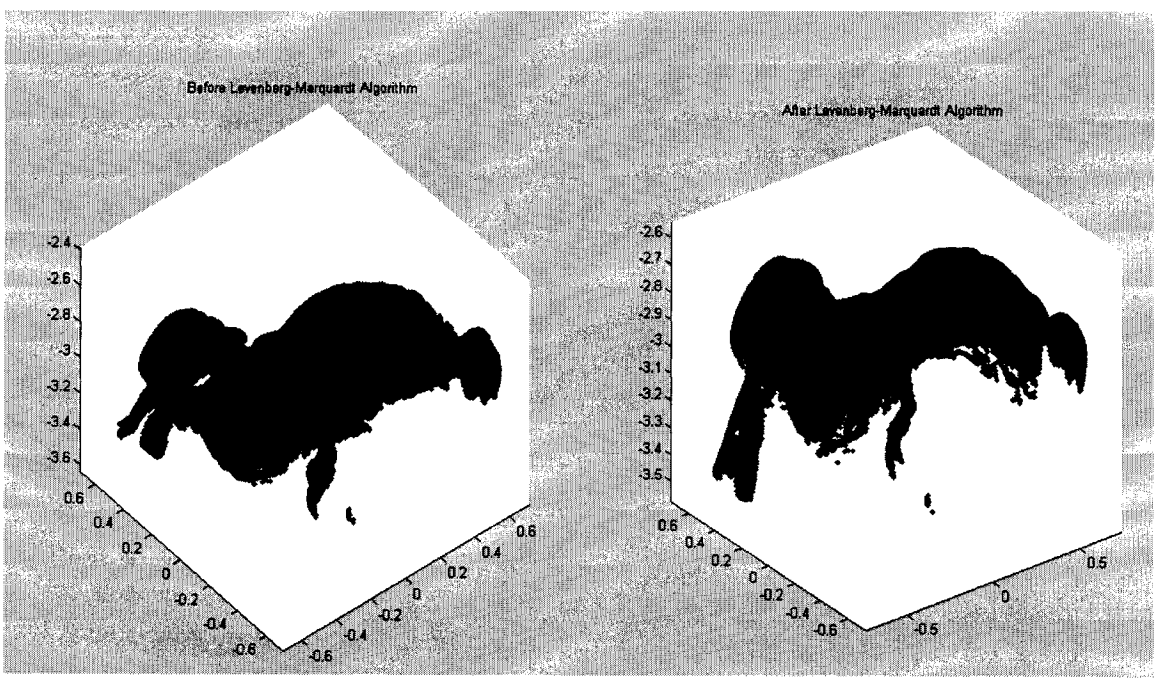


Figure 55: Registration for 2 scanned point sets (bunny)

Following images are using Distance Function Algorithm to register 3 scanned data point sets: (See Figure 56, Figure 57 and Figure 58)

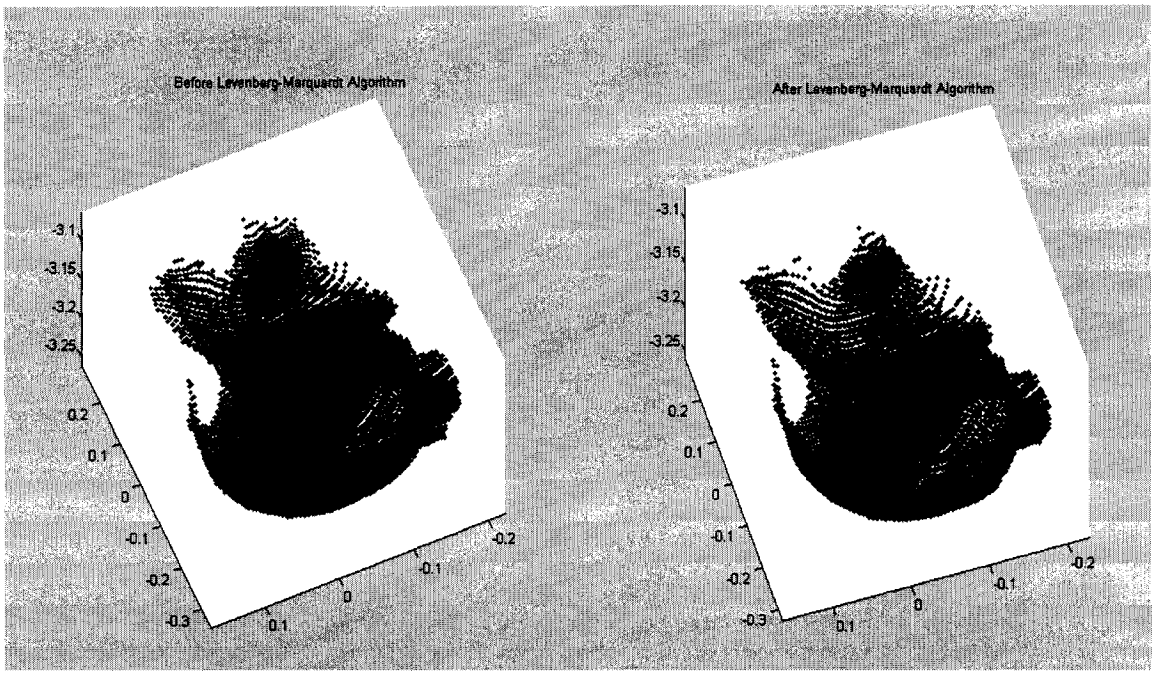


Figure 56: Registration for 3 scanned point sets (frog)

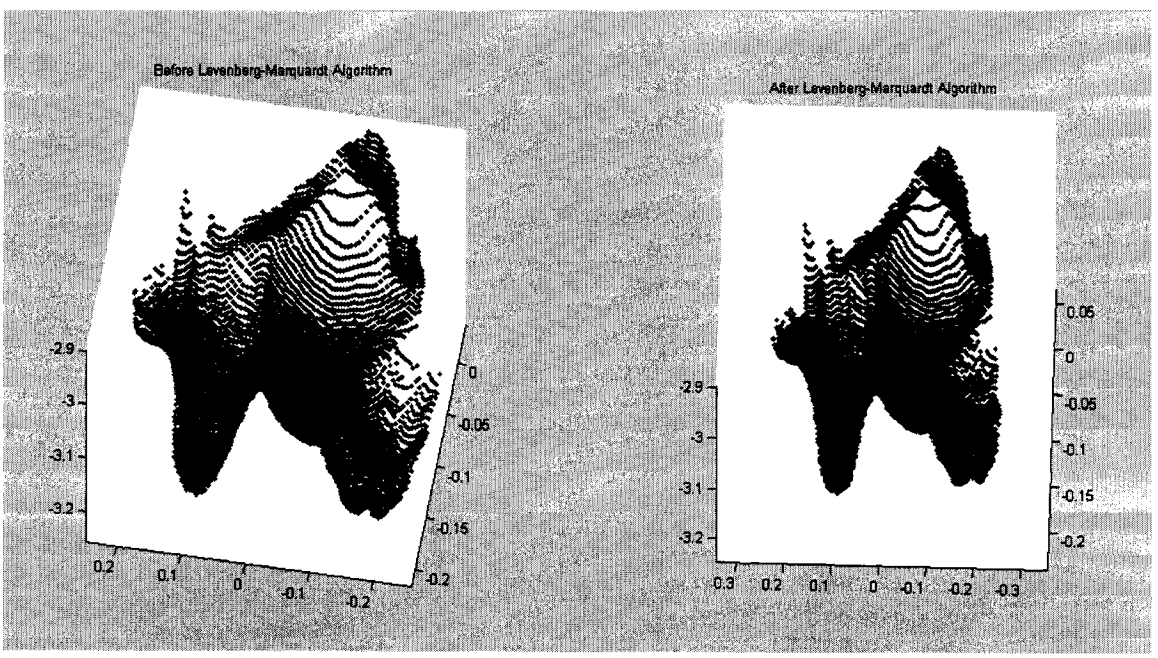


Figure 57: Registration for 3 scanned point sets (frog)

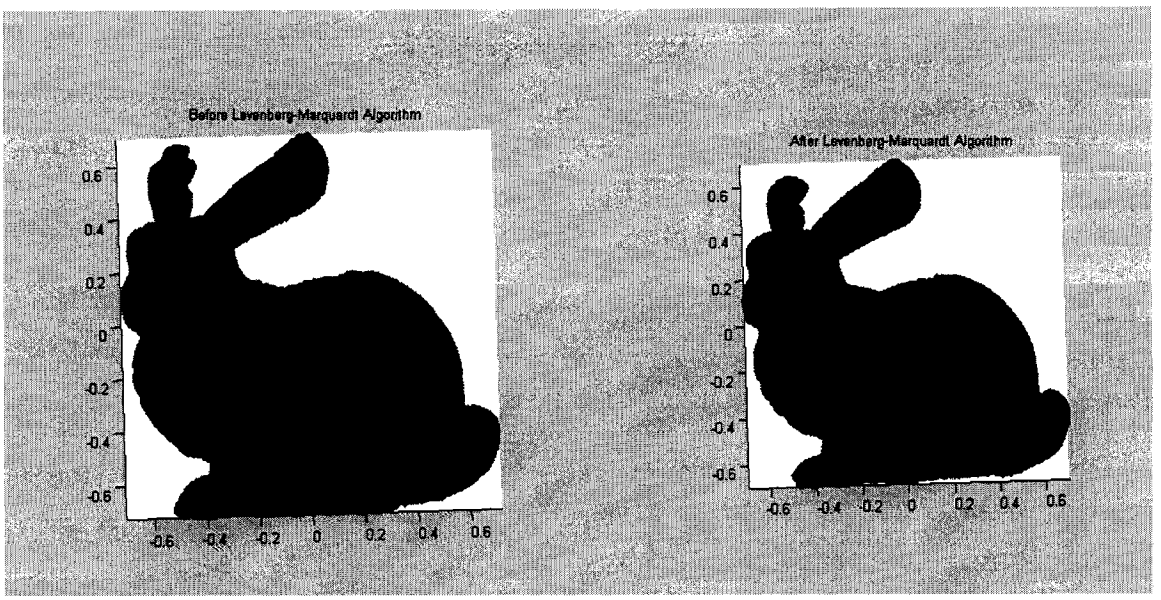


Figure 58: Registration for 3 scanned point sets (bunny)

Following images are using Distance Function Algorithm for 4 scanned data point sets:

(See Figure 59, Figure 60)

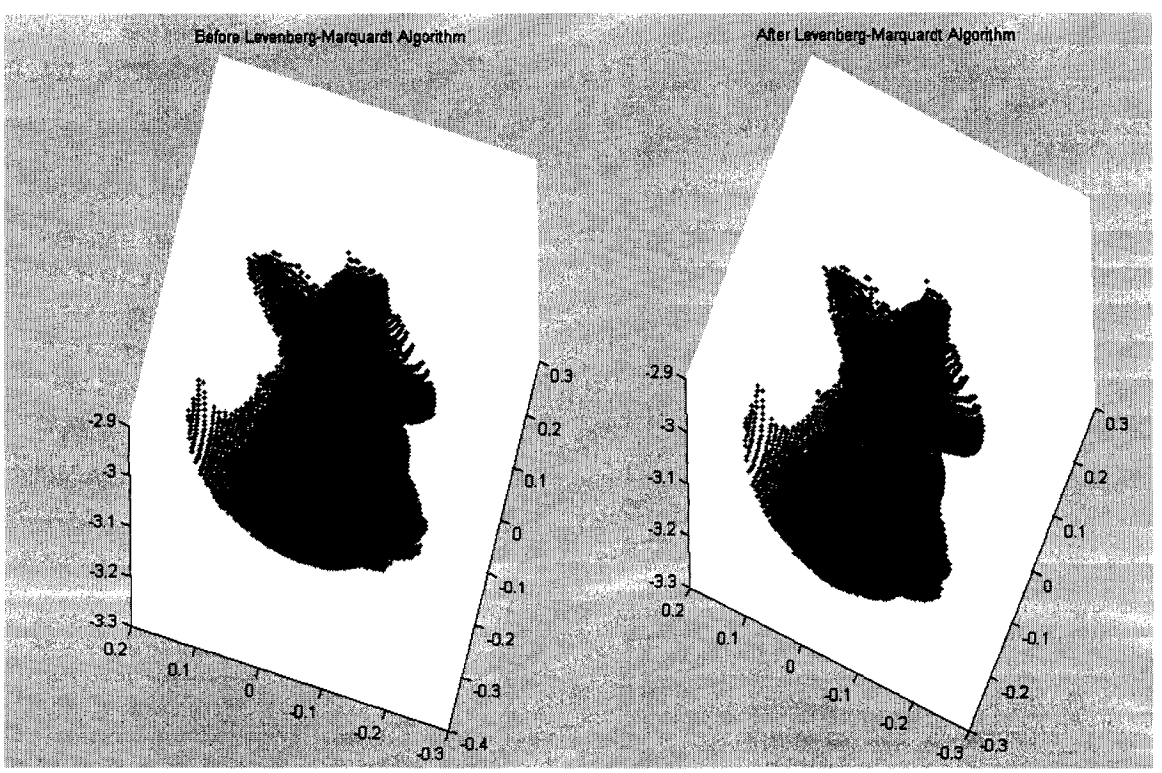


Figure 59: Registration for 4 scanned point sets (frog)

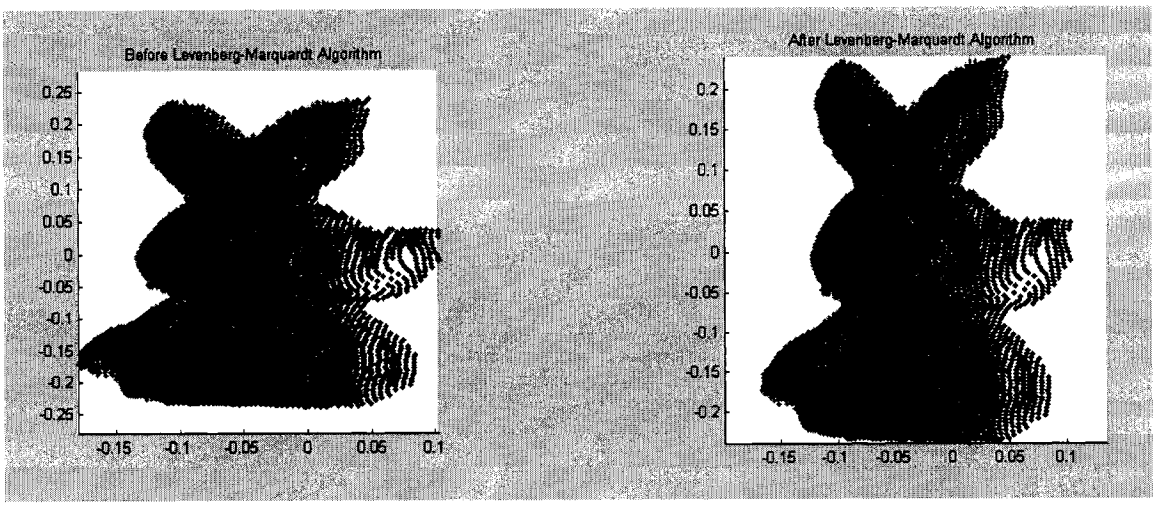


Figure 60: Registration for 4 scanned point sets (rabbit)

Following images are using Distance Function Algorithm for 4 scanned data point sets (Frog Model): (See Figure 61)

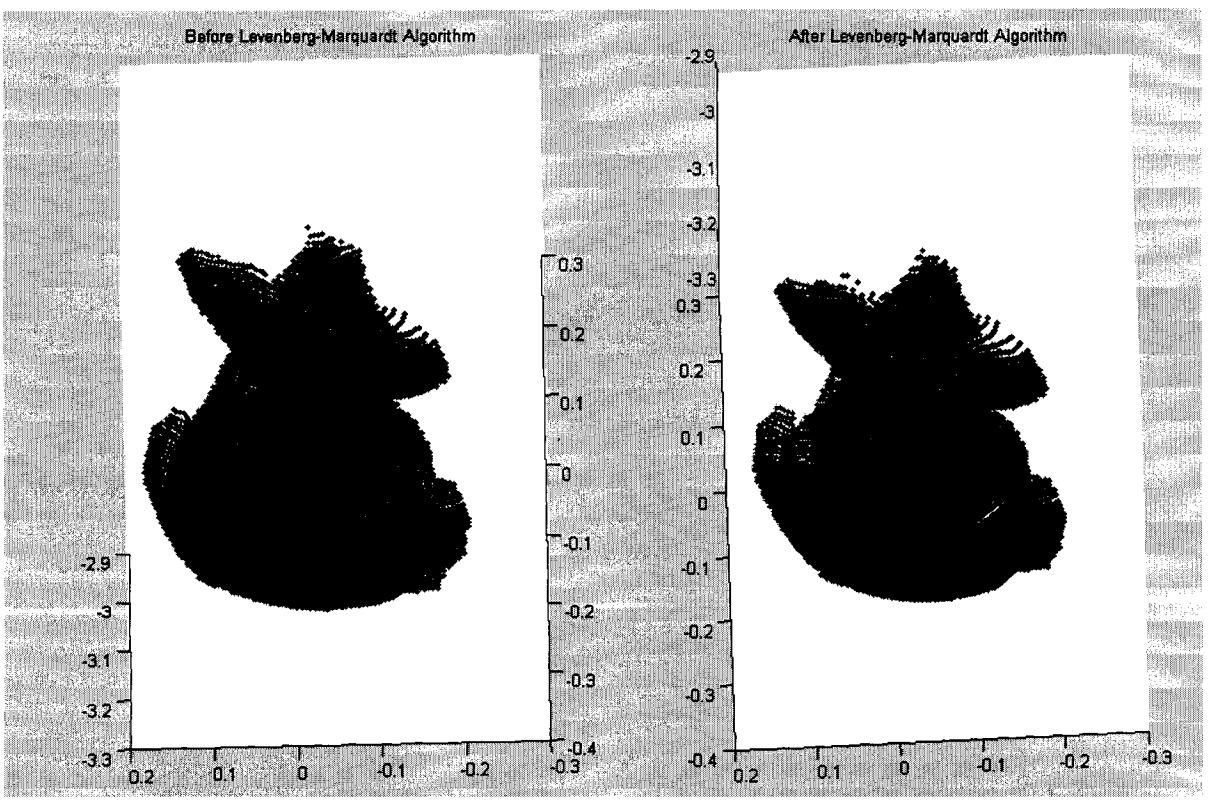


Figure 61: registration for 4 scanned point sets (frog)

Following images are using Distance Function Algorithm for 5 scanned data point sets (Frog Model): (See Figure 62)

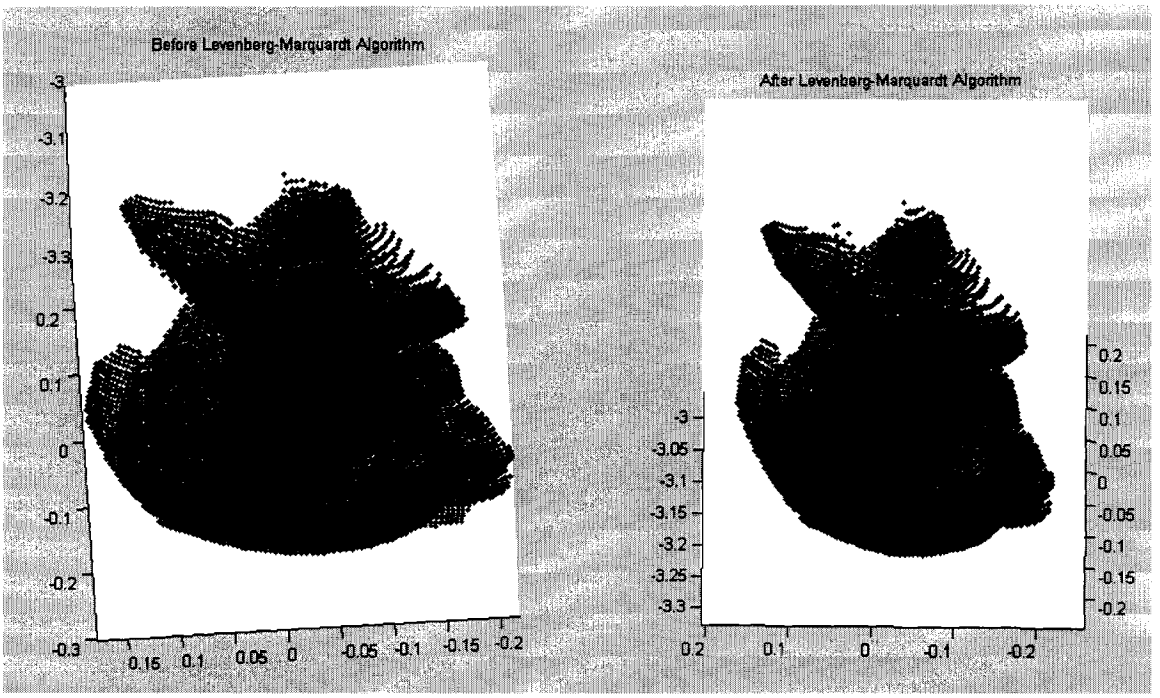


Figure 62: registration for 5 scanned point sets (frog)

Following images are using Distance Function Algorithm for 16 scanned data point sets (Frog Model): (See Figure 63)

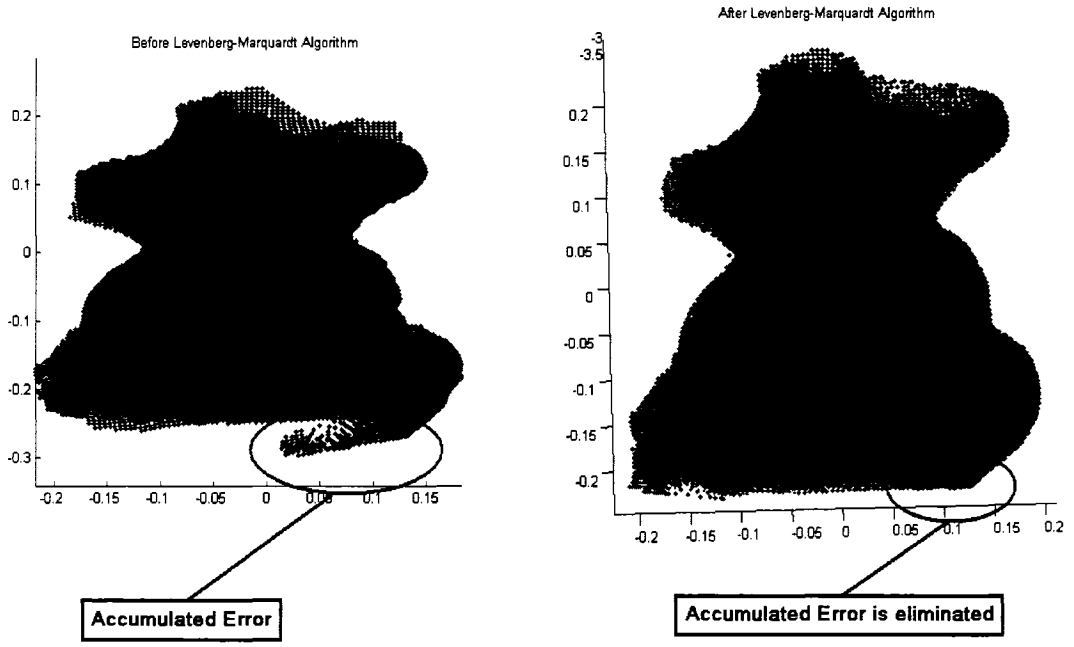


Figure 63: registration for 16 scanned point sets (frog)

Following images are using Distance Function Algorithm for 24 scanned data point sets (Rabbit Model): (See Figure 64, Figure 65)

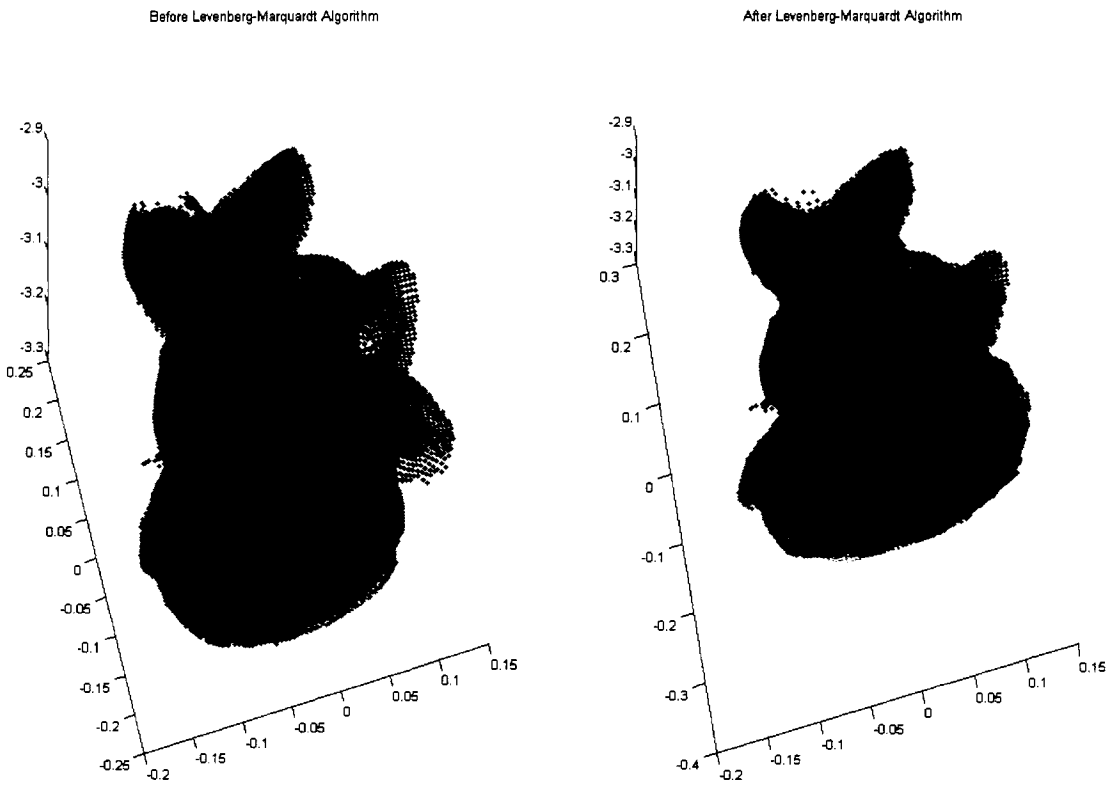


Figure 64: registration for 24 scanned data point sets (rabbit from side)

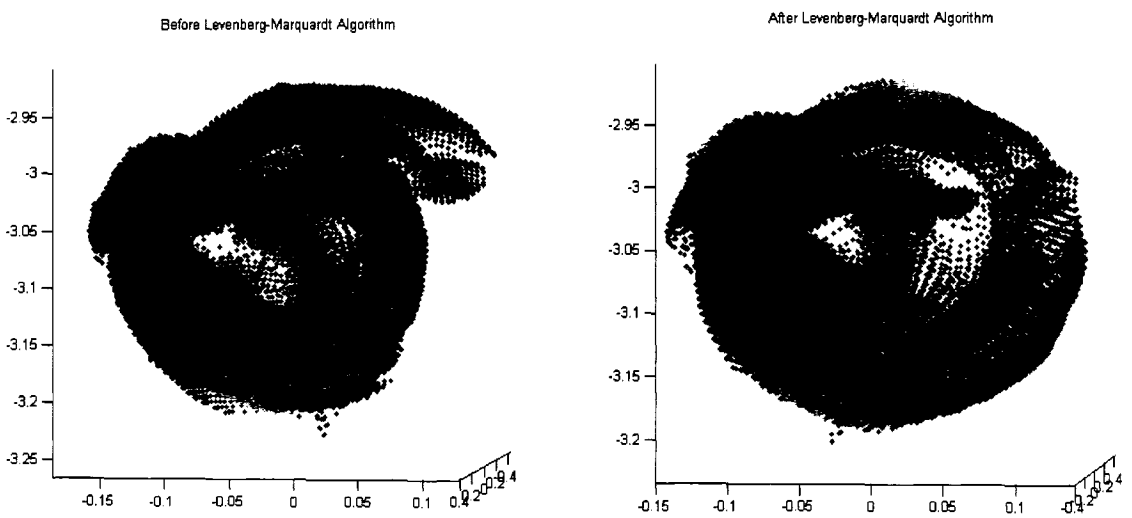


Figure 65: registration for 24 scanned data point sets (rabbit from the bottom)

Following image is using Distance Function Algorithm for 7 scanned data point sets (Bunny Model): (Figure 66)

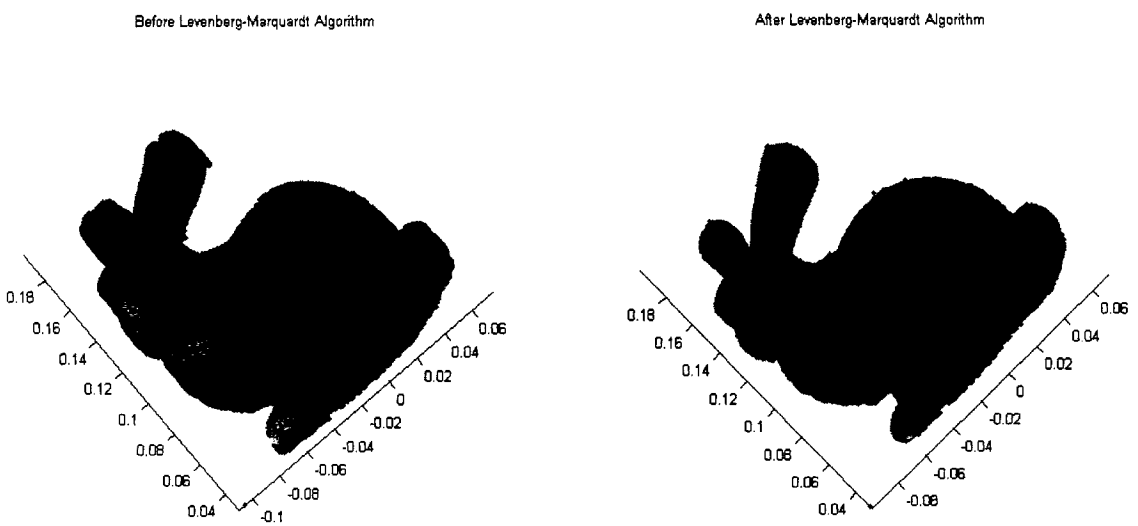


Figure 66: registration for 7 scanned data point sets (bunny)

4.2.9 Conclusion

In this chapter, we introduced a new method for registering multiple scanned data point sets simultaneously by aligning common portion of all the overlapped sets. We compared our approach with the pair-wise approach through a number of experiments which show the significant performance improvements we have been able to achieve through our approach. Our method is both more accurate and faster than the ICP approach.

Chapter 5: Conclusions and Future Work

Rapid creation of geometric models of 3D objects is the need of many different applications today both in the entertainment and the engineering sectors. The research reported in this thesis lies primarily in this domain. Low cost, simple to use and high accuracy 3D scanners are becoming available. These scanners provide a convenient mechanism to acquire the 3D geometry of existing objects, even with complex surface geometries. While different scanning technologies are in place, all scanners acquire 3D data in the format of sample points of the object's surface "visible" to the scanner head. Most scanners operate using a single scan head. In order to acquire the 3D geometry of the complete surface of an object, scanning from multiple views becomes essential. Each view presents a different portion of the object's surface to the scan head. Even when multiple scan head systems are used, complex surfaces would have self-occlusions that would mandate scanning from different views. Since each view is in its own coordinate system, the scan data sets from multiple views have to be merged together to create the 3D geometry of the entire surface of the object in a single coordinate system. The most important problem to be addressed in this task is the accurate registration of all the views with respect to each other. This has formed the main problem researched by us and reported in this thesis.

In this thesis, we have proposed a new approach for simultaneously registering multiple scanned point data sets. Our approach is based on matching overlapping portions of 3D distance fields defined for each scan data set. Compared to existing methods, our

approach is faster and more efficient. Since most of the work is done automatically, it considerably reduces human operator work, and in the process it also reduces mistakes that could be typically made by the human operator. It is very useful, especially for registering large number of scanned point sets. The advantages and disadvantages of our approach are listed below.

5.1 Advantages

5.1.1 Initial Registration

<1> We use both texture based and eigen test based technique for the initial registration. This reduces human interaction during the first step.

<2> The texture based approach is very useful, when surface color based features are more prominent than geometric features and the Eigenvalue analysis based technique is more useful when geometric features are prominent. Combining these techniques yields better Initial registration results in more cases, then using just one of them.

5.1.2 Accurate Registration for multiple sets

<1> We can simultaneously register all the datasets. This avoids the common problems of pair-wise registration algorithms: unnecessary work and accumulated errors.

<2> Since conceptually, we embed one signed distance field into another signed distance field and these are defined everywhere, we completely avoid the need for explicitly establishing point to point or point to surface correspondence. We can also estimate the error using much fewer points in the field, thus making the algorithm much faster.

<3> We match distance fields and not sampled surface points. Since partial differentials of any order are now available in the field, we can make use of robust and fast converging numerical techniques to search for the optimal solution. Further, the ability to randomly select a small number of distance field points for error estimation makes the algorithm both robust and fast. Note that this is difficult if explicit correspondences are to be searched in each iteration, as would be the case for point based alignment.

5.2 Disadvantages

5.2.1 Initial Registration

<1> Since the texture based approach uses the captured image of the object, it is very sensitive to lighting. It may be noted that color captured depends both on inherent object color and global illumination effects.

<2> We use an image processing approach to extract model texture from the back ground. This again is very sensitive to shadowing effects. Different lighting conditions need different thresholds to remove the effects of shadowing.

5.2.2 Accurate Registration

<1> We group the sets manually in our experiment. It can also be attempted automatically by letting the program try all the combinations and decide how to group the sets. But it is not guaranteed to find the best combination of groups.

5.3 Future work

There are still a number of avenues to explore this work further. A few of these are listed below.

- The method of extracting object's texture from the background highly depends on the working environment, especially the algorithm to remove the shadow. In the future work, it is better to find a more general way of extracting the texture from the background.

- During our research, we tried out a specific algorithm for selecting up to 4 matching feature points. But in some extreme cases, this algorithm fails to find adequate number of matching feature pints. It is necessary to work out more robust techniques to select color based feature points.

- Grouping strategy is not automatic. We could develop a better approach for grouping sets.

- The algorithm for multiple scanned data point sets registration can only be applied when at least part of the static scanned data point sets is overlapping with other scanned data point sets. Ideally, we would like to extend this algorithm to deal with all the data sets in one single registration operation.

References

[Blais 1995] Gtrard Blais, Martin D. Levine, *Registering Multiview Range Data to Create 3D Computer Objects*, IEEE Trans. PAMI, Vol. 17 no. 8, pp. 820-824, 1995

[Besl 1992] P. Besl and N. McKay. *A method for registration of 3-D shapes.* IEEE Trans. PAMI, Vol. 14, No 2, pp. 235-256, 1992

[Chen 1991] Chen, Y. and Medioni, G., *Object Modeling by Registration of Multiple Range Images*, Proc. IEEE Conf. on Robotics and Automation, Vol. 3, pp. 2724-2729, 1991

[Chen 1998] Chen, C., Hung, Y., and Cheng, J. *A Fast Automatic Method for Registration of Partially-Overlapping Range Images*, Proc. ICCV, pp.242-248, 1998

[Chen 1999] Chen, C., Hung, Y., and Cheng, J. *RANSAC-Based DARCES: A New Approach to Fast Automatic Registration of Partially Overlapping Range Images*, Trans. PAMI, Vol. 21, No. 11, pp. 1229-1234, 1999

[Chetverikov 2005] Dmitry Chetverikov, Dmitry Stepanov, *Robust Euclidean alignment of 3D point sets: the Trimmed Iterative Closest Point algorithm*, Image and Vision Computing, Vol. 23, No. 3. pp. 299-309, 2005

[David 2004] David G. Lowe, *Distinctive Image Features from Scale-Invariant Keypoints*, International Journal of Computer Vision, Vol. 60, No. 2, P91-110, 2004

[Dorai 1997] Dorai, C., Weng, J., and Jain, A. *Optimal Registration of Object Views Using Range Data*, Trans. PAMI, Vol. 19, No. 10, pp. 1131-1138, 1997

[Dorai 1998] Chitra Dorai, Gang Wang, Anil K. Jain, Carolyn Mercer, *Registration and Integration of Multiple Object Views for 3D Model Construction*, IEEE Trans. PAMI, Vol. 20, No.1, pp. 83-89, 1998

[Faugeras 1986] Faugeras, O. and Hebert, M. *The Representation, Recognition, and Locating of 3-D Objects*, The International Journal of Robotics Research, Vol. 5, No.3, pp. 27-52, 1986

[Fitzgibbon 2001] Andrew W. Fitzgibbon, Robust *Registration of 2D and 3D Point Sets*, In Proc. British Machine Vision Conference, Vol. 2, pp411-420, 2001

[Holt 2001] Olaf Hall-Holt, Szymon Rusinkiewicz, *Stripe Boundary Codes for Real-Time Structured-Light Range Scanning of Moving Objects*, Eighth International Conference on Computer Vision, 2001

[Huber 2003] Daniel F. Huber, Martial Hebert, *Fully automatic registration of multiple 3d data sets*, Image and Vision Computing, Vol. 21, No. 7, pp. 637-650, 2003

[Johnson 1996] Andrew Johnson, Sing Bing Kang, *Registration and Integration of Textured 3D data*, Image and Vision Computing, Vol. 17, No. 2, pp.135-147, 1996

[Johnson 1997] Johnson, A. and Hebert, M. *Surface Registration by Matching Oriented Points*, Proc. 3DIM, pp. 121-128, 1997

[Krishnan 2005] Shankar Krishnan, Pei Yean Lee, John B. Moore, Suresh Venkatasubramanian, *Global Registration of Multiple 3D Point Sets via Optimization-on-a-Manifold*, Eurographics Symposium on Geometry Processing, 2005

[Lomonosov 2004] Evgeny Lomonosov, Dmitry Chetverikov, Aniko Ekart, *Fully automatic, robust and precise alignment of measured 3D surfaces for arbitrary*

orientations, Proc. 28th Workshop of the Austrian Association for Pattern Recognition, Hagenberg, pp. 39-46, 2004

[Mitra 2004] Niloy J. Mitra, Natasha Gelfand, Helmut Pottmann, and Leonidas Guibas, *Registration of Point Cloud Data from a Geometric Optimization Perspective*, Eurographics Symposium on Geometry Processing, 2004

[Natasha et al. 2005] Natasha Gelfand, Niloy J. Mitra, Leonidas J. Guibas, Helmut Pottmann. *Robust Global Registration*, Eurographics Symposium on Geometry Processing, 2005

[Neugebauer 1997] Peter J. Neugebauer, *Geometrical Cloning of 3D Objects via Simultaneous Registration of Multiple Range Images*, Proceedings of 1997 International Conference on Shape Modeling and Applications, 1997

[PENNEC 1996] Xavier PENNEC, *Multiple Registration and Mean Rigid Shapes: Application to the 3D case*, In 16th Leeds Annual Statistical Workshop, pp. 178-185

[Pfister 2004] Pfister, H., Gross, M. *Point-Based Computer Graphics.* Computer Graphics and Applications, IEEE Vol. 24, pp. 22 – 23, 2004

[Press et al. 1992] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, England, second edition, 1992.

[Pulli 1999] Kari Pulli, Multiview *Registration for Large Data Sets*, 3-D Digital Imaging and Modeling. Second International Conference, 1999

[Richard 1994] Richard Szeliski, *Image Mosaicing for Tele-Reality Applications, Cambridge Research Laboratory*, Tech Rep: 94-2, 1994

[Rusinkiewicz 2001] Szymon Rusinkiewicz, Marc Levoy, Efficient Variants of the ICP Algorithm, Third International Conference on 3D Digital Imaging and Modeling, 2001

[Sharp 2001] Gregory C. Sharp, Sang W. Lee, David K. Wehe, *Toward Multiview Registration in Frame Space*, IEEE ICRA, Vol.4, pp. 3542-3547, 2001

[Sharp 2004] G.C. Sharp, S.W. Lee, D.K. Wehe, *Multiview Registration of 3D Scenes by Minimizing Error between Coordinate Frames*, Trans. PAMI, Vol. 26, No. 8, pp. 1037 – 1050, 2004

[Shih 2006] Sheng-Wen Shih, Yao-Tung Chuang, Tzu-Yi Yu, *Relaxation of Multiview Registration Error*, National Chi Nan University

[Simon 1996] David A. Simon, *Fast and Accurate Shape-Based Registration*, doctoral dissertation, tech. report CMU-RI-TR-96-45, Robotics Institute, Carnegie Mellon University

[Stein 1992] Stein, F. and Medioni, *G. Structural Indexing: Efficient 3-D Object Recognition*, Trans. PAMI, Vol. 14, No. 2, pp. 125-145, 1992.

[Turk 1994] Greg Turk, Marc Levoy, *Zippered Polygon Meshes from Range Images*, Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, pp. 311-318, 1994

[Weik 1997] S. Weik, *Registration of 3-D partial surface models using luminance and depth information*, First International Conference on Recent Advances in 3-D Digital Imaging and Modeling (3DIM '97) , pp. 93, 1997

[Williams 2000] J.A. Williams, M. Bennamoun, *Simultaneous Registration of multiple point sets using orthonormal matrices*, IEEE International Conference on Acoustics, Speech, and Signal Processing, 2000